Instituto de Computación
Facultad de Ingeniería
Universidad de la República

Proyecto de Grado

Extensiones al protocolo EPP para el sistema de registro de dominios .uy

Autores:
Pablo Cal
Ernesto Pin

Tutor:
Pablo Rodríguez

Usuario responsable:
Sergio Ramírez – SeCIU

Tribunal corrector:
Carlos Martínez
Pedro Piñeyro
Leonardo Vidal

Mayo de 2011
Montevideo, Uruguay

Resumen

El *Extensible Provisioning Protocol* (Protocolo Extensible de Provisión, *EPP*) es un protocolo para el registro centralizado de objetos. Su principal característica es la extensibilidad, la cual lo hace apto para el registro de una gran variedad de objetos. Además, hace posible modificar el mapeo del protocolo a diversos objetos para adaptarlo a las necesidades de su usuario.

La primer intención del protocolo fue el registro centralizado de nombres de dominio de Internet, con la información asociada de servidores de nombres y personas que responden por dichos dominios, para su posterior publicación en el Servicio de Nombres de Dominio (*Domain Name Service*, *DNS*). Esto se logró a través del mapeo del protocolo a los objetos *domain* (que representan un nombre de dominio), *host* (los cuales representan a los servidores de nombre que responden por los dominios), y *contact* (utilizados para almacenar la información relativa a las personas).

En Uruguay, la tarea de registro de nombres de dominio, es llevada a cabo por el Servicio Central de Informática Universitaria (SeCIU), el cual tiene delegado el registro de los dominios *.com.uy* a la estatal Administración Nacional de Telecomunicaciones (ANTEL).

Actualmente, estos registros son disjuntos, pero es interés del servicio, el centralizarlos y abrir la posibilidad de más registradores para los diversos sub-dominios. Para esto, SeCIU, plantea el uso de *EPP* como protocolo de comunicación entre los *registrars* y el *registry* central. Por tal motivo, se plantea este proyecto, el cual busca reducir la brecha existente entre los requerimientos de SeCIU respecto al registro de dominios, y la utilización de *EPP* para cubrirlos.

Palabras clave *DNS*, dominio, zona, *EPP*, *registry*, *registrar*, *registrant*.

Tabla de contenidos

1. Introducción **1**
- 1.1. Presentación . . . 2
  - 1.1.1. Contexto de trabajo . . . 2
  - 1.1.2. Objetivos . . . 2
- 1.2. Marco de trabajo . . . 4
  - 1.2.1. Sistema de Nombres de Dominio . . . 4
  - 1.2.2. Sistemas de Registro de Nombres de Dominio . . . 4
  - 1.2.3. Protocolo Extensible de Provisión . . . 5
  - 1.2.4. Extensiones a *EPP* . . . 5
  - 1.2.5. Software de Repositorio . . . 6
- 1.3. Desarrollo . . . 7
  - 1.3.1. Definición de Extensiones para el Registro de Dominios .UY . . . 7
  - 1.3.2. Repositorio Extendido de Dominios . . . 7
  - 1.3.3. Modelo de pruebas . . . 7
- 1.4. Final . . . 8

I Marco de trabajo **9**

2. DNS **10**
- 2.1. Introducción . . . 11
- 2.2. El Sistema de Nombres de Dominio . . . 12
  - 2.2.1. Características claves del diseño del *DNS* . . . 12
- 2.3. Extensiones de seguridad para *DNS* . . . 18
  - 2.3.1. Beneficios . . . 19
  - 2.3.2. Modificaciones a *DNS* . . . 20
  - 2.3.3. Proceso de Validación . . . 25
  - 2.3.4. Firmado de una Zona . . . 26
  - 2.3.5. Gestión de Claves . . . 28
  - 2.3.6. Limitaciones de *DNSSEC* . . . 29

2.4. El Servidor de Nombres *BIND* 30

3. Sistemas de Registro **32**
3.1. Introducción 33
3.2. Registro de dominios .UY 33
3.2.1. Registro de Nombres de Dominio en SeCIU 34
3.2.2. Sistema Integral de Gestión de Dominios 38
3.2.3. Registro de Nombres de Dominio en ANTEL 40
3.2.4. Validación de nombres de dominio 41
3.3. Registro de dominios .CL 42
3.4. Registro de dominios .PL 44
3.4.1. Reglas para el registro de Nombres 44
3.4.2. Utilización de *EPP* 45
3.5. Registro de dominios .BR 46
3.5.1. Dominios de segundo nivel 47
3.5.2. Soporte para *DNSSEC* 48

4. EPP **49**
4.1. Introducción 50
4.2. El protocolo *EPP* 50
4.2.1. Comandos del *EPP* 52
4.2.2. Identificación de objetos 53
4.2.3. Mensajes básicos 55
4.2.4. Códigos de resultado 56
4.2.5. Consideraciones de seguridad 56
4.3. Extendiendo el *EPP* 57
4.3.1. Marco de trabajo para las extensiones 58
4.3.2. Guías para extender el protocolo 59
4.4. Utilizando *EPP* 60
4.4.1. Contactos 62
4.4.2. *Hosts* 62
4.4.3. Nombres de dominio de Internet 63
4.5. Resumen del capítulo 64

5. Extensiones a EPP **65**
5.1. Introducción 66
5.2. Extensiones para el dominio .COOP 66
5.3. Extensiones para *IDN* 68
5.4. Extensiones para *DNSSEC* 69
5.5. Extensiones para el dominio .PL 75
5.6. Extensiones para el dominio .BR 78

5.7. Período de gracia 80
5.8. Resumen del capítulo 82

6. Software de Repositorio 84
6.1. Introducción 85
6.2. *OpenReg* 85
6.3. *CoCCA* 88
6.4. *FRED* 91
6.5. Otros sistemas 93
6.6. Librerías *EPP* 94
6.6.1. *Universal Registry/Registrar Toolkit* 94
6.6.2. *Perl EPP Library (Net::EPP)* 97
6.6.3. *PHP EPP Library (Net_EPP_Client)* 97
6.6.4. *NeuLevel Registrar Toolkit* 98
6.6.5. *Net::DRI* 99
6.6.6. *EppClient API* 100
6.7. Clientes *EPP* genéricos 100
6.8. Clientes *EPP* específicos 101
6.9. Extendiendo *OpenReg* 101
6.9.1. Modificación del *XML* de entrada de un comando *EPP* 102
6.9.2. Modificación del *XML* de salida de un comando *EPP* . 104
6.9.3. Creación de un nuevo comando *EPP* 104
6.9.4. Modificación de generación de archivo de zona 104
6.9.5. Corrección de problemas de portabilidad 105
6.10. Conclusiones 106

II Desarrollo 107

7. Extensiones para .UY 108
7.1. Introducción 109
7.2. Manejo de Dominios Críticos 110
7.2.1. Problema 110
7.2.2. Solución 110
7.2.3. Implementación 111
7.3. Registro de nombres *IDN* 111
7.3.1. Problema 111
7.3.2. Solución 112
7.3.3. Implementación 115
7.4. Extensiones *DNSSEC* 119
7.4.1. Solución 119

7.4.2. Implementación 119
7.5. Validación de acciones 120
7.5.1. Problema 120
7.5.2. Solución 121
7.5.3. Implementación 128
7.6. Listados de objetos 131
7.6.1. Problema 131
7.6.2. Solución 132
7.7. Período de Gracia 137
7.7.1. Problema 137
7.7.2. Solución 139
7.7.3. Implementación 144
7.8. Auditoría de acciones 145
7.8.1. Problema 145
7.8.2. Solución 145
7.8.3. Implementación 146

8. Repositorio Extendido de Dominios 148
8.1. Introducción 149
8.2. Modelo de Casos de Uso 150
8.2.1. Actores 150
8.2.2. Casos de uso 150
8.3. Casos de uso críticos 155
8.3.1. Registro de dominios con conflictos *IDN* 155
8.3.2. Registrar un dominio que necesita autorización previa . 157
8.3.3. Tipos de datos 159
8.4. Modelo de Arquitectura 160
8.4.1. Vista lógica y de componentes 160
8.4.2. Vista de subsistemas 162
8.4.3. Vista de implementación y despliegue 164
8.5. Modelo de implementación 166
8.5.1. Tecnologías empleadas 166
8.5.2. Implementación de extensiones 169
8.5.3. Implementación de un Agente Legislador 171
8.5.4. Implementación de un Agente Ejecutor 172

9. Modelo de pruebas 174
9.1. Introducción 175
9.2. Entorno de pruebas 176
9.3. Casos de prueba 176
9.3.1. Creación de dominio especial 176

9.3.2. Creación de dominio *IDN* . . . 177
9.3.3. *DNSSEC* . . . 177
9.3.4. Reportes . . . 177
9.3.5. Dominios Críticos . . . 178
9.3.6. Logs . . . 178
9.3.7. Pruebas de Performance . . . 178
9.4. Conclusiones . . . 178

III Final 180

10. Conclusiones 181
10.1. Conclusiones . . . 182
10.1.1. Sistemas de repositorio . . . 182
10.1.2. *DNSSEC* . . . 183
10.1.3. Repositorio Extendido de Dominios . . . 184
10.2. Conocimientos Adquiridos . . . 185
10.3. Evaluación del trabajo . . . 186
10.4. Trabajo a futuro . . . 188
10.4.1. Tareas fuera del alcance . . . 188
10.4.2. Problemas conocidos . . . 189
10.4.3. Oportunidades de mejora . . . 190

Bibliografía 192

IV Apéndices 197

A. Ejecución del Modelo de Pruebas 198

B. Instalación y configuración *D-REX* 241

C. Manual de uso de las aplicaciones 252

Glosario 263

Lista de Figuras 270

Lista de Cuadros 272

Capítulo 1

Introducción

1.1. Presentación

Desde el año 1990, el dominio de primer nivel *.UY*, es administrado por el Servicio Central de Informática Universitaria (SeCIU) dependiente de la Universidad de la República (UdelaR). En 1994, la zona *.com.uy* le fue delegada a la Administración Nacional de Telecomunicaciones (ANTEL), quien a partir de 2000, la administra a través de ANTELDATA, su servicio de datos. De esta manera, SeCIU queda encargado de la administración del resto de los dominios bajo *.uy*: *.edu.uy, .org.uy, .net.uy, .gub.uy* y *.mil.uy*.

Actualmente, SeCIU, se encuentra implementando un sistema de registro de nombres de dominio basado en el *Extensible Provisioning Protocol* (*EPP*) con la intención de centralizar dicho registro en una única base, y abrir la posibilidad de que existan registradores externos, como pueden ser las diversas empresas que actualmente se dedican a vender servicios de administración de dominios (registro, alojamiento, etc.) [1].

EPP está especificado en *XML*, y define objetos y operaciones sobre los mismos. Además se especifica un mecanismo para definir extensiones al protocolo para la creación de nuevos objetos y nuevas operaciones.

El proyecto presentado en este documento, consiste en estudiar el protocolo *EPP* y sus mecanismos de extensión, y aplicarlos al Sistema de Registro de Dominios UY, evaluando sus necesidades al respecto.

1.1.1. Contexto de trabajo

El servidor *EPP* de SeCIU está implantado en base a un desarrollo de software libre llamado *OpenReg* (ver Sección 6.2). El mismo implementa los objetos y comandos *EPP* estándar definidos en las *RFC* correspondientes para el registro de nombres de dominio ([2], [3], [4]).

Debido a que cada país puede tener particularidades en sus políticas de registro de dominios, puede suceder que ciertos procedimientos o datos requeridos no estén debidamente contemplados en el *EPP* estándar. Por lo tanto, se debe estudiar profundamente el protocolo y sus mecanismos de extensión, así como las extensiones que pudieran existir, y evaluar las necesidades del registro en Uruguay, para definir nuevas extensiones que permitan adaptar el uso de *EPP* a las particularidades existentes en dicho registro.

1.1.2. Objetivos

La propuesta de trabajo, comprende el dominar los sistemas de base subyacentes (*DNS* y *EPP*), así como la comprensión del funcionamiento de otros sistemas de registro de nombres de dominio diferentes al uruguayo, el estudio

de herramientas que implementen *EPP*, y la interiorización con los mecanismos de extensión del protocolo. Asimismo, se deberán estudiar extensiones existentes para evaluarlas contra las necesidades de SeCIU, y para utilizarlas como insumos en la definición e implementación de nuevas funcionalidades específicas de la realidad uruguaya.

Se divide la propuesta en los siguientes ítem:

1. Interiorización en el funcionamiento de los sistemas de registro de dominios.
2. Estudio del *DNS* y su administración.
3. Estudio de las extensiones de seguridad del *DNS* (*DNSSEC*).
4. Relevamiento del Sistema de Registro de Dominios UY.
5. Relevamiento de otros sistemas de registros.
6. Estudio del protocolo *EPP* y los mecanismos de extensión.
7. Relevamiento y análisis de las herramientas de software existentes relativas al *EPP*, y de extensiones ya definidas.
8. Estudio de los aspectos del *EPP* relativos al *DNSSEC* y análisis del impacto de su implantación (cambios al diseño de la bases de datos, procedimientos, etc.).
9. Estudio del actual sistema del SeCIU que implementa el *EPP*.
10. Análisis y definición de nuevos objetos y procedimientos, así como posibles mejoras al actual sistema.
11. Definición de nuevas extensiones al protocolo *EPP*.
12. Programación de las extensiones definidas.

Para abarcar dichos puntos, se dividió el trabajo en tres grandes etapas claramente diferenciadas: Marco de trabajo, Desarrollo y Final, las cuales se presentan en la próximas secciones.

1.2. Marco de trabajo

La primer etapa del trabajo, consiste en entender el contexto en el cual se va a trabajar, es decir, analizar el marco de trabajo en el cual está inscrito el proyecto. Esto abarca los puntos 1 al 8 de la lista anterior, y, claramente, es la etapa más importante en cuanto a tiempo y relevancia del trabajo aquí presentado. A continuación, se introducen los temas tratados en esta parte.

1.2.1. Sistema de Nombres de Dominio

El *Sistema de Nombres de Dominio* de Internet *(DNS)*, es un protocolo de capa de aplicación que resuelve direcciones de máquinas a partir de sus nombres ([5], [6] y diversas actualizaciones). La comprensión de este sistema, es clave en una investigación que aborda un sistema que provee la capacidad de registro de estos nombres, como lo es *EPP*.

A (muy) grandes rasgos, *DNS* es una base de datos ampliamente distribuida, que almacena el espacio de nombres de dominio, y los recursos asociados a éstos, en una estructura arbórea dividida en zonas, asociadas a los diferentes sub-árboles. Los interesados en este sistema, o, dicho de otra forma, los actores involucrados, son los *resolvers* y los *nameservers*, o *servidores de nombres*. Los primeros, consultan a los segundos, para averiguar a qué dirección *IP* corresponde un determinado nombre de dominio o cuáles son los recursos de un determinado dominio.

Este sistema, no es inmune a los ataques maliciosos, que buscan, por ejemplo, que los servidores respondan a las consultas con direcciones o datos falsos. Por este motivo, *DNS*, se extiende con las Extensiones de Seguridad para *DNS*, *DNSSEC*, que introduce el firmado de las zonas, buscando garantizar autenticidad e integridad de los datos obtenidos en una consulta.

Todos estos puntos, son tratados con mayor profundidad en el Capítulo 2.

1.2.2. Sistemas de Registro de Nombres de Dominio

La administración del *ccTLD* (*Country Code Top Level Domain*) .UY, correspondiente a la República Oriental del Uruguay, es realizada por la Universidad de la República, a través de SeCIU, estando delegada la administración de la zona *.com.uy* a ANTEL. Resulta de interés el analizar las características de cada sistema de registros, para evaluar en qué manera puede ayudar la extensión del protocolo *EPP* (presentado en el Capítulo 4).

El Sistema Integral de Gestión de Dominios (*SIGD*) es la herramienta desarrollada por SeCIU para el Sistema de Registro de Nombres de Dominio

.UY. Es una aplicación Web basada en páginas dinámicas que cumple el papel de *registrar* ante un servidor *EPP* que actúa como *registry*, y que permite que sus usuarios registren y administren sus nombres de dominios. En la Sección 3.2.2, se presentan los detalles más relevantes de esta aplicación, como lo son su arquitectura, aspectos de persistencia, e interacción con el *registry*.

De la misma manera, resulta interesante realizar una investigación sobre los sistemas implementados por otros *ccTLD* o *gTLD* (*Generic Top Level Domain*), analizando particularidades en el proceso, prestando especial atención al uso que éstos dieren a *EPP*, para ayudar a la definición de nuevos aspectos en el dominio .UY, así como brindar la posibilidad de reutilización de elementos existentes.

El Capítulo 3, presenta los aspectos más importantes del registro de nombres de dominio .UY por parte de las instituciones actuales, y se realiza un análisis de las proyecciones a futuro sobre este escenario. Asimismo, se realiza un análisis similar para el registro de nombres en los dominios .CL (Chile), .PL (Polonia) y .BR (Brasil). La elección de estos *ccTLD* surge de particularidades existentes en sus procesos, así como de la utilización que hacen de determinadas tecnologías.

1.2.3. Protocolo Extensible de Provisión

EPP es un protocolo que permite la provisión de diversos objetos de registros (*registries*), que son quienes almacenan centralmente los objetos, hacia registradores (*registrars*) que interactúan con los registrantes (*registrants*), interesados en registrar objetos.

Las operaciones que los *registrars* pueden realizar en los repositorios, son las típicas de una colección. Se pueden crear nuevos objetos que se almacenen en el repositorio, modificar los que ya se tienen registrados y renovar su validez, borrarlos y transferirlos.

Si bien el protocolo está pensado para diversos dominios de aplicación, su primer fin, y uso principal, es el de registro de nombres de dominio en Internet. Asimismo, este proyecto se enmarca en una institución (SeCIU) que se dedica, entre otras tareas, al registro de nombres bajo el dominio .UY. Es por estos motivos, que el foco de la investigación presentada en el Capítulo 4, es *EPP* aplicado a nombres de dominio.

1.2.4. Extensiones a *EPP*

A pesar de que la primera intención de *EPP* es el registro de nombres de dominios de Internet, y sus contactos y *nameservers* asociados, está planteado de una manera genérica (para “cualquier” tipo de objeto) y extensible.

Los mecanismos de extensión definidos, son los que permiten adaptar un protocolo genérico, por ejemplo, para su uso con los nombres de dominio [7].

Aparte de las extensiones para los mapeos referentes a los nombres de dominio, que se analizan en el Capítulo 4, existen otro tipo de extensiones que diversos interesados han ido definiendo para adaptar el protocolo a sus necesidades. En el Capítulo 5, se presentan, de todo el universo de extensiones, algunas que resultan de interés para el proyecto, y así poder tener una visión del trabajo ya realizado en el tema, y tener presentes algunos elementos manejados por otros agentes, a la hora de definir las extensiones necesarias para el Sistema Registro de Nombres de Dominio .UY.

1.2.5. Software de Repositorio

Existen diversas soluciones de software que implementan Sistemas de Registro Centralizado. Algunas, como *OpenReg*, el sistema utilizado por SeCIU, implementan *EPP* estándar. Pero ésta no es la única, ya que existen otras soluciones que implementan el servicio, incluso, realizando extensiones propias al protocolo básico. Por ejemplo, el sistema *FRED*, realiza un manejo diferente de los *nameservers* asociados a cada registro. Además, existen otras alternativas de repositorios, que no implementan *EPP*, o lo implementan de manera incompleta. En el Capítulo 6, se realiza un relevamiento de tales sistemas de registro centralizado.

Las características fundamentales tomadas en cuenta a la hora de este análisis, son las interfaces ofrecidas, el modelo de datos manejado, y el tipo de licenciamiento ofrecido. De esta manera, se analizan en profundidad tres sistemas que implementan *EPP*: los ya mencionados en el párrafo anterior, y *CoCCA*. Luego, se hace mención a otras alternativas que no utilizan *EPP* o que lo implementan de manera parcial.

En la Sección 6.6, se presentan varias librerías que operan del lado del cliente, permitiendo conectarse a servidores *EPP* para ejecutar comandos, buscando, con dicho análisis, ofrecer los puntos de vista que deberían ser tomados en cuenta para elegir una.

También existen diversos aplicativos que permiten conectarse a servidores *EPP* para realizar diversas tareas sin necesidad de programar. Estos clientes, pueden ser genéricos, o estar desarrollados para un *registry* específico, y se analizan en las secciones 6.7 y 6.8, respectivamente.

Además, debido a que es el sistema utilizado en SeCIU, se presta especial atención a *OpenReg*, analizando, en la Sección 6.9, los aspectos relativos a las modificaciones que hay que realizar en su código para implementar extensiones *EPP*.

1.3. Desarrollo

1.3.1. Definición de Extensiones para el Registro de Dominios .UY

En el Capítulo 3, se analizan las características más importantes del registro de dominios .UY. Éste, especialmente el realizado por SeCIU, posee ciertas reglas que no son contempladas por la versión estándar de *EPP*. Por ejemplo, en SeCIU, está definido que los dominios gubernamentales (*.gub.uy*) deben cumplir con cierto proceso de validación, asunto que no está contemplado en el protocolo.

Como se analiza en el Capítulo 5, otros *registries* han enfrentado este tipo de problemas y adaptado *EPP* a sus necesidades a través de extensiones. En el Capítulo 7, se presenta un trabajo de extensión que, tomando como insumos estas extensiones y los requerimientos particulares de SeCIU, y utilizando los mecanismos de extensión de *EPP*, adapta el protocolo a las características del negocio en Uruguay.

1.3.2. Repositorio Extendido de Dominios

En el Capítulo 8, se presenta un desarrollo realizado sobre la base del sistema *EPP* empleado en SeCIU, *OpenReg*, que agrega extensiones y gestión de los procesos asociados a éstas.

Para realizar dicho desarrollo, se empleó una metodología ágil, basada en la generación de código, y centrada en las extensiones a cubrir. Se tomó como punto de partida la extensión más sencilla, que permitió un acercamiento a la plataforma involucrada. Luego, utilizando como punto de partida las interacciones requeridas por las extensiones definidas, se realizaron los casos de uso del sistema. Mediante la identificación de los casos críticos del sistema, se definió una arquitectura, sobre la cual se trabajó hasta obtener el sistema.

El resultado de este desarrollo es el prototipo de un sistema que permite el registro de dominios y la ejecución de las acciones asociadas, denominado Repositorio Extendido de Dominios, *D-REX*. Este prototipo se divide en dos partes fundamentales: el repositorio *OpenReg* y la aplicación Web de Administración del Repositorio Central de Openreg, *ARCO*.

1.3.3. Modelo de pruebas

Para verificar la correctitud de cualquier sistema de *software*, es necesario probarlo. Las pruebas pueden ser sobre las funcionalidades, sobre el desempeño, y sobre muchos otros aspectos. Además, la realización de éstas, puede

tener diversos objetivos.

En el caso de *D-REX*, apuntan a determinar la factibilidad de un prototipo desde el punto de vista funcional, para validar los procesos implementados.

Los objetivos del modelo de pruebas, el entorno de pruebas, y los casos de prueba, son analizados con detalle en el Capítulo 9.

1.4. Final

El trabajo realizado a lo largo del proyecto, permitió un conocimiento profundo del protocolo *EPP*, del protocolo *DNS*, y de diversas extensiones realizadas a ambos, entre las que se destaca *DNSSEC*. Además, se realizó un inventario de herramientas para servidores y clientes *EPP*.

Asimismo, del análisis detallado de varios sistemas de registro de dominios, entre los que se encuentra el de Uruguay, surgió un conjunto de extensiones que permiten extender *EPP*, adaptándolo a varias necesidades de este registro. Varias de las extensiones definidas, pudieron ser implementadas y probadas, demostrando así, la factibilidad del marco de trabajo propuesto, y proporcionando un buen conjunto de guías para extender las posibilidades, tanto de los protocolos de base, como del sistema de registro de nombres.

Los anteriores elementos, son analizados en el Capítulo 10. En el mismo capítulo, se detallan los aspectos que están fuera del alcance del proyecto.

Parte I

Marco de trabajo

Capítulo 2

Sistema de Nombres de Dominio

2.1. Introducción

En la década de 1960, el Departamento de Defensa de los Estados Unidos creó una red experimental de computadoras, llamada *Arpanet*, que interconectaba importantes centros de investigación en ese país. En sus comienzos, gracias a que la cantidad de *hosts* era reducida, el mapeo entre nombres y direcciones de *host* era mantenido mediante un archivo *HOSTS.TXT* que era actualizado y distribuido periódicamente.

Con el correr de los años *Arpanet* sufrió un crecimiento explosivo, y este aumento en la cantidad de *hosts* de la red hizo que el esquema utilizado se volviera inmanejable.

Los principales problemas de la utilización de un archivo de nombres son:

1. La distribución del archivo *HOSTS.TXT* se había vuelto muy costosa en términos de tráfico y de carga de los servidores.

2. El mecanismo no tenía como defenderse frente a las colisiones de nombres de *host*, provocando que si se agregaban dos nodos con el mismo nombre existía la posibilidad de generar problemas de conectividad de parte de la red.

3. La consistencia del archivo era un problema, ya que en la práctica el tiempo entre que surgía una modificación o un nuevo *host* era menor al tiempo que tomaba que todos los nodos actualizaran el archivo.

Por estos motivos, se decidió llevar a cabo una investigación para diseñar el sucesor del archivo *HOSTS.TXT*. Éste surgió en el año 1984, cuando fueron publicadas las *RFC* 882 y 883 que describen al Sistema de Nombres de Dominio, conocido como *DNS* por su sigla en inglés [5][6].

Para evitar los problemas de distribución del archivo *HOSTS.TXT*, se diseñó *DNS* de modo que la descentralización fuese un factor clave. Al descentralizar la administración de los nombres de *host*, se logró simplificar la tarea de mantener los datos actualizados. Además, el problema del tráfico se subsanaba, al eliminar la restricción que obligaba a que la información estuviera en un punto, permitiendo que la carga se repartiera entre diferentes servidores.

Otro factor clave es la jerarquización. Gracias a ella, desaparecen los problemas de colisión de nombres y se ayuda a delimitar la delegación de los diferentes segmentos dentro del espacio de los *hosts* de la red.

DNS no está exento de problemas: en los '90, Steven M. Bellovin descubre vulnerabilidades en la seguridad del protocolo. En 1999, surge las Extensiones de seguridad para *DNS* (*DNSSEC*), el primer intento de solución a estos

problemas. En años posteriores, *DNSSEC* evolucionó para poder ser implantado en toda la red, y poder convertir a *DNS* en un protocolo más seguro y menos vulnerable.

En lo que sigue de este Capítulo, se describen las principales características del servicio *DNS*, se analizan sus extensiones de seguridad, y se presenta brevemente una de las implementaciones más populares, *BIND*.

2.2. El Sistema de Nombres de Dominio

2.2.1. Características claves del diseño del *DNS*

Nombres de dominio

Los nombres de dominio se organizan bajo una estructura de árbol invertido que se muestra en la Figura 2.1. Al nodo raíz se lo denomina con ' . '. A los dominios correspondientes al primer nivel de hijos, se los denomina *TLD* (*Top Level Domains*, dominios de primer nivel), y está compuesto por los *gTLD* (*TLDs* genéricos, com por ejemplo *.edu*), los *ccTLD* (*Country Code TLD*, *TLD* de código de país), y el dominio especial *.arpa*, que se utiliza para aspectos de la infraestructura de Internet, por ejemplo el *.in-addr.arpa*, para el mapeo inverso de direcciones a nombres de dominio. La administración de estos dominios se delega a diferentes organizaciones, por ejemplo, SeCIU en Uruguay.

Cada uno de estos dominios se divide en diferentes sub-dominios, por ejemplo *com.uy* y *edu.uy*, pero esta subdivisión no es igual para todos los *TLD*. Por citar un ejemplo, Gran Bretaña utiliza *co.uk* y no *com.uk*. A su vez, la estructura de sub-dominios se sigue repitiendo dentro de cada dominio.

Los *host* tienen nombres de domino apuntando a ellos. Por ejemplo *margarita.fing.edu.uy*, donde *margarita*, es el nombre del *host*, el cual pertenece al sub-dominio *fing*, el cual a su vez pertenece al sub-dominio *edu* del *ccTLD uy*. Además del nombre, es posible que los *host* tengan alias apuntando a ellos.

Esta estructura, si bien es mucho más compleja que el archivo *HOSTS*, es lo que garantiza que no exista colisión de nombres. Siguiendo con el ejemplo, el sub-dominio *iie.fing.edu.uy* puede elegir el nombre *margarita* sin que exista colisión con otro *host* con igual nombre, pero de diferente sub-dominio.

Administración de zonas

Cada dominio consiste en un número de sub-dominios, y la administración de cada uno de ellos puede delegarse o no a otra organización. Una *zona*

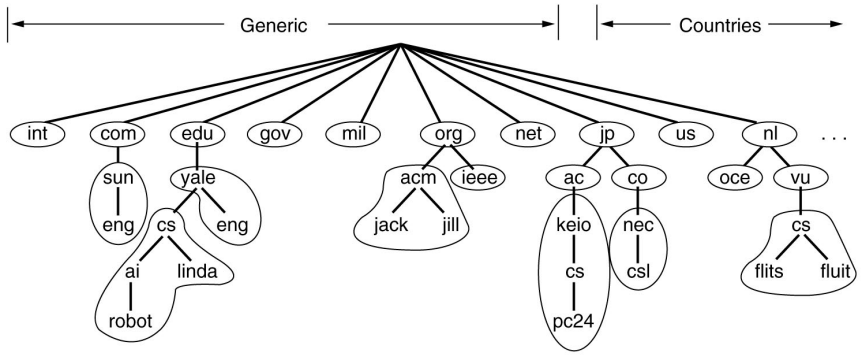


Figura 2.1: Árbol de *DNS*

es una parte del espacio de nombres de dominio, administrada de manera autónoma, y surge de la delegación de un sub-dominio, correspondiente a una rama del árbol de la Figura 2.1. Las organizaciones, por lo tanto, administran zonas, las cuales están compuestas por sub-dominios. De la misma manera, los servidores de nombres de dominio, toman como unidad de administración la zona y no el dominio. La razón para que esto sea así, es que un dominio puede contener sub-dominios delegados, los cuales no se deben administrar.

Actores de la resolución de nombres

Los dos actores principales en la resolución de nombres, son los *resolvers* y los *nameservers*. Los *resolvers* son los clientes que utilizan el sistema de nombres de dominio, y son librerías cuya función es resolver nombres de dominio a los diferentes programas que lo requieran, como por ejemplo un navegador o un cliente *FTP*. Las acciones que realizan son consultar a los *nameservers*, interpretar la respuesta y retornar la información al programa que la solicitó.

Los *nameservers* son los programas que contienen la información de las diferentes zonas y responden a los *resolvers*. Existen varias maneras de clasificar *nameservers*. Éstos pueden ser autoritativos o no autoritativos, recursivos o iterativos, y primarios o secundarios. Un *nameserver* es autoritativo cuando está respondiendo por un dominio que depende de él. Los servidores iterativos, en caso de no poder contestar autoritativamente a una consulta, responden con una referencia a otro *nameserver*, mientras que los recursivos son capaces de consultar otros *nameservers* para encontrar la información buscada. Finalmente, los primarios cargan la información de la zona desde archivos y los secundarios obtienen la información desde otros *nameservers*,

utilizando transferencias de zonas. Como un *nameserver* podría ser primario para una zona, y secundario para otra, se lo llama primario (o secundario) cuando es primario (o secundario) para la mayor parte de las zonas en las que es autoritativo.

En la práctica, los *nameservers* se configuran como recursivos cuando forman un servicio de conectividad, y como autoritativos cuando se tienen dominios propios.

Consultas

De manera similar a lo que ocurre con los *nameservers*, las consultas pueden clasificarse en dos tipos: *recursivas* o *iterativas*. En el caso de las consultas recursivas, el *nameserver* consultado debe, o bien responder con el dato solicitado, o bien con un error, por ejemplo que el dominio buscado no existe. El *nameserver* no puede simplemente responder con una referencia a otro *nameserver*. Por lo tanto, si él no tiene la respuesta a la consulta solicitada, deberá consultar a otros *nameservers* hasta obtener la respuesta y devolverla al *resolver* solicitante.

En las consultas iterativas, el *nameserver* nunca envía nuevas consultas a otros *nameservers*, sino que contesta con los datos que ya posee. Si tiene el dato localmente, lo responde, y si no, a partir de los datos de su zona, retorna una referencia a otro *nameserver* más cercano al buscado, tomando en cuenta la posición de éstos en la estructura jerárquica.

Resolución de nombres

Debido a las limitaciones de los *resolvers*, los *nameservers* no sólo pueden resolver nombres correspondientes a las zonas para las que son autoritativos, sino también respecto a zonas donde no lo son. Debido a la estructura de árbol invertido de los nombres de dominio, alcanza con conocer la raíz del árbol para que, recorriendo las ramas, se pueda llegar a cualquiera de las hojas.

Los *nameservers* raíz se denominan *root nameservers* y su funcionamiento es crucial para que toda Internet siga en pie. Por esto se encuentran distribuidos, siendo actualmente 13 los *root nameservers*, nombrados desde la A hasta la M.

El proceso de resolución de un nombre de dominio, por ejemplo *lulu.fing.edu.uy*, comienza cuando un *resolver* consulta a su *nameserver* local por el nombre *lulu.fing.edu.uy*. Si esta solicitud parte desde fuera de la red de *fing*, el *nameserver* local no tendrá información autoritativa de la zona *fing.edu.uy*. Las consultas de los *resolvers* son normalmente recursivas, por lo que el *na-*

meserver deberá hacer el trabajo completo de obtener el dato buscado. Si ya tuviese ese dominio (o parte de él) en su *caché*, podría ahorrarse el realizar la consulta, pero, en el supuesto de que no sea así, el *nameserver* deberá consultar a uno de los *root nameservers* por el *host lulu.fing.edu.uy*.

Las consultas que realiza el *nameserver* local para encontrar el dato buscado, normalmente son iterativas. Por lo tanto, el *nameserver* raíz contestará con una referencia al *nameserver* de la zona *uy*. El proceso se repite tres veces más, primero consultando al *nameserver* de la zona *uy* (obteniendo una referencia al *nameserver* de la zona *edu.uy*), luego consultando al *nameserver* de la zona *edu.uy* y finalmente consultando al *nameserver* de la zona *fing.edu.uy*, el cual es autoritativo y responde al *nameserver* local con el dato buscado, el cual a su vez se lo responde al *resolver* que inició la consulta.

Registros

Los servidores de nombres de dominio no se utilizan únicamente para el mapeo de nombres de dominio a direcciones, sino que se pueden consultar varios tipos de registros de recursos. Éstos, llamados también *resource records*, tienen los siguientes campos:

1. Nombre
2. Tipo
3. Clase
4. *TTL (Time To Live)*
5. Largo de *RData*
6. *RData* (datos específicos del *resource record*)

En las siguientes secciones se detallan los tipos de registros más usuales.

Registros de tipo *A*

El tipo de registro *A* es utilizado para mapear nombres de dominio a direcciones *IP* versión 4. El campo nombre del registro contiene el nombre de dominio, mientras que el campo *RData* contiene la dirección *IP*. Un ejemplo de este tipo de registro puede verse en el Cuadro 2.1.

En la resolución de nombres, éste es el tipo normalmente buscado como resultado final por los *resolvers*, porque una vez conocida la dirección *IP* correspondiente al nombre de dominio consultado, es posible establecer la comunicación de manera directa.

lulu.fing.edu.uy.	86400	IN	A	164.73.44.3

Cuadro 2.1: Ejemplo de un registro *A*

Registros de tipo *AAAA*

De manera análoga a los registros *A*, los registros *AAAA* se utilizan para mapear nombres de dominio a direcciones *IP*, pero en este caso se trata de direcciones *IP* versión 6. La razón para que se nombre con 4 letras A, es que las direcciones *IPv6* son 4 veces más largas que las direcciones *IPv4* (128 *bits* contra 32 *bits*). En el Cuadro 2.2 se puede observar un ejemplo de este tipo de registro.

seciu.edu.uy.	85423	IN	AAAA	2001:1328:6::5

Cuadro 2.2: Ejemplo de un registro *AAAA*

Registros de tipo *CNAME*

DNS permite especificar más de un nombre para un mismo *host*. Se denomina nombre canónico al nombre oficial, o sea, el nombre para el cual existe un registro de tipo *A* que lo define. Los otros nombres del *host* serán alias que apuntan al nombre canónico. Para estos últimos, deberá existir registros de tipo *CNAME* con cada uno de sus nombres, todos indicando en el campo *RData* el nombre de dominio apuntado. Ver Cuadro 2.3 por un ejemplo de esto.

www.fing.edu.uy.	85061	IN	CNAME	margarita.fing.edu.uy.

Cuadro 2.3: Ejemplo de un registro *CNAME*

Las razones para utilizar registros *CNAME* son varias. La más común de ellas es cuando existen varios servicios ejecutándose en una misma dirección *IP* y se necesita diferenciarlos (por ejemplo un servidor *HTTP* que responde por varios dominios, o un servidor *FTP* y otro *HTTP* en el mismo *host*).

Durante la resolución de un nombre de dominio, al encontrarse con un registro *CNAME*, se sustituye el nombre buscado por el nombre canónico devuelto y se ejecuta nuevamente la consulta.

Registros de tipo *NS*

Los registros *NS* indican, para un dominio dado, cuales son los *nameservers* autoritativos que responden por él. Un ejemplo de este tipo de registro

puede verse en el Cuadro 2.4.

fing.edu.uy.	86400	IN	NS	ns.fing.edu.uy.

Cuadro 2.4: Ejemplo de un registro *NS*

La importancia de este tipo de registro radica en que permite conocer, para cada dominio, qué *nameserver* debe ser consultado para una zona dada.

Registros de tipo *PTR*

Otra funcionalidad ofrecida por *DNS*, es la posibilidad de, a partir de una dirección *IP*, obtener los nombres de dominio que apuntan a ella. Debido a que los registros se encuentran indexados por nombre, sería necesario hacer una búsqueda exhaustiva entre todos los dominios, lo cual sería altamente ineficiente. La solución a este problema consiste en crear una región del espacio de nombres de dominio que utilice las direcciones como índices. Dicho espacio es el dominio *in-addr.arpa*.

Para el caso de las direcciones *IPv4*, el dominio *in-addr.arpa* contiene hasta 256 sub-dominios, y esto se repite 4 veces. De esta forma, estos 4 niveles de sub-dominios se corresponden a las 4 partes de una dirección *IP* cuando es escrita en el formato xxx.xxx.xxx.xxx) lográndose una indexación por direcciones *IP*. A modo de ejemplo, un *host* con la *IP 15.16.192.152* se corresponde con el nombre de dominio *152.192.16.15.in-addr.arpa*.

Como se sigue la misma estructura jerárquica, en la que se va de lo general a lo particular, los nombres de dominio de *in-addr.arpa* quedan ordenados al revés de la notación habitual. La razón para esto es que, como para cada organización las direcciones comparten el mismo prefijo, se simplifica la delegación si los primeros dígitos de las direcciones, que son comunes a las direcciones, se toman como sub-dominios de más alto nivel que los últimos dígitos.

Un ejemplo de esto puede verse en el Cuadro 2.5.

3.44.73.164.in-addr.arpa.	86400	IN	PTR	lulu.fing.edu.uy.

Cuadro 2.5: Ejemplo de un registro *PTR*

Otros tipos de registros

Existen muchos otros tipos de registros. Por citar algunos, los registros *SOA* describen la zona en la que se encuentra el dominio (especificando servidores de nombres, contacto, y varios datos técnicos más), los registros *MX*

se utilizan para obtener la lista de servidores de correo para un nombre de dominio dado, o los registros *SRV* que son utilizados en la implementación de nuevos protocolos. En la Sección 2.3.2 se detallan aquellos tipos de registros necesarios para dar soporte a *DNSSEC*.

Caché

Para acelerar el proceso de resolución de nombres de dominio, los *nameservers* almacenan una copia de los datos que ya consultaron para ahorrar consultas. Además de guardar el dato buscado en las consultas anteriores, guardan los datos intermedios, es decir, las referencias a otros *nameservers* que tuvieron que ser consultados antes de alcanzar el nodo buscado. De esta forma, una consulta por un *host* anteriormente buscado, no necesita comenzar desde los servidores raíz, sino que reutiliza parcialmente las consultas anteriores.

Debido a que los datos no son estáticos y van cambiando, se define un tiempo de vida (*TTL, Time To Live*) que especifica cuanto tiempo se le permite al servidor almacenar el dato en *caché*. No existe un valor ideal para este parámetro, ya que un valor pequeño maximiza la coherencia de los datos, mientras que un valor alto maximiza la performance evitando consultas. Por lo tanto, se debe siempre elegir un valor que balancee estos dos aspectos.

Además del valor de *TTL* usual, se almacenan *TTL* negativos. Una vez que un servidor autoritativo respondió que un dominio no existe, los demás pueden guardar esta respuesta negativa. El *TTL* negativo, especifica por cuánto tiempo se puede seguir contestando que no existe el objeto consultado sin volver a consultar al servidor autoritativo.

2.3. Extensiones de seguridad para *DNS*: *DNSSEC*

Los comienzos de *DNSSEC* se remontan a 1990 cuando Steven Bellovin descubre vulnerabilidades en el protocolo *DNS*. Estas vulnerabilidades son recién publicadas en un artículo en el año 1995. En 1999, *DNSSEC* ve la luz en la *RFC* 2535 [8].

Esta especificación de *DNSSEC* se comienza a utilizar a comienzos de la década del 2000, pero son detectados problemas en lo que respecta a escalabilidad que hacen inviable su utilización en grandes redes, en particular, el cambio de clave de una zona implicaba un gran volumen de datos a intercambiar. Esto llevó a que en 2005, se desarrollara una segunda especificación

llamada *DNSSEC BIS* (*RFC* 4033[9], *RFC* 4034[10] y *RFC* 4035[11]). Esta nueva versión tiene como principal novedad un nuevo tipo de registro de recurso llamado *DS* (*Delegation Signer*) el cual soluciona el problema antes mencionado al agregar un nivel más de indirección en los puntos de delegación. Ese mismo año, Suecia se convierte en el primer *ccTLD* en comenzar a utilizar *DNSSEC*. Posteriormente trasciende a los medios el *Ataque Kaminsky*, una vulnerabilidad del sistema *DNS* basada en *Cache Poisoning* [12]. Si bien no pasó a mayores, obligó a parchear los servidores *DNS* y dio mayor impulso a *DNSSEC*.

A pesar de que el objetivo de *DNSSEC* es lograr un aumento en la seguridad, su especificación tiene un efecto colateral que es permitir la enumeración de una zona. Este tema provoca controversias, al posibilitar la exposición de los datos guardados por *DNS*, aspecto contrario a las buenas prácticas respecto a la privacidad de los datos. En el año 2008, se plantea una solución a este problema agregando una extensión llamada *NSEC3*, la cual utiliza *hashing* para esconder los nombres reales de los registros y de esta manera evitar exponer los datos de las zonas [13].

2.3.1. Beneficios

DNSSEC ofrece tres aspectos claves: autenticación del origen de los datos, verificación de integridad de los datos y autenticidad de no existencia. Lo que no brinda *DNSSEC* es confidencialidad de los datos, ya que esto no fue considerado como un requerimiento.

La verificación de integridad se logra gracias a la generación de firmas digitales criptográficas de la información de los diferentes registros *DNS*, los cuales a su vez también se guardan como registros *DNS* (llamados registros *RRSIG*).

La Autenticación del origen de los datos, se realiza utilizando una cadena de confianza que se forma a partir de los cortes de zonas, al comparar la clave pública de la zona hija con un *digest* de ella en la zona padre. Para que la seguridad quede garantizada, sólo se necesita que el *nameserver* que está ejecutando la consulta recursiva, tenga un punto de entrada seguro, el cual se toma como inicio en la cadena de confianza. Este punto de entrada podría ser la zona raíz, que fue firmada recientemente, o podría utilizarse un punto de entrada seguro, llamado *Trusted Anchor*, conocido de antemano (por ejemplo configurado estáticamente).

Finalmente, la autenticidad de no existencia, se logra gracias al nuevo tipo de registro *NSEC*. Cuando se consulta por un registro inexistente, lo que se retorna es un registro *NSEC* el cual indica el intervalo de no existencia dentro de la zona en orden canónico. La razón para hacer esto es que, por

performance, las respuestas no se firman *on-line*, sino que son firmadas de antemano, por lo que no existe manera de saber por cual registro se preguntará. Por lo tanto, la solución encontrada al problema es firmar los intervalos de no existencia antes de que sean publicados. De esta manera, el *nameserver* que está ejecutando la consulta tiene la confirmación de que el registro buscado cae dentro de un intervalo en el que no existe ningún registro dentro de la zona.

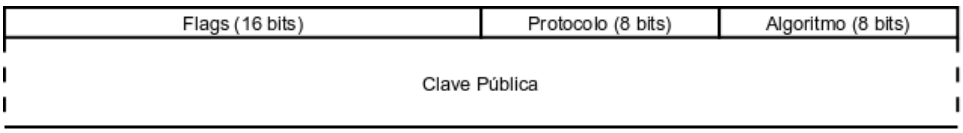
2.3.2. Modificaciones a *DNS*

Para que *DNSSEC* funcione, son necesarios algunos cambios al protocolo *DNS*. Se agregan cuatro nuevos tipos de registros de recursos: firma de registro de recurso (*RRSIG*), clave *DNS* pública (*DNSKEY*), firmante de la delegación (*DS*) y próximo seguro (*NSEC, NSEC3*).

Además, se agregan dos nuevos bits en el cabezal de los mensajes: chequeo deshabilitado (*CD*) y datos autenticados (*AD*). Para soportar los tamaños más grandes en los mensajes al agregar estos registros, se necesita también soporte para *EDNS0* [14]. Por último, *DNSSEC* requiere el agregado otro bit adicional, llamado *DO*, para que los *resolvers* puedan indicar, en sus consultas, que desean recibir registros *DNSSEC*.

Registro *DNSKEY*

El registro *DNSKEY* se emplea para guardar las claves públicas del proceso de firmado. Se utilizará esta información para validar las firmas de los demás registros de recursos.



Cuadro 2.6: Registro *DNSKEY*

Es habitual encontrar dos registros *DNSKEY* en las zonas firmadas. Uno, llamado *ZSK* (*Zone Signing Key*), con valor de *flags* 256, es el utilizado para firmar los registros, y tiene una vida útil corta. El segundo, llamado *KSK* (*Key Signing Key*) con valor de *flags* 257, tiene vida útil mayor, y con ella se firman las claves *ZSK*.

El objetivo de esta separación es brindar mayor seguridad, y a la vez disminuir el impacto de los cambios de clave, ya que con esta jerarquización se puede cambiar la clave *ZSK* sin invalidar la clave *KSK*. Por lo tanto,

Campos generales	
Nombre	Especifica el dueño (ej. uy.)
TTL	TTL del registro (ej. 21600)
Clase	Clase del registro: IN
Tipo	Tipo de registro: DNSKEY
Campos específicos	
Flags	Banderas (ej. 256 indica clave de zona)
Protocolo	Identifica el Protocolo, en este caso 3
Algoritmo	Identifica el Algoritmo utilizado (ej. 5 es *RSA/SHA1*)
Clave	Clave pública codificada como *Base64*

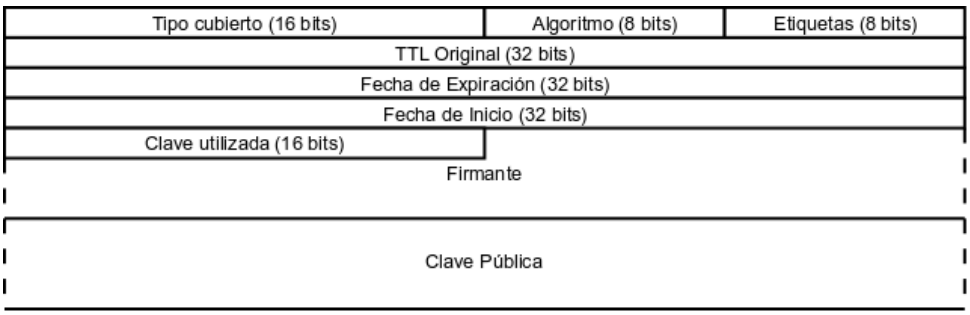
Cuadro 2.7: Detalle de los campos de un registro *DNSKEY*

aquellos registros que dependan de la clave *KSK* no se verán afectados ante un cambio de *ZSK*. Ejemplos de esto son los registros *DS* y los *Security Entry Point* (explicados más adelante en este Capítulo).

Registro RRSIG

El registro *RRSIG* guarda las firmas de los registros de recursos para un nombre, clase y tipo dado, permitiendo que sean validados. Dentro de una zona firmada, todos los registros autoritativos de la misma deben tener un registro *RRSIG* correspondiente.

Esto produce un cambio respecto a la especificación original de *DNS*: ésta dice que si un registro *CNAME* está presente para un nombre, éste es el único tipo de registro permitido para ese nombre. Con *DNSSEC*, una zona firmada tendrá además del registro *CNAME*, registros *RRSIG* y *NSEC* para el mismo nombre.



Cuadro 2.8: Registro *RRSIG*

La firma se calcula a partir de la información del registro *RRSIG*, y de los registros "cubiertos".

Campos generales	
Nombre	Especifica el dueño (ej. ultra.seciu.uy.)
TTL	TTL del registro (ej. 76314)
Clase	Clase del registro: IN
Tipo	Tipo de registro: RRSIG
Campos específicos	
Tipo cubierto	Tipo de registro sobre el que aplica la firma (ej. registro A de ultra.seciu.uy.)
Algoritmo	Identifica el Algoritmo utilizado (ej. 5 es *RSA/SHA1*)
Cant. Etiquetas	Número de etiquetas en el nombre original. El objetivo de este campo es determinar si el registro original fue o no generado a partir de comodines, ya que para poder validar la firma es necesario reconstruir el nombre utilizado. A modo de ejemplo, ultra.seciu.uy. tiene 3 etiquetas, mientras que *.seciu.uy. tiene 2
TTL Original	TTL del registro cubierto
Expiración	Fecha de expiración de la firma
Comienzo	Fecha de inicio de la validez de la firma
Etiqueta de Clave	Identifica la clave utilizada para firmar el registro
Firmante	Es el nombre de quien firma el registro (ej. seciu.uy.)
Firma	Firma codificada como *Base64*

Cuadro 2.9: Detalle de los campos de un registro *RRSIG*

$$firma = firmar\left(RRSIG\_RDATA | RR\left(1\right) | RR\left(2\right) ...\right)$$

RRSIG_RDATA es la información del registro *RRSIG* (excluyendo la firma) en un formato específico.

Para cada uno de los registros "cubiertos",

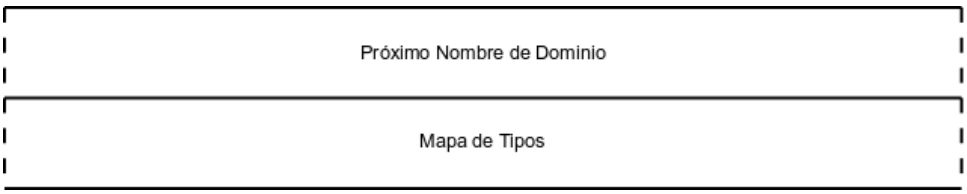
$$RR\left(i\right) = nombre | tipo | clase | TTL | largo\left(RDATA\right) | RDATA$$

Los valores de los campos algoritmo, firmante y etiqueta de la clave permiten identificar cual registro *DNSKEY* debe ser utilizado para validar la firma.

Registro *NSEC*

El registro *NSEC* muestra dos cosas. Primero, indica el siguiente registro de recurso al buscado, según el orden canónico de la zona. Segundo, retorna la lista de tipos de registro de recurso existentes para el registro solicitado. Se utiliza esta información para proveer la no existencia autenticada de datos *DNS*.

El orden canónico de una zona corresponde a un orden alfabético creciente basado en las etiquetas según su importancia (de derecha a izquierda). O sea: el primer nivel de ordenación se realiza según la última etiqueta (por ejemplo uy), luego según la penúltima etiqueta (por ejemplo com), etc.



Cuadro 2.10: Registro *NSEC*

Campos generales	
Nombre	Especifica el dueño e inicio del intervalo de no existencia (ej. a.gub.uy.)
TTL	TTL del registro (ej. 86400)
Clase	Clase del registro: IN
Tipo	Tipo de registro: NSEC
Campos específicos	
Próximo Dominio	Indica el fin del intervalo de no existencia, o sea, el próximo domino en el orden canónico (ej. b.gub.uy.)
Mapa de Tipos	Lista de tipos de registros existentes para el dueño (ej. a.gub.uy.)

Cuadro 2.11: Detalle de los campos de un registro *NSEC*

a.gub.uy.	86400	IN	NSEC	b.gub.uy	A	RRSIG	NSEC

Cuadro 2.12: Ejemplo de un registro *NSEC*

El ejemplo de la Figura 2.12 podría ser utilizado para probar dos aspectos: primero, que, por ejemplo, no existe un registro AAAA asociado a a.gub.uy.,

y segundo, que no existe un dominio algo.gub.uy., ya que según el orden canónico, de existir estaría antes que b.gub.uy.

Registro *DS*

Los registros *DS* apuntan a registros *DNSKEY* utilizados en el proceso de autenticación. Para esto guardan la etiqueta de la clave, el algoritmo, y un *digest* de la clave. Un punto muy importante es que los registros *DS* y *DNSKEY* se guardan en zonas diferentes: el registro *DS* se guarda en el lado padre de una delegación, mientras que el correspondiente registro *DNSKEY* está en la zona hija.

Durante el proceso de autenticación, el *nameserver* que está ejecutando la consulta recursiva validará que el *digest* indicado por el registro *DS* en la zona padre, se corresponda con la clave *DNSKEY* encontrada en la zona hija. Este tipo de registro no existía en la primera especificación de *DNSSEC*, pero fue agregado posteriormente en *DNSSEC BIS* ya que simplifica el proceso de cambio de claves. De todos modos, cuando una zona cambia su clave, es necesario que notifique, a su padre, el registro *DS* actualizado.

Cuadro 2.13: Registro *DS*

Campos generales	
Nombre	Especifica el dueño (ej. presidencia.gub.uy.)
TTL	TTL del registro (ej. 86400)
Clase	Clase del registro: IN
Tipo	Tipo de registro: DS
Campos específicos	
Etiqueta de Clave	Identifica la Clave *DNSKEY* a la que apunta el registro *DS* (ej. 20295)
Algoritmo	Indica el algoritmo utilizado en la construcción de la clave *DNSKEY*
Tipo de Digest	Indica el algoritmo utilizado en el cómputo del digesto (ej. 1 es *SHA-1*)
Digest	Valor del *digest*

Cuadro 2.14: Detalle de los campos de un registro *DS*

Un ejemplo de registro *DS* es el del Cuadro 2.15. Su correspondiente registro *DNSKEY* se muestra en el Cuadro 2.16.

presidencia.gub.uy.	86400	IN	DS	20295	7	1
CF416942C662990F5CA04410A5DC4F99A4EDC5C0						

Cuadro 2.15: Ejemplo de un registro *DS*

presidencia.gub.uy.	21600	IN	DNSKEY	257	3	7
AwEAAblaEaapG4inrQASY3HzwXwBaRSy5mkj7mZ30F +huI7zL8g0U7dv7ufnSEQUlsC57OHoTBza+TQIv/mgQ ed8Fy4XGCGzYiHSYVYvGO9iWG3O0voBYy/zv0z7AN frA7Z3lY51CI6m/qoZUcDlNM0yTcJgilaKwUkLBHMAp 9NJPuKVt8A7OHab00r2RDEVjiLWIIuTbz74gCXOVfA mvW07c8c=						

Cuadro 2.16: Ejemplo de un registro *DNSKEY* correspondiente al registro *DS*

2.3.3. Proceso de Validación

DNSSEC requiere que los servidores de nombres soporten la extensión *EDNS0*, que consiste en varias modificaciones a los mensajes intercambiados por *DNS*. Estos mensajes normalmente estaban limitados a 512 bytes cuando son enviados con *UDP*, pero, sin embargo, hoy la mayoría de los *hosts* conectados a Internet son capaces de aceptar paquetes de mayor tamaño. En este sentido, *DNSSEC* especifica que los nombres de dominio deben soportar tamaños de mensajes de al menos 1220 bytes, siendo deseable que soporten mensajes de 4000 bytes de extensión. La razón para este requisito es que *DNSSEC* requiere que, además de los datos que ya se envían en una consulta *DNS*, se agreguen firmas de los registros, por lo que los tamaños de los mensajes *DNS* necesitan ser agrandados.

Para el proceso de validación existen dos variantes, dependiendo de si se trata de un *nameserver* recursivo o de un *stub resolver*. En el primer caso, el proceso comienza cuando el *resolver* marca el bit de *DO* en la consulta. A modo de ejemplo, supongamos que se quiere validar el registro de tipo A devuelto para *www.nic.br*. En teoría, la validación comenzaría comprobando el registro *DS* para *br.* de la zona raíz. A continuación se verificaría la zona *.br*, validando el registro *DNSKEY* para ver si corresponde al registro *DS* anterior. Luego, se verificaría si existe una zona *nic.br.*, repitiéndose el proceso anterior de verificar el corte de la zona con su registro *DS*. Finalmente se

verificaría si la firma *RRSIG* del registro de tipo A de *www.nic.br*. Esta conexión padre-hijo en cada delegación de zona, es lo que se conoce como *cadena de confianza*. Por ejemplo, si en un punto de la validación nos encontramos que en el padre existe un registro *DS*, mientras que en el hijo los registros no están firmados, o la firma no coincide, es posible que estemos ante un ataque de tipo *man in the middle* o frente a un problema de configuración de la zona.

En el caso del *stub resolver*, éste reenvía la consulta a un servidor de nombres recursivo y utiliza el bit *AD* (Datos Autenticados) de la respuesta como indicativo para determinar si los datos retornados pudieron ser verificados. También existe la posibilidad de que el *resolver* marque la consulta con el bit de *CD* (Chequeo Deshabilitado), para que las verificaciones de las firmas las realice él mismo.

2.3.4. Firmado de una Zona

Las implementaciones del firmado de zonas, varían mucho de un *ccTLD* a otro. Entran en juego diversas consideraciones, fundamentalmente, de seguridad que llevan, por ejemplo, a utilizar *hardware* específico para el firmado, en lugar de *software*. Sin embargo, se puede delinear este proceso mediante los pasos aquí indicados. Los ejemplos se utilizan *BIND* (ver Sección 2.4) ya que es la implementación de *DNS* de referencia a nivel mundial y, en particular, es la utilizada en SeCIU.

Lo primero que se debe hacer es habilitar *DNSSEC* en los servidores autoritativos. Esto se hace modificando su archivo de configuración de *BIND* y agregando la siguiente opción:

```
options {
  dnssec-enable yes;
}
```

Es importante utilizar versiones actualizadas de *BIND*, ya que el soporte para *DNSSEC* se encuentra únicamente a partir de la versión 9.3, y en particular, para poder utilizar *NSEC3*, se necesita la versión 9.6 o posterior. Además, se requiere que las librerías *OpenSSL* se encuentren instaladas en los servidores.

El siguiente paso es habilitar, además, la validación de *DNSSEC* en los servidores recursivos. Esto es necesario únicamente en éstos servidores, ya que, como fue explicado, los servidores autoritativos no realizan validación alguna, sino que son los servidores recursivos quienes se encargan de corroborar las firmas de los registros devueltos por los servidores consultados durante

la resolución de la consulta recursiva. Para el caso de *BIND*, la configuración para los servidores recursivos es la siguiente:

```
options {
  dnssec-enable yes;
  dnssec-validation yes;
}
```

Luego es necesario generar las claves *ZSK* (utilizada para firmar la zona) y *KSK* (utilizada para firmar otras claves) para cada una de las zonas administradas. Por ejemplo, para generar la clave *ZSK* utilizando el algoritmo *RSA/SHA1* y claves de 1024 bits de largo, se utiliza el siguiente comando:

```
dnssec-keygen -a RSASHA1 -b 1024 -n ZONE <nombre_de_zona>
```

Para generar la clave *KSK* utilizando el algoritmo *RSA/SHA1* y claves de 4096 bits de largo, se utiliza el siguiente comando:

```
dnssec-keygen -a RSASHA1 -b 4096 -n ZONE -f KSK
  <nombre_de_zona>
```

En ambos casos, se generan dos archivos con extensiones *key* y *private*, correspondientes a las claves pública y privada respectivamente.

En los archivos de zona es necesario agregar las claves públicas. Esto se puede lograr de varias formas. Por ejemplo, utilizando el comando *$INCLUDE* dentro del archivo, o sino, simplemente, anexando el contenido de ambos archivos de clave púbica al archivo de zona.

Una vez completados estos pasos, se debe proceder a firmar el archivo de zona. Esto permite obtener un nuevo archivo de zona que incluye los registros *RRSIG*, *NSEC* y *DNSKEY*, que tiene la extensión *signed*. El comando a ejecutar es el siguiente:

```
dnssec-signzone [-o <nombre_de_zona>] [-N <incremento>]
  [-k <archivo_ksk] <archivo_de_zona> [<archivo_zsk>]
```

Finalmente, se debe actualizar los servidores de nombres autoritativos primarios para que apunten al nuevo archivo de zona firmado en vez del archivo sin firmar. En el caso de *BIND*, esto implica reemplazar:

```
zone "<nombre_de_zona>" {
  file "<dir>/<archivo_de_zona>";
};
```

Con:

```
zone "<nombre_de_zona>" {
 file "<dir>/<archivo_de_zona>.signed";
};
```

Sólo falta recargar la configuración de los servidores para empezar a servir registros firmados. En *BIND* esto se logra ejecutando:

```
rndc reconfig
rndc flush
```

Para que los registros firmados que se están sirviendo puedan ser validados, es necesario notificar a la zona padre, insertando en ésta un registro *DS* de tal forma que se pueda crear una cadena de confianza. Para generar este registro se debe ejecutar nuevamente el comando *dnssec-signzone* pero esta vez utilizando el argumento *-g*.

```
dnssec-signzone -g [-o <nombre_de_zona>] [-N <incremento>]
  [-k <archivo_ksk] <archivo_de_zona> [<archivo_zsk>]
```

El archivo obtenido debe suministrarse por algún medio seguro al *registry*, lo cual normalmente es realizado a través del *registrar*.

Una vez que el *registry* recibió el archivo correspondiente, debe anexarlo a la zona padre de la misma forma que fueron anexadas las claves públicas a la zona hija (comando *$INCLUDE*, etc). A partir de la próxima publicación de la zona padre, el registro *DS* de la zona hija quedará disponible, por lo que los servidores de nombres recursivos podrán validarlo con el registro *DNSKEY* correspondiente, formándose así una cadena de confianza.

2.3.5. Gestión de Claves

Normalmente, se utilizan dos claves para minimizar las interacciones necesarias en los cambios de claves. Una se denomina *KSK (Key Signing Key)* y la otra es la *ZSK (Zone Signing Key)*. La diferencia fundamental entre ambas, es que se utilizan para firmar diferentes registros: la *KSK* se utiliza para firmar únicamente las claves contenidas en una zona, mientras que la *ZSK* se utiliza para firmar los registros de recursos. Debido a esto, la *KSK* tiene un período de vigencia mayor (normalmente en el entorno de 1 o 2 años) que la *ZSK* (algunos meses). Como la *KSK* se utiliza para firmar la *ZSK*, y la *ZSK* para firmar los registros, sólo se requiere la clave de *KSK* para validar un registro firmado.

Dentro de las buenas prácticas de *DNSSEC*, se especifica que las claves privadas deben mantenerse *off-line* por seguridad, y deben planificarse cambios de clave periódicos para minimizar la posibilidad de que pueda ser descubierta. La *RFC* 5011 [15] describe los mecanismos de actualización que deben ser utilizados para cambios de clave, especialmente para casos de urgencia cuando la clave privada fue comprometida.

2.3.6. Limitaciones de *DNSSEC*

DNS fue diseñado bajo la premisa de que todos los datos almacenados son, por definición, públicos. Dos usuarios diferentes que realizan consultas idénticas, van a obtener respuestas idénticas. En este aspecto, ni *DNS* ni *DNSSEC* brindan ningún mecanismo de control del tipo de listas de acceso o similares. Además, tampoco se brinda confidencialidad ya que las respuestas no se encuentran encriptadas.

DNSSEC tampoco brinda protección respecto a los ataques de denegación de servicio, y, en algunos casos, puede agravar la problemática provocada por éstos, ya que existen variaciones del ataque específicamente diseñadas para servidores de nombres de domino *DNSSEC*. Básicamente explotan dos debilidades: hecho de que se requiere un trabajo de cálculo importante para validar las firmas, y que, como las respuestas tienden a ser bastante grandes en bytes respecto al tamaño de las consultas, es posible utilizarlas como agente multiplicador durante un ataque.

Otro asunto importante, es que *DNSSEC* aumenta bastante el nivel de complejidad respecto a *DNS*, lo cual aumenta las posibilidades de que un error de configuración deje zonas aisladas. Además, esta complejidad agregada, incrementa la posibilidad de defectos en los programas utilizados, y se necesita de una infraestructura importante para soportar el proceso de firmado.

A lo anterior se suma el efecto colateral ya mencionado: al utilizar *DNSSEC* se expone el contenido de la zona entera a que pueda ser consultada. Esta información de la estructura de la zona, podría ser utilizada con malas intenciones. Esto se conoce como *Enumeración de Zona*, y se basa en la propiedad de los registros *NSEC* que apuntan al siguiente registro válido. Si bien el carácter público de los datos almacenados no fue modificado, el punto de discusión es que, cualitativamente, existe una diferencia importante entre consultar los registros asociados a partir de un nombre, y que exista un mecanismo que permita listar el contenido completo de los registros de una zona. Este último aspecto fue solucionado recientemente con una extensión a *DNSSEC* la cual tiene como novedad la creación del nuevo tipo de registro *NSEC3*, que, en lugar de apuntar al siguiente registro, guarda un *hash* del

siguiente nodo, logrando evitar así este problema.

2.4. El Servidor de Nombres *BIND*

BIND es un acrónimo que quiere decir *Berkeley Internet Name Domain* y es, hoy por hoy, la implementación más popular de un software de nombres de dominio [16]. La configuración se realiza en base a archivos de zonas, como el que se muestra en la Figura 2.2.

```
; Authoritative data for cs.vu.nl
cs.vu.nl.          86400   IN  SOA     star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.          86400   IN  TXT     "Divisie Wiskunde en Informatica."
cs.vu.nl.          86400   IN  TXT     "Vrije Universiteit Amsterdam."
cs.vu.nl.          86400   IN  MX      1 zephyr.cs.vu.nl.
cs.vu.nl.          86400   IN  MX      2 top.cs.vu.nl.

flits.cs.vu.nl.    86400   IN  HINFO   Sun Unix
flits.cs.vu.nl.    86400   IN  A       130.37.16.112
flits.cs.vu.nl.    86400   IN  A       192.31.231.165
flits.cs.vu.nl.    86400   IN  MX      1 flits.cs.vu.nl.
flits.cs.vu.nl.    86400   IN  MX      2 zephyr.cs.vu.nl.
flits.cs.vu.nl.    86400   IN  MX      3 top.cs.vu.nl.
www.cs.vu.nl.      86400   IN  CNAME   star.cs.vu.nl
ftp.cs.vu.nl.      86400   IN  CNAME   zephyr.cs.vu.nl

rowboat                    IN  A       130.37.56.201
                           IN  MX      1 rowboat
                           IN  MX      2 zephyr
                           IN  HINFO   Sun Unix

little-sister              IN  A       130.37.62.23
                           IN  HINFO   Mac MacOS

laserjet                   IN  A       192.31.231.216
                           IN  HINFO   "HP Laserjet IIISi" Proprietary
```

Figura 2.2: Archivo de zona en BIND

El registro *SOA*, indica los datos de la autoridad para esa zona y los registros *NS*, indican cuáles son los servidores de nombres de la zona. Finalmente, se encuentra la lista de registros (por ejemplo los A, PTR, CNAME, etc).

Por cada zona, se requieren dos archivos: uno para la resolución de nombres normal y otro para la resolución de nombres inversa. En éste último caso, los registros son direcciones con el formato *in-addr.arpa*. Además, es necesario un archivo más para la dirección de *loopback*.

Finalmente, como fue explicado anteriormente, el servidor de nombres necesita conocer los servidores raíz, para lo cual se utiliza otro archivo que contiene esta información. El mismo está publicado en Internet, y es responsabilidad de los administradores del servidor contar con una copia actualizada. Estos archivos indicados anteriormente se especifican en un archivo de configuración habitualmente llamado *named.conf*.

Se recomienda tener al menos dos servidores corriendo para cada zona. La configuración habitual consiste en definir uno de ellos como primario y el otro como secundario. Sus archivos de zonas serán diferentes. En el caso del primario, su registro *SOA* indicará que es maestro, y el archivo tendrá toda la información de la zona. En el secundario, su registro SOA indicará que es esclavo y desde qué *nameserver* deberá transferir su zona.

Capítulo 3

Sistemas de Registro de Nombres de Dominio

3.1. Introducción

Uno de los principales objetivos del proyecto, es adaptar el protocolo *EPP* (ver Capítulo 4) al dominio de primer nivel .UY, apuntando a obtener un registro centralizado. Por tal motivo, es de especial importancia analizar todos los aspectos y procesos asociados al registro de nombres de dominios en Uruguay, para evaluar la utilidad de *EPP* para este sistema y proponer nuevas extensiones que adapten el protocolo a esta realidad.

También puede resultar interesante realizar una investigación sobre los sistemas de registro de otros países, para obtener ideas aplicables al proyecto y a la mejora del sistema uruguayo.

En lo que sigue del capítulo, se presentan, en profundidad, las características principales del Sistema de Registro .UY, y se muestran las más sobresalientes de otros dominios que eran de interés para el usuario responsable del proyecto.

3.2. Registro de dominios .UY

Desde el año 1990, el dominio de primer nivel *.uy*, es administrado por el Servicio Central de Informática Universitaria (SeCIU). En 1994, la zona *.com.uy* le fue delegada a la Administración Nacional de Telecomunicaciones (ANTEL), quien, a partir de 2000, la administra a través de ANTEL, su servicio de datos. Queda así configurado, un escenario donde existen dos registradores (*registars*), cada uno con un registro (*registry*) propio, siendo estos dos disjuntos entre sí. Las reglas para el uso de ambos sistemas de registros, se encuentran definidas en el Instructivo Técnico de por SeCIU [17] y en las Condiciones de Uso de ANTEL [18].

El registro de nombres de dominio de Internet, implica diversas obligaciones por parte de los *registrants*, que pueden ser las personas que tengan interés directo en registrar dichos nombres, o intermediarios contratados por dichas personas, que además, pueden ofrecer otro tipo de servicios, como el alojamiento o *hosting*.

En ambos casos, el procedimiento de registro es el mismo, definiendo cada *registry*, por separado, cuál es este procedimiento. En las sub-secciones a continuación, se presentan los puntos necesarios para comprender los procedimientos de registro en cada *registry*, y para evaluar sus necesidades y posibilidades de mejora. La información aquí presentada, fue extraída de las respectivas páginas web de ambas instituciones.

En un futuro, se planea abrir el negocio de registro de nombres de dominio a otras instituciones registradoras, utilizando para ello el protocolo *EPP* para

comunicarse con una base de datos centralizada en SeCIU. Además, se planea abrir el registro de dominios plano .UY (sin dominio de segundo nivel, por ejemplo, *presidencia.uy*) y el registro de nombres *IDN*.

3.2.1. Registro de Nombres de Dominio en SeCIU

Se puede dividir el registro de dominios en dos aspectos importantes: las entidades involucradas y los procedimientos para registrarlas. A continuación, se describen los datos recabados sobre las entidades y los procedimientos ejecutados para registrar nombres. Luego, se describen los procedimientos que permiten modificar los datos de los dominios registrados, y realizar sus bajas.

Entidades involucradas en el registro de Dominios

El registro de los dominios administrados por SeCIU (*.org.uy, .net.uy, .edu.uy, .mil.uy* y *.gub.uy*) comprende dos etapas: el registro de personas y el registro del dominio propiamente dicho. Durante el registro de personas, se registran individuos que luego podrán asociarse a dominios, como titular o como alguno de los contactos del mismo. Para registrarse, se deben proporcionar datos que permiten identificar y contactar a la persona:

- Nombre de la organización, institución o persona física
- Número de *RUT*[1], o de Cédula de identidad (si se trata de una persona física).
- Dirección (postal)
- Localidad
- Departamento
- Teléfono
- Fax
- Correo electrónico.

[1] *Registro Único Tributario*, registro de contribuyentes al sistema tributario en Uruguay. Proporciona el número identificador para todos los contribuyentes que tributan alguna clase de impuestos ante la Dirección General Impositiva.

El nombre, la dirección, la localidad y el departamento, y el correo electrónico, son datos obligatorios. Además, no puede existir otra persona registrada con el mismo correo. A todas las personas registradas, sean físicas o jurídicas, se les asigna un identificador denominado *NIC-Handle*, el cual puede ser usado para vincular a la persona a diversos dominios.

Para el registro de dominios, es necesario contar con un nombre de dominio que no haya sido registrado previamente y una descripción de la actividad a la cual está destinado el nombre. Además, se necesita asociar al dominio personas que cumplan los siguientes roles:

Titular: es el titular para el uso del dominio

Contacto Administrativo: es la persona que será designada a efectos de relacionarse con SeCIU en cuanto a la comunicación de los actos de disposición, modificación o cancelación del registro, a quien serán dirigidas las comunicaciones concernientes al cobro, dudas de datos o citaciones.

Contacto Técnico: es quien se encargará de relacionarse con el SeCIU en los aspectos técnicos de la gestión del nombre de dominio.

Contacto de Pago: es el encargado de relacionarse con SeCIU en los aspectos concernientes al cobro de las tarifas correspondientes. A este contacto serán dirigidas todas las comunicaciones concernientes al cobro.

A los efectos de la delegación de autoridad en la resolución del dominio, se deben proporcionar 2 servidores, uno primario y otro secundario, que contengan la información autoritativa correspondiente. Adicionalmente, se pueden agregar hasta 11 servidores más. En todo caso, si el servidor a asociar se encontrase dentro del dominio objeto del registro, se necesita, además, su dirección en formato *IPv4* o *IPv6*.

El procedimiento para el registro de estos datos y el registro de los dominios, ha sido modificado recientemente (setiembre de 2009), cambiando sustancialmente la forma en que éste se realiza. Entre otras cosas, es un sistema basado en usuarios registrados, que pueden realizar las diferentes operaciones de alta, baja o modificación. A pesar de esto, las consideraciones aquí hechas, siguen siendo la base conceptual para los procesos actuales para el registro de dominios, los cuales son presentados a continuación, contrastados con los anteriores.

Procedimientos para el registro de dominios

Previo a la implantación del actual Sistema Integral de Registro de Dominios (ver Sección 3.2.2), en el sistema anterior, el registro de personas y de dominios, se realizaba mediante dos formularios distintos: uno para las personas y otro para los dominios. En el formulario de personas, se registraban los datos de una persona para actuar como titular o alguno de los contactos de los dominios a registrar, debiéndose completar este procedimiento para cada persona nueva a registrar. Este proceso, culminaba con la asignación del identificador antes mencionado (*NIC-Handle*). En la actualidad, el registro de personas, está integrado al registro de dominios en un único *wizard* o asistente, que guía al usuario durante los pasos del registro.

El primer paso para el registro de dominios, es indicar el nombre del dominio, el dominio de segundo nivel que corresponda (*.org.uy, .net.uy, .edu.uy, .gub.uy,* o *.mil.uy*) y una descripción de la actividad a la que se destina el dominio. Adicionalmente, para los dominios educacionales (*.edu.uy*) se debe indicar si el dominio es para una institución pública, ya que estas no deben pagar ninguna tarifa por el dominio.

A continuación, se deben indicar el titular del dominio, y los contactos administrativo, técnico y de pago. El procedimiento para estas personas es idéntico: se puede indicar el *NIC-Handle* de una persona ya registrada, o indicar los datos de una persona nueva. Si se indican los datos de una persona nueva, a dicha persona se le asigna su respectivo *NIC-Handle* para poder ser utilizado en las demás asociaciones del dominio con las personas. Aquí, se plantea otra diferencia al proceso previo: las personas nuevas, actualmente, no son registradas hasta que se completó el proceso de registro del dominio (si bien sus *NIC-Handle* están disponibles durante el mismo). En resumen, si un usuario inicia un proceso de registro de dominios, creando nuevas personas, y no lo finaliza, dichas personas no podrán ser utilizadas en nuevos procesos, salvo que se ingresen nuevamente todos sus datos.

Una vez asociadas las personas a los distintos contactos, se deben indicar los servidores de nombres que responderán de manera autoritativa a las consultas sobre el dominio. Se precisan 2, uno primario y otro secundario, y, en el sistema actual, no así en el anterior, se pueden indicar hasta 11 servidores adicionales. Luego de indicados los servidores, el usuario tiene la opción de seguir registrando más dominios, dentro del mismo proceso, o de generar el documento de pago correspondiente a los dominios registrados durante éste. El usuario tiene la opción de elegir el período de tiempo por el cual registrar el dominio (entre 1 y 5 años), cuando antes se registraban, por defecto, durante 1 solo.

Inmediatamente de finalizado el procedimiento, de manera automática,

todos los dominios registrados son activados, y, 20 minutos después, quedan disponibles en Internet. Anteriormente, antes de activar el dominio registrado (recordar que en el procedimiento se podía registrar 1 por vez), se seguían algunos pasos de verificación que requerían el envío de faxes, y la intervención de un operador humano, quien, a veces, realizaba llamadas telefónicas para confirmar la validez del dominio registrado. El plazo disponible para hacer el pago correspondiente, es de 10 días, luego de los cuales el dominio es dado de baja.

En el caso de los dominios gubernamentales, militares, o de educación pública, tanto en la actualidad como en el pasado, se requiere un paso de verificación adicional, que implica el envío de una nota oficial que certifique que el/los dominio/s, efectivamente, pertenecen a una institución de los tipos mencionados. Los dominios con estas características, no son activados mientras no se reciba dicha nota.

Procedimientos para bajas y modificaciones de Dominios

En el sistema anterior, cada dominio tenía una clave secreta asociada, que los usuarios podían utilizar para acceder a las modificaciones o bajas de dominios. Quien tuviera dicha clave, podía acceder a un formulario para la modificación, o a uno de baja, para el dominio asociado a la clave ingresada.

Para las modificaciones, se disponía de un formulario similar al de alta del dominio, con todos los campos modificables en blanco. El usuario, completaba los que deseaba modificar y enviaba el formulario. No se podían modificar el nombre del dominio, la descripción de la actividad, y el nombre del titular, y si se deseaba modificar los servidores de nombres, había que llenar los dos pares $< nombre, IP >$, modificando el que se deseara y poniendo los datos ya registrados en el que no (o modificando ambos). En todos los casos, los campos que no se completaban quedaban tal como estaban.

Para el caso que se deseara iniciar el proceso de baja de un dominio, el usuario ingresaba con la clave asociada al mismo al formulario de bajas, y como resultado de esto, se enviaba un correo de confirmación al contacto administrativo del dominio.

En ambos casos, bajas y modificaciones, como resultado se obtenía una página que debía ser impresa, firmada, y enviada por fax a SeCIU, para confirmar la modificación/baja del dominio. Si el dominio en cuestión era militar (*.mil.uy*) o gubernamental (*.gub.uy*), se requería, además, que dicha página fuera impresa en papel membretado o acompañada de nota oficial.

Como ya fue mencionado más arriba, el sistema actual está basado en usuarios. Los usuarios se registran con sus correos electrónicos, y, al ingresar al sistema, el usuario adquiere los permisos de aquellos contactos cuya direc-

ción de correo coincide con la identificación del usuario. Los permisos que se pueden adquirir, según el tipo de contacto, son:

- El *contacto administrativo* tiene autorización para modificar todos los datos del dominio, excepto la titularidad, y para gestionar los aspectos relativos al pago. También tiene autorización para iniciar el procedimiento de baja, el cual deberá ser autorizado por el titular del dominio.
- El *contacto técnico* tiene autorización para modificar, solamente, los datos de los servidores de nombres.
- El *contacto de pago* únicamente puede gestionar los aspectos relativos al pago.

Existen, además del procedimiento iniciado por el titular o el contacto administrativo, circunstancias bajo las cuales los dominios son dados de baja automáticamente: se pueden dar de baja dominios por solicitud de una autoridad competente, o por el no pago de la renovación.

Continúa existiendo el concepto de *dominio critico*: un *dominio crítico*, sólo puede ser dado de baja de manera manual por parte del registro central (SeCIU).

3.2.2. Sistema Integral de Gestión de Dominios

El *SIGD* (Sistema Integral de Gestión de Dominios), es la aplicación desarrollada por SeCIU para efectuar su rol de *registrar*, comunicándose con el software de *registry* utilizado.

Ofrece funcionalidades relativas al registro de dominios y a la facturación de los mismos, como la generación, búsqueda, eliminación y cancelación de documentos de pago.

También se ofrecen funcionalidades asociadas a la contabilidad del sistema. Entre ellas se encuentran el alta, baja, y modificación de tarifas, el manejo de los diversos tipos de tarifas y de las bonificaciones. Las funcionalidades contables permiten, además, recibir los datos de facturas pagas en centros externos a SeCIU, y diversas consultas sobre el estado contable del sistema.

Es importante destacar el concepto de *dominio crítico*. Un dominio es crítico si no puede ser dado de baja, salvo de manera manual. Esto significa que, un dominio crítico, no será borrado automáticamente, por ejemplo, a pesar de que haya expirado su período de registro, o que se haya excedido el plazo para el pago de la factura de alta. Los dominios críticos se configuran directamente sobre la base de datos (*SIGD* no ofrece una funcionalidad para

incluir o excluir un dominio de la categoría *crítico*). Existen dos maneras para dar esta calidad a los dominios:

Por extensión Se indica un conjunto de dominios a través de su extensión. Todos los dominios que tengan una extensión considerada crítica, serán considerados como críticos. Por ejemplo, si se dice que los dominios *.gub.uy* son críticos, entonces el dominio *presidencia.gub.uy* es crítico.

Por atributo Para cada dominio que se considera crítico, se marca un atributo que lo define como tal.

Arquitectura

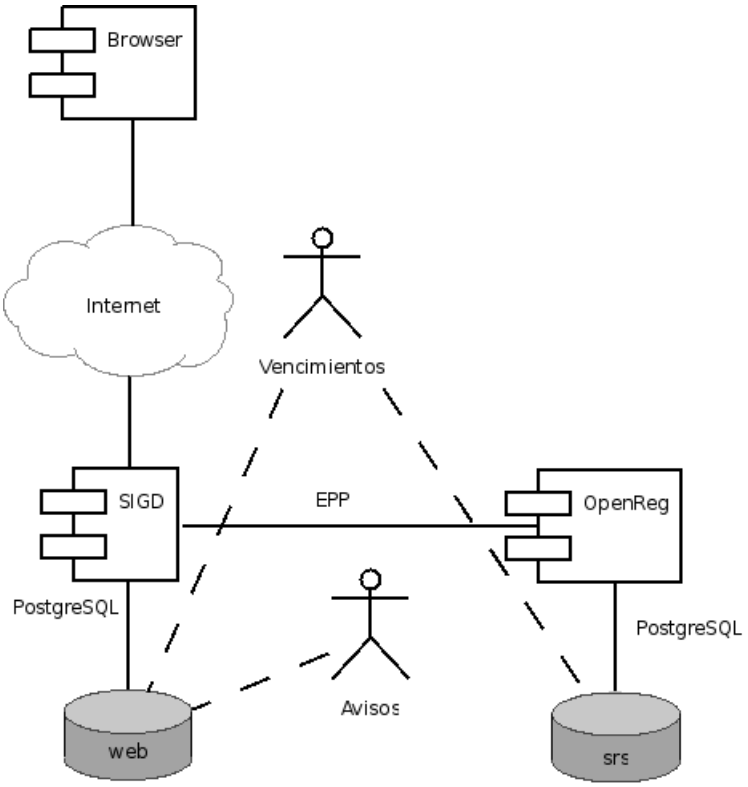


Figura 3.1: Arquitectura del SIGD

La arquitectura del *SIGD* puede apreciarse en la Figura 3.1. Como allí se muestra, es una aplicación Web monolítica que ofrece sus funcionalidades a través de diversas páginas. Dicha aplicación reside en un servidor Web, y accede a una base de datos que replica la base de *OpenReg*, agregando las

tablas necesarias para cumplir con las funcionalidades específicas del *registrar*, como ser facturación, manejo de usuarios, etc. El acceso a la base de datos se realiza utilizando *JDBC* estándar, encapsulando la lógica de acceso en una clase, y responsabilizando de dicho acceso a las diversas clases que representan las entidades de la base de datos.

Existen, además de la aplicación web, algunos demonios encargados de manejar los vencimientos de los pagos de los dominios registrados. Uno, se encarga de enviar los avisos automáticos (por correo electrónico) a los contactos de los dominios cuyo pago está por vencer, para lo cual analiza la información de la base *web*. El segundo, realiza las tareas administrativas relativas a los vencimientos efectivos de dichos dominios (suspensión y baja definitiva). Para esto, analiza la información almacenada en la misma base, y envía los comandos *EPP* correspondientes a la modificación y borrado de los dominios al servidor *OpenReg*.

En un servidor diferente, se encuentra activo el servicio de *OpenReg*, con el cual *SIGD* se comunica utilizando un transporte *TCP* estándar. Para la comunicación este servidor, se utiliza la librería *EPP Registry/Registrar Toolkit (epp-rtk)*.

3.2.3. Registro de Nombres de Dominio en ANTEL

El registro de Nombres de Dominio *.com.uy*, como se mencionó anteriormente, está delegado a ANTEL. Dicha delegación, se rige por normas contenidas en un convenio específico, que determina, entre otros aspectos, los datos a guardar para el registro de dominios. Dichos datos, no difieren de los registrados por SeCIU, siendo la única diferencia importante, el proceso mediante el cual se registran los dominios [19].

Para registrar un dominio *.com.uy*, en primera instancia se debe chequear la disponibilidad del nombre. Si el nombre está disponible, se puede proceder a registrarlo. En caso de que el usuario no estuviera *loggeado* en el sistema, al elegir registrar el dominio, se lo redirige a una página para que realice el *login*. Es necesario contar con una cuenta de ADINET[2] para poder acceder al registro de dominios en ANTEL.

El primer paso del registro es indicar titular y contactos para el dominio. El usuario registrante, tiene la opción de indicar un *NIC-Handle* existente, o crear un contacto nuevo. En este caso, se tiene la opción de crear contactos o titulares (se registran en formularios separados), a diferencia del registro de SeCIU, por separado del proceso de registro de dominios. En el caso de los contactos, el nombre se ingresa en campos separados, y en el caso de los

[2]Portal de ANTEL que brinda diversos servicios, http://www.adinet.com.uy

titulares, se ingresa en un único campo, ya que se contempla explícitamente la posibilidad de titularidad para organizaciones. Luego se ingresan dirección, correo electrónico, teléfono y, en el caso de los titulares, el fax. Una vez confirmados los datos, se ingresa la información de pago.

Para pagar el registro de dominio, el usuario debe registrar sus datos en Idóneo Pagos[3] (también utilizando la cuenta de ADINET para autenticarse). Este registro se hace por única vez.

Se dispone de una página de administración, donde el usuario podrá consultar el estado de sus dominios y realizar procesos de modificación y baja. Los estados de un dominio, pueden ser los siguientes:

Pendiente de pago. No se terminó de pagar el dominio.

En proceso de alta. Se registró el dominio y hay que esperar unos minutos para que se active.

Activo. El dominio está pagado y funcionando.

Por defecto, el dominio es alojado en los servidores de nombres de ANTEL. Para modificar esto, el usuario debe acceder a la página de administración del dominio, donde deberá realizar la delegación correspondiente. En dicha página, además, el usuario podrá modificar los registros del dominio (opción administrar), cambiar los contactos técnico o financiero, o iniciar el proceso de baja del dominio.

La única persona capaz de realizar todas estas modificaciones, es el administrador del dominio, o sea, quien lo registra.

Los datos correspondientes a *todos* los dominios bajo .UY, se almacenarán en una base de datos centralizada administrada por SeCIU, la cual será accedida por ANTEL a través de un servicio de altas, bajas y modificaciones, basado en el protocolo *EPP*.

3.2.4. Validación de nombres de dominio

Los nombres de dominio a registrar, deben, en ambos casos (SeCIU y ANTEL), cumplir con ciertos requerimientos. En resumen, serán validos nombres de dominio conteniendo únicamente caracteres alfanuméricos y guiones (*ASCII* 0x2D), no pudiendo ser un guión el primer carácter del nombre [20].

Existen restricciones sobre los nombres de dominio posibles, como por ejemplo, que los nombres elegidos no atenten contra la moral y las buenas

[3]Sistema de pagos para comprar productos o servicios a través de Internet con cargo a una factura de ANTEL, http://www.pagos.adinet.com.uy

costumbres. Además, el nombre elegido no podrá comenzar con la secuencia de caracteres `xn--`, ni contener un guión en la tercer o cuarta posición. Tampoco se podrán elegir nombres que sean confundibles con instituciones u otras oficinas del Estado, o con organizaciones internacionales, aún cuando se trate de diferentes sub-dominios.

Tampoco se admitirán nombres que sean *iguales, análogos o confundibles con algún derecho de propiedad intelectual.* A los efectos de resolver conflictos al respecto, SeCIU, define un mecanismo de arbitraje basado en el reglamento de arbitrajes definido por el Centro de Conciliación y Arbitraje[4] de la Cámara de Comercios y Servicios del Uruguay[5].

3.3. Registro de dominios .CL

El Registro de Nombres del Dominio CL, denominado *NIC-Chile*[6], es administrado por el *Departamento de Ciencias de la Computación de la Universidad de Chile*[7] y posee ciertas características (por ejemplo, ver más abajo, *Período de Publicidad*) que hacen interesante su estudio. Su respectivo reglamento[21], establece las condiciones para el registro, las cuales se resumen a continuación.

Pueden solicitar nombres de dominio .CL, personas naturales, residentes o avecinadas en Chile, y personas jurídicas, públicas o privadas, y Corporaciones de derecho público o privado, constituidas en Chile, o debidamente autorizadas para operar en Chile. Las personas naturales o jurídicas que no residen en Chile, deben realizar el trámite a través de un representante con domicilio en el país, quien actuará como contacto administrativo, y será considerado como el solicitante a los efectos de la reglamentación.

Las solicitudes se envían de manera electrónica, y una vez recibidas, son publicadas por *NIC-Chile*, para que, en un plazo de 30 días, denominado *Período de Publicidad*, quienes se sintieran afectados por la solicitud ingresada, pudieran presentar las suyas para ese nombre de dominio. Cada solicitante dispone un plazo de 20 días para hacer efectivo el pago de la solicitud, luego del cual, se considerará que el solicitante desistió de adquirir el nombre.

Una vez recibido el pago y transcurrido el período de publicidad, si no hubieran más solicitudes para el nombre involucrado, *NIC-Chile*, lo asignará al solicitante. Si al final del período de publicidad, existieran varias solicitudes en trámite para el nombre, se iniciará un procedimiento de mediación y

[4] http://www.arbitraje.com.uy/
[5] http://www.camaradecomercio.com.uy/
[6] Network Information Center Chile, http://nic.cl
[7] http://www.dcc.uchile.cl

arbitraje, mediante el cual, el nombre será finalmente asignado [21].

Para ingresar una solicitud, se deben indicar el número de *RUT*[8] y un correo electrónico para la comunicación oficial entre el *NIC* y el titular del dominio. Opcionalmente, asociados al titular, se pueden ingresar observaciones, un teléfono, y el giro en el cuál se desempeña el solicitante. Luego, se piden un contacto administrativo y uno técnico, los cuales deben tener asociados un nombre, una organización (se pide el nombre de la misma), el correo electrónico, la dirección, y el teléfono (el cual es opcional para el contacto administrativo).

La información de *DNS* es opcional, y consiste del nombre y la dirección *IP* de los servidores. Se pueden ingresar un servidor primario, y hasta 3 servidores secundarios, y, opcionalmente, 4 más, llegando hasta un máximo de 7. Los usuarios pueden utilizar un servicio de *DNS* secundario provisto por *NIC-Chile*.

Los datos requeridos para la facturación, incluyen el número de *RUT*, el domicilio, la comuna[9], la ciudad, el giro comercial, y la persona a la cual se enviará la factura. Existe la posibilidad de que estos datos, sean los mismos que luego se utilizarán para la generación del cupón de pago. El cliente tiene la opción de que la factura, le sea enviada en forma electrónica, o por correo tradicional.

Para finalizar el registro, el cliente confirma los datos proporcionados, y debe indicar su conocimiento y aceptación de las normas que rigen el registro de dominios .CL. Una vez hecho esto, se informa de la inscripción del dominio, y se muestra un enlace para consultar las diferentes formas de pago.

La generación de los archivos de la zona CL (para el sistema de *DNS*), se realiza cada 30 minutos, con las modificaciones recibidas hasta 10 minutos antes. Esto, incluye dominios nuevos que han sido pagados, o dominios antiguos en los cuales se actualizó la información de servidores de nombres.

Para modificar los datos de un dominio, el cliente debe ingresar el nombre de dicho dominio, y luego, a través de un formulario, cambiar los datos que desee. No es posible modificar ni el número de *RUT* ni el nombre del titular, y, para realizar cualquier modificación, se requiere de un código de verificación que es enviado por correo electrónico a las casillas de correo vigentes para el dominio.

El cliente, puede dar de baja un dominio accediendo directamente al sitio de *NIC-Chile* utilizando un código de autorización que es enviado a las casillas registradas para contacto del dominio. Si el titular del dominio, no tiene

[8]Rol Único Tributario: número que identifica a todas las personas naturales o jurídicas en Chile, susceptibles de pagar impuestos

[9]Las comunas de Chile son la división político-administrativa menor y básica del país. Corresponde a lo que en términos genéricos se conoce como municipio.

acceso a dichas casillas, puede generar un formulario para solicitarlo, el cual debe ser enviado vía fax.

Una vez que el cliente tiene el código de autorización, puede modificar el nombre de dominio y quitar los servidores de nombres, y en la siguiente generación de zona, dicho dominio no será tenido en cuenta.

Además, son causales de baja automática:

- No tener el pago al día. Para el caso de los dominios que cumplieron su fecha de expiración, se avisa por correo electrónico de la baja.
- No tener servidores de nombres asociados.
- Tener servidores de nombres que dependan de otros dominios .CL que no están en la zona.
- Por orden judicial.

3.4. Registro de dominios .PL

Los nombres de dominio polacos (dominio .PL) son administrados por *NASK*, instituto pionero en el desarrollo de Internet en Polonia. Una de las particularidades de este registro, es la utilización de *EPP* en las transacciones con los *registrars*. Además de utilizarlo, lo utilizan extendido, lo cual resulta de sumo interés para el trabajo aquí presentado.

3.4.1. Reglas para el registro de Nombres

Para registrar un nombre de dominio, se debe entrar en un acuerdo en el cual intervienen las siguientes partes:

NASK. Es la Red de Investigación y Académica de Computadoras, una entidad de Investigación y Desarrollo con sede en Varsovia, Polonia.

Socio. Es una entidad asociada a *NASK* a través de un acuerdo de cooperación para el servicio técnico y administrativo.

Ofertante. Es una entidad o persona natural que ha enviado una *oferta* a *NASK* directamente o a través de un *Socio*, para entrar en el acuerdo. El *ofertante* propone una *oferta* para entrar en el acuerdo.

Suscriptor Es un *ofertante* que ha entrado en el acuerdo con *NASK*

La *oferta* se realiza a través del sitio web de *NASK*[10]. Implica proveer un nombre para el dominio, la información de los servidores de nombres, y los datos del *ofertar*, sea una persona natural o de otro tipo.

Luego de ingresada la *oferta*, se debe obtener una versión impresa, la cual debe ser claramente firmada y enviada por fax en un plazo no mayor a 7 días. Todas las *ofertas* tienen asignado un identificador único a los efectos del trato entre el *ofertar* y *NASK*.

Cualquier modificación o baja de dominios, debe ser realizada por escrito, a través de formularios que están disponibles en el sitio correspondiente [22].

3.4.2. Utilización de *EPP*

NASK actúa como un *Thin Registry*[11] centralizado, permitiendo que compañías o instituciones especializadas en el registro de nombres realicen sus registros utilizando *EPP*. El servicio *EPP* ofrecido por *NASK*, incluye ciertas modificaciones al mapeo básico del protocolo para el registro de nombres de dominio.

Se extiende el comando *create* para posibilitar la reserva de nombres, agregando la extensión *book*. Cuando un *registrar* envía una solicitud para crear un dominio con esta extensión, nadie podrá solicitar el nombre por un determinado período de tiempo.

Se crea el objeto *future*, que representa la intención de un registrante de obtener un nombre de dominio cuando éste quede disponible.

Otra extensión es la realizada al comando *create*, al cual se le agrega el elemento *reason* para que el *registrar* exprese la razón por la cual debería obtener dicho nombre de dominio.

También es posible, a través del agregado del elemento *consentForPublishing* a los comandos *create*, *update*, y *info*, en caso de que el contacto involucrado sea una persona privada, ocultar los datos privados de dicha persona a las consultas *WHOIS* que la impliquen.

Por último, existe el concepto de *individual*. A través del elemento *individual* agregado a los comandos *create*, *update*, y *info*, es posible indicar si el contacto involucrado en el comando, representa una persona privada.

Algunas de estas extensiones, están explicadas con mayor detalle en la Sección 5.5.

[10] http://www.dns.pl

[11] Un registro fino, o *thin registry*, es aquel en que parte de la información social asociada a las entidades registradas, está distribuida entre un registro compartido y los registradores que lo utilizan [23]. En contraposición, un registro grueso, o *thick registry*, es aquel que centraliza toda la información sobre los dominios, incluyendo, por ejemplo, datos de facturación [24].

3.5. Registro de dominios .BR

El Registro de Nombres del Dominio BR, es gestionado por *Registro.br*[12], el cual es parte del *NIC.br*[13]. Esta organización es la encargada de gestionar diversos aspectos relativos a Internet en Brasil. Se encuentra dividida en diferentes áreas según sus funciones. *Registro.br*, como ya fue mencionado anteriormente, es la encargada del registro de dominios. Otras secciones son *CERT.br*[14], *CETIC.br*[15], *CEPTRO.br*[16] y *W3C Brasil*[17].

Los pasos para registrar un nuevo dominio son similares a los de registros vistos, contando con una interfaz web para ello. A diferencia de estos, el registro de Brasil cuenta con un sistema de *ticket* electrónico integrado al proceso de registro.

Registro del usuario. Se solicitan los datos personales habituales (nombre, e-mail, dirección, teléfono y contraseña). Luego de completarlo, se envía un e-mail con un enlace al cual el solicitante debe ingresar dentro de las 24 horas para confirmar el registro. Una vez hecho esto, se recibe por e-mail un identificador del contacto. En caso de contar además con un contacto técnico o de pago, se debe repetir el proceso.

Ingresar al sistema. Se debe ingresar al sistema con el identificador del paso anterior y la contraseña correspondiente.

Registrar el nuevo dominio. Se debe completar el nombre, el sub-dominio, y número de *CNPJ* o *CPF*[18]. Es posible especificar un contacto técnico o de pago (se ingresan los identificadores devueltos por el primer paso). Si se omiten se asume que el usuario que está registrando el dominio tomará estos roles. Son necesarios al menos dos servidores *DNS* configurados para hacer efectivo el registro. Si se omiten durante el registro, se tienen dos semanas para informarlos (utilizando el número de *ticket* que brinda el sistema).

[12]Registro de Domínios para a Internet no Brasil, http://registro.br

[13]Núcleo de informação e Coordenação de Ponto BR

[14]Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, http://www.cert.br

[15]Centro de Estudos sobre as Tecnologias da Informação e da Comunicação, http://www.cetic.br

[16]Centro de Estudos e Pesquisas en Tecnologia de Redes e Operações, http://www.ceptro.br

[17]World Wide Web Consortium Brasil, http://www.w3c.br

[18]Cadastro de Pessoas Físicas

Envío del formulario. Al completar el formulario, si no ocurre ningún error en la validación automática, el registrante recibe por correo electrónico su número de ticket. Este estará disponible para consultar mientras la solicitud no se completa (se activa el dominio o se cancela la solicitud)

Espera del registro. Si el dominio que desea está en una categoría que necesita documentación específica, la solicitud quedará en espera hasta recibir la misma para su liberación. Si la documentación no es recibida dentro del tiempo estipulado, la solicitud se cancela automáticamente. Las solicitudes canceladas no se pueden reactivar, siendo necesario realizar nuevamente el registro. Cuando el dominio está registrado o se cancela el *ticket*, se le notifica por e-mail al solicitante.

Después del registro. Después de recibir el correo electrónico con la confirmación y las instrucciones de pago, el dominio quedará visible en Internet en la próxima publicación de *DNS* (30 minutos).

3.5.1. Dominios de segundo nivel

El registro de dominios de Brasil tiene la particularidad de que se encuentra subdividido en una gran cantidad de dominios de segundo nivel.

Genéricos. Los sub-dominios genéricos son *com.br* y *net.br*. Pueden ser solicitados tanto por personas físicas como jurídicas.

Para personas jurídicas. A modo de ejemplo se citan *agr.br* (para empresas agrícolas y granjas) *art.br* (para artes: música, pintura, etc) y *b.br* (bancos). La lista es bastante amplia y pueden consultarse en *http://registro.br/info/dpn.html*.

Para profesionales libres. Esta categoría es sólo para personas físicas, y contempla muchas profesiones. A modo de ejemplo *adm.br* (administradores), *adv.br* (abogados) y *arq.br* (arquitectos). El listado se encuentra en *http://registro.br/info/dpn.html*.

Para personas físicas. Este grupo está compuesto por los siguientes subdominios: *blog.br*, *flog.br*, *nom.br*, *vlog.br* y *wiki.br*.

Algunos sub-dominios dentro de la categoría de personas jurídicas requieren documentación especial:

- Para *am.br* es necesario el número de *CNPJ* y el comprobante de *ANATEL* para radiodifusión sonora AM.

- Para *coop.br* es necesario el número de *CNPJ* y el comprobante de registro en la Organización de Cooperativas del Brasil.
- Para *edu.br* el número de *CNPJ*, comprobación de la actividad específica a través de un documento del MEC, y algún documento que compruebe que verifique que el nombre a registrar no es genérico.
- Para *fm.br* el número de *CNPJ* y el comprobante de *ANATEL* para radiodifusión sonora FM.
- Para *g12.br* el número de *CNPJ*.
- Para *gov.br* el número de *CNPJ* y comprobación de que la entidad pertenece al gobierno federal.
- Para *mil.br* se requiere la autorización del Ministerio de Defensa.
- Para *org.br* el número de *CNPJ* y documentación probatoria de la naturaleza no gubernamental sin fines de lucro. En los casos en que la institución es un consulado o una embajada, se requiere *CNPJ*.
- Para *psi.br* el número de *CNPJ* y comprobación de que la entidad es un proveedor de acceso a Internet.

3.5.2. Soporte para *DNSSEC*

El Registro de Brasil tiene disponible utilizar *DNSSEC* para todos sus subdominios, sin embargo, para los subdominios *b.br* y *jus.br*, su uso es obligatorio. Estos dos subdominios corresponden a entidades bancarias y a entidades del Poder Judicial respectivamente.

Capítulo 4

EPP, Protocolo Extensible de Provisión

4.1. Introducción

EPP es un protocolo flexible diseñado para la asignación de objetos dentro de registros en Internet. El mismo, surge como resultado del grupo de trabajo *provreg*[1] de la *IETF*, que vio la necesidad de desarrollar una herramienta que permitiera la interacción entre aplicaciones propias de los diversos *registrars* y diferentes *registries*.

Dicho grupo plantea la creciente distinción, en la Administración del registro dentro del Servicio de Nombres de Dominio (*DNS*), entre la operación de un Registro *back-end* para almacenar los objetos registrados, y servicios de *front-end* brindados por los *registrars* a los *registrants*. Especialmente, fija la necesidad de permitir el acceso de diversos *registrars* a un mismo *registry*. De manera opuesta, interesa que un *registrar* pueda acceder a diversos *registries*, sin importar el protocolo, incluso si los *registries* difieren en su modelo operacional.

La solución propuesta, proporciona un servicio de provisión de nombres de dominio, donde *provisión* significa registrar, renovar, modificar, borrar y transferir los objetos provistos [25]. Esta solución, es un protocolo con estado, transaccional e idempotente, basado en *XML*, y su última versión (en Julio de 2009) se encuentra especificada en la *RFC* 4930 [26].

4.2. El protocolo *EPP*

EPP es un protocolo *XML* con estado para la capa de aplicación que puede soportar varios protocolos de transporte. Protegidos usando la seguridad de capas inferiores, los clientes intercambian datos de seguridad y opciones disponibles, para luego entrelazarse en un diálogo (iniciado por el cliente) de comando-respuesta.

Todos los comandos son atómicos, es decir, terminan o fallan completamente, además de idempotentes, ya que la aplicación repetida del mismo comando, tiene el mismo efecto de red que haberlo aplicado una sola vez. En la Figura 4.1 se muestra la máquina de estados que define el protocolo, con las transiciones que producen los diversos comandos y eventos.

[1] http://www.ietf.org/wg/concluded/provreg.html

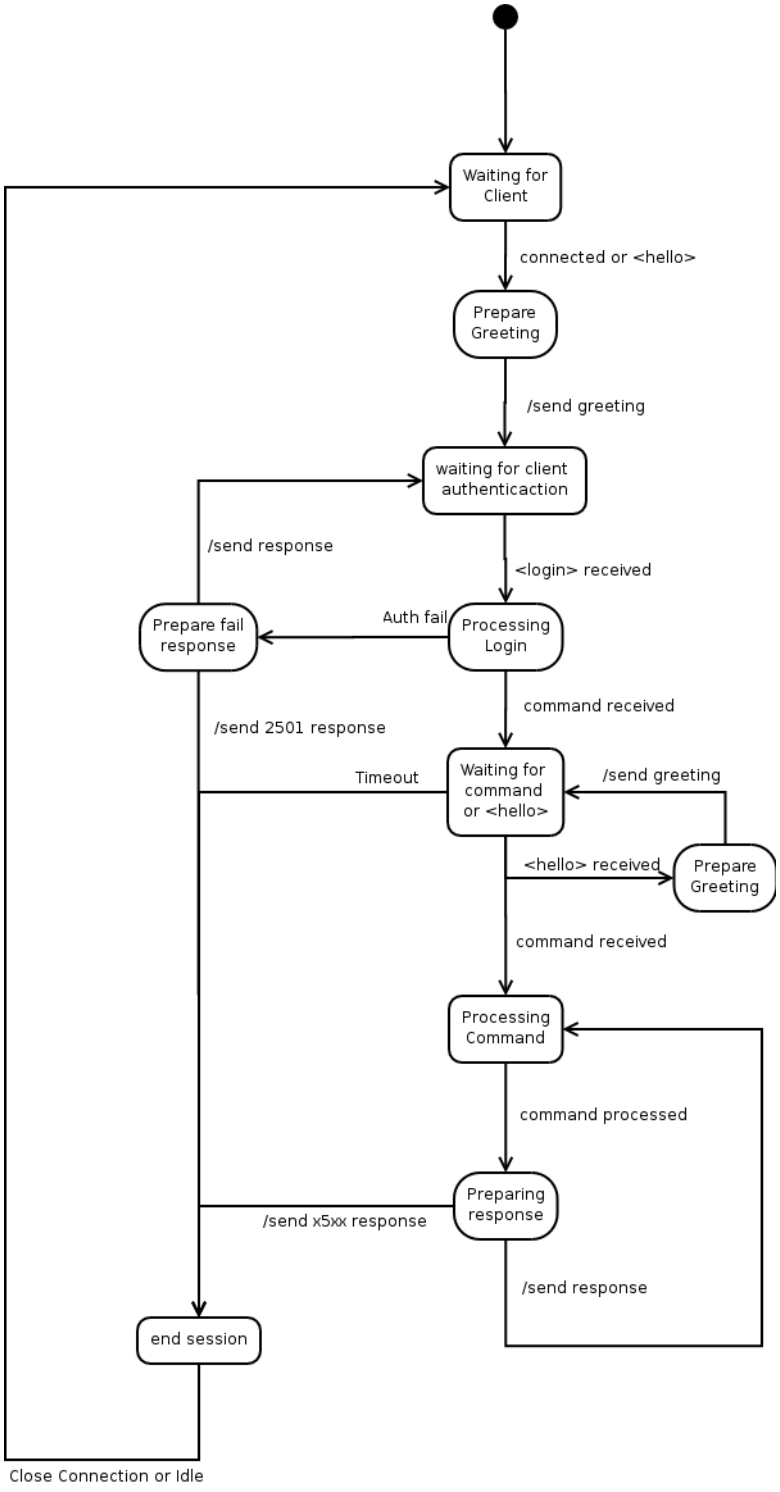


Figura 4.1: Máquina de estados del *EPP*

4.2.1. Comandos del *EPP*

Manejo de sesiones *EPP*

Para manejar las sesiones de usuario, se proveen los comandos *login* y *logout*. En respuesta a un *greeting* (saludo) enviado por un servidor *EPP* activo, el cliente responderá con un comando *login*, indicando información de usuario y datos sobre los objetos a manejar y opciones tomadas, para establecer una sesión de trabajo con el servidor. La sesión abierta, se terminará cuando el servidor la corte por inactividad o duración excesiva, o cuando el cliente envíe un comando *logout*.

EPP check

El comando *check*, permite chequear la disponibilidad de un objeto en el repositorio. En un mismo comando *check*, se puede consultar por varios objetos. La respuesta a este comando, indica si el objeto está disponible o no, y, opcionalmente y en caso de no disponibilidad, un texto (específico de cada servidor) indicando la razón de la no disponibilidad.

EPP info

Para obtener información sobre los objetos almacenados, los clientes autorizados disponen del comando *info*. Con dicho comando, el servidor mostrará la información sobre el objeto consultado, que es específica de cada tipo de objeto, además del *Identificador de Objeto del Repositorio (ROID)*.

EPP poll

Con el comando *poll*, los clientes obtienen los mensajes de servicio que les envía el servidor. Estos mensajes pueden ser creados ante eventos que no requieren intervención alguna de un cliente, como ser borrado o renovación automática por expiración, o como resultado de comandos que requieran notificaciones (además de la respuesta al ejecutor del comando, la cual no debe ser sustituida por el mensaje de servicio). Asimismo, algunos comandos, como la creación de ciertos objetos, pueden requerir un procesamiento *off-line*. En estos casos, luego de ejecutado el comando, se responde inmediatamente anunciando que la ejecución de la acción solicitada, fue retrasada. A medida que dicha solicitud se va procesando, el cliente es notificado mediante mensajes. El servidor, envía los mensajes de a uno, y los va desencolando a medida que obtiene los acuses de recibo del cliente.

EPP transfer

El comando de consulta *transfer*, permite, a los clientes autorizados, consultar el estado de las solicitudes de transferencia pendientes o completadas. En respuesta a este comando, el servidor indica algunas informaciones administrativas sobre el estado de transferencia del objeto consultado. El asunto de las transferencias, será abordado más adelante en éste capítulo.

Comandos de modificación

Por último, se puede alterar el estado de objetos del repositorio con los comandos de transformación de objetos. Utilizando los comandos *create, delete* y *update*, los clientes pueden realizar altas, bajas y modificaciones de objetos (respectivamente). Para los objetos que tienen un período de vida limitado, es posible extender el período de validez con el comando *renew*.

Además de actuar como comando de consulta, el comando *transfer* permite manejar cambios en la esponsorización cliente-objeto. Cuando un cliente desea asumir la esponsorización de un objeto, ejecuta un comando *transfer*, el cual es procesado por el servidor, de manera de completar el proceso de transferencia del objeto. El la Figura 4.2, se puede ver dicho proceso.

Las notificaciones que se envían a los involucrados, pueden ser colocadas en las colas de mensajes asociadas, y luego obtenidas con el comando *poll*, o pueden ser enviadas por algún medio *off-line*. Para el caso de expiración de solicitudes, el protocolo no indica un plazo ni la acción a tomar. Estos son asuntos locales a cada servidor, que deben estar debidamente documentados. Todos los objetos "transferibles", deben tener asociada la respectiva información de autorización, necesaria para realizar los cambios.

4.2.2. Identificación de objetos

Algunos objetos, como los servidores de nombres y contactos, pueden tener utilidad en varios repositorios. Esto puede llevar a inconsistencias que perjudiquen a Internet. Por ejemplo,cambiar un servidor de nombres en un repositorio, pero no en otro, podría llevar a respuestas erróneas sobre las consultas *DNS*. Es por esto, que se define un identificador global único *(GUID)*, asignado cuando un objeto es creado, que debe ser indicado al cliente en cada respuesta entregada, para que éste pueda obtener información detallada sobre el o los objetos involucrados en la respuesta. Este *GUID*, es conocido como *Repository Object Identificator* (*ROID*).

Los valores específicos de los *GUID*, se asignan en base a políticas del repositorio, pero se sugiere que sean construidos según el siguiente algoritmo:

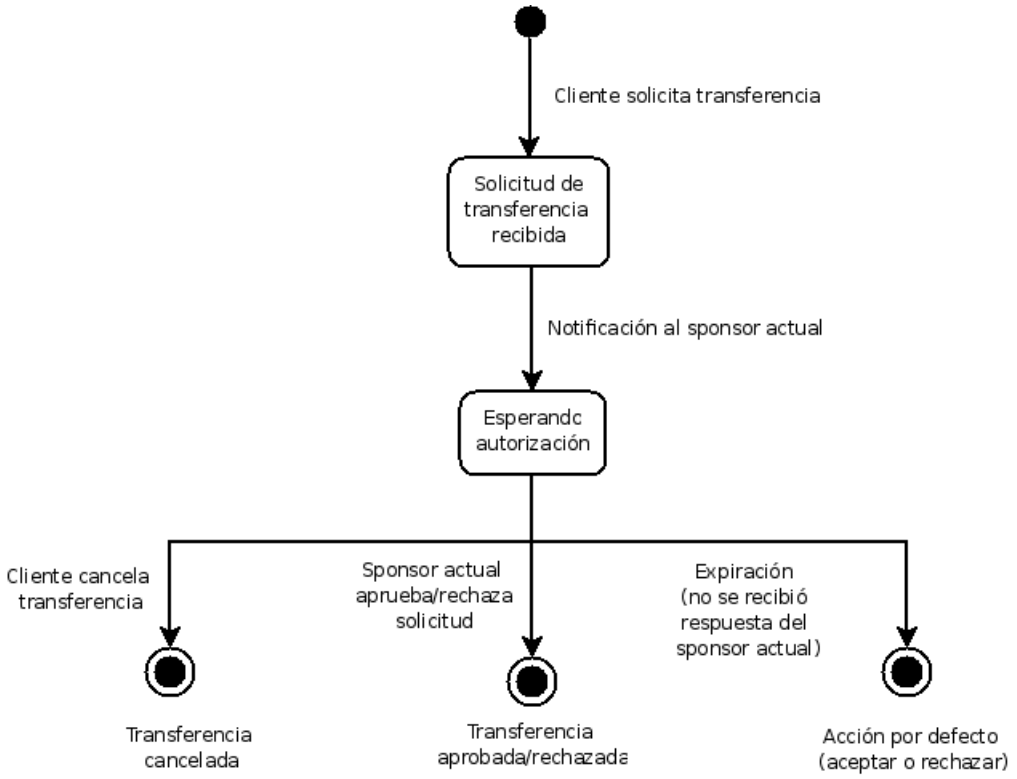


Figura 4.2: Proceso de transferencia de objetos

1. Dividir el universo de los repositorios proveedores de objetos en clases de objetos de repositorio.

2. A cada repositorio dentro de una clase de repositorio se le asigna un identificador mantenido por la *IANA*.

3. Cada repositorio, es responsable, localmente, de asignar identificadores únicos para cada objeto contenido en él.

4. El *GUID*, se forma concatenando el identificador local y el del repositorio, separados por un guión (*ASCII* 45).

Por ejemplo, el servicio ofrecido por SeCIU es identificado por la cadena "uynic", y dentro de SeCIU los dominios se identifican por su *FQDN*. Por lo tanto, el *ROID* del dominio *fing.edu.uy* dentro del repositorio de SeCIU, sería *fing.edu.uy-uynic*.

4.2.3. Mensajes básicos

Los clientes, intercambian información con el servidor a través de mensajes que tienen determinados componentes. Sin entrar en detalles, aquí se describen o enumeran algunos de ellos.

Al iniciar una conexión, o al enviar un mensaje *hello*, que no tiene elementos a destacar, el servidor responde con un *greeting*. Dicho saludo, contiene información descriptiva del repositorio disponible.

Uno de los elementos es el denominado *svcMenu*, que describe los servicios proporcionados por el servidor, indicando versiones soportadas, idiomas soportados, *URI* los espacios de nombres de los objetos manejados, y de las extensiones de objetos soportadas.En este saludo, también se describen las políticas tomadas por el servidor sobre su colección de datos. Los elementos de esa sección, describen el acceso que pueden tener los clientes sobre la información, y afirmaciones sobre la información guardada, que describen el propósito de la colección, los receptores de los datos y la retención de datos. Opcionalmente, se proporciona información sobre el ciclo de vida de la política.

El resto de los mensajes, que permiten a los clientes interactuar con el servidor, se refieren a los comandos y sus respuestas.

Los comandos, contienen un elemento que indica de qué comando *EPP* se trata, el cual puede tener como hijos, entidades definidas por el protocolo o el comando. Luego, se pueden indicar aspectos sobre las extensiones de comandos definidas por el servidor, y, finalmente, un identificador de transacción para poder asociar el cliente y el comando.

Las respuestas contienen elementos que describen el éxito ó la falla del comando (los resultados de éxito o falla no se mezclan). En caso de éxito, se provee un único resultado, mientras que en caso de falla, se proveen varios resultados que permiten documentar las condiciones de falla. En todo caso, se provee un código de respuesta y una descripción de su significado.

El resto de la información permite identificar los objetos involucrados en la operación, para diagnóstico de errores, describe el estado de la cola de mensajes que el servidor tiene para el cliente, en caso de estar respondiendo a un comando *poll* (ver Sección 4.2.1), y puede contener información especifica al comando y al objeto asociado.

4.2.4. Códigos de resultado

Todas las respuestas a comandos *EPP* incluyen un código de resultado que indica el resultado de la ejecución del comando que la originó. Para describir el éxito o fracaso de un comando, *EPP* utiliza un código de 4 dígitos, que se construye de manera similar al utilizado en los comandos de *SMTP*. Siguiendo dicha línea, el primer dígito indica el éxito o el fracaso del comando, mientras que el segundo indica la categoría del resultado. Existen 6 categorías, que se muestran en el Cuadro 4.1.

x0zz	Sintaxis del protocolo
x1zz	Reglas específicas de la implementación
x2zz	Seguridad
x3zz	Manejo de datos
x4zz	Sistema servidor
x5zz	Manejo de conexiones

Cuadro 4.1: Categorías de resultados según código

Los restantes dos dígitos, proveen el detalle de la respuesta, dentro de cada categoría. Además del código, la respuesta debe incluir una descripción legible del estado del resultado. En el Cuadro 4.2 se pueden ver algunos códigos específicos y su descripción.

4.2.5. Consideraciones de seguridad

EPP sólo provee servicios simples de autenticación y autorización de clientes, haciéndolo vulnerable a los ataques más comunes. Cualquier servicio de seguridad más robusto, debe ser provisto por protocolos de capas inferiores. En particular, *EPP* debería ser desplegado sobre una capa de transporte, o

Códigos de éxito	
1000	*Command completed successfully*
1001	*Command completed successfully; action pending*
Códigos de error	
2001	*Command syntax error*
2101	*Unimplemented command*
2200	*Authentication error*
2303	*Object does not exists*
2400	*Command failed*
2500	*Command failed; server closing connection*

Cuadro 4.2: Códigos de resultado y descripciones

protegido con un protocolo de aplicación, que lo dote de confidencialidad, integridad[2] y una autenticación cliente/servidor fuerte.

Para evitar intentos de adivinar contraseñas por repetición, *EPP* podría limitar el número de intentos de *login* fracasados. El mecanismo utilizado por *EPP* para dar el servicio de autenticación, es una variante de *PLAIN SASL* que aumenta o complementa los servicios disponibles en los protocolos de otras capas [27].

Para autorizar transferencias de objetos, *EPP* utiliza información asociada a los objetos, la cual debe ser intercambiada a través de un servicio que provea privacidad e integridad, evitando divulgación y/o modificación, mientras ésta está en tránsito. Además, las instancias de *EPP*, deberían ser protegidas contra ataques de *replay*[3]. Si bien la idempotencia de los comandos de *EPP* garantizaría algo de esto, aún es posible realizar algún ataque de este tipo, por ejemplo, haciendo *replay* de un comando *create* luego de que un comando *delete* tuvo éxito para un determinado objeto.

En la práctica, se utiliza *SSL* para encriptar el canal de comunicación, evitando así que los atacantes puedan aprovechar la información transmitida.

4.3. Extendiendo el *EPP*

EPP fue desarrollado, originalmente, como un estándar de Internet para el registro de nombres de dominio, para uso entre registradores y registros

[2]En seguridad informática, *confidencialidad* significa que la información intercambiada sólo puede ser vista por los involucrados en el negocio, e *integridad* se refiere a que, la información intercambiada, no puede ser modificada en el camino entre emisor y receptor

[3]En seguridad informática, se habla de *replay*, cuando se habla de utilizar información válida vista antes

de dichos objetos. A pesar de esto, el protocolo fue diseñado teniendo en cuenta la posibilidad de extenderlo para ser usado en otros contextos de provisión. Por lo tanto, el protocolo está representado en *XML*, y la especificación del mismo describe funciones generales del protocolo, y no objetos a ser administrados por el protocolo, permitiendo un bajo acoplamiento entre la funcionalidad mínima requerida, y la lógica operacional del manejo de objetos.

Se proveen diversos mecanismos de extensión. Dichos mecanismos, dotan de gran flexibilidad al protocolo, pero, en ningún caso, deben ser aplicados para modificarlo. Una extensión, *agrega* funcionalidades, no *quita* o *modifica* las existentes. Por otro lado, las extensiones en sí mismas, deberían ser extensibles.

Debido a que los mecanismos de extensión de *EPP* pueden llevar a malos diseños, se hace necesario una guía que explique lineamientos para realizar las decisiones de manera ordenada, sin llegar a cometer errores. En esta sección, se presentan el marco de trabajo que permite extender el protocolo, y las guías provistas para hacerlo en buena forma.

4.3.1. Marco de trabajo para las extensiones

Las extensiones se pueden hacer en tres niveles: sobre el protocolo, sobre los objetos o sobre los comandos y sus respuestas. También existe la posibilidad de agregar información de autenticación a los objetos manejados.

Extensiones al protocolo Las extensiones al protocolo, son aquellas que refieren a nuevas funcionalidades agregadas. La sintaxis del *XML* para *EPP*, permite, a través del elemento `<extension>`, agregar dichas funcionalidades. Estas extensiones deben ser definidas cada una por su lado, sin alterar la especificación base del protocolo.

Extensiones a los objetos *EPP* proporciona un marco de trabajo extensible para el manejo de objetos que define las operaciones que pueden realizarse sobre los objetos. Este marco, coloca la definición de las operaciones en el contexto específico de un objeto, lo cual posibilita agregar mapeos para nuevos objetos sin alterar el protocolo básico. Nuevamente, estas extensiones deben ser definidas cada una por su lado, sin alterar la especificación base del protocolo.

Extensiones comando-respuesta Se pueden extender los comandos existentes (y los agregados también), agregando elementos `<extension>` como

hijos de los respectivos elementos `<command>`. Las respuestas, que dependerán del contexto indicado por el comando, se extienden de manera similar.

Extensión de la información de autenticación Algunos objetos, pueden requerir el adjuntado de información de autenticación. Por ejemplo, se puede querer asociar una contraseña a un nombre de dominio en Internet. En caso de necesitarse esto, dicha información deberá ser especificada teniendo en cuenta la posibilidad de múltiples formas de autenticación.

4.3.2. Guías para extender el protocolo

De lo anterior, surge que existen diversas maneras de extender *EPP*, pero, ¿cuál elegir? El aplicarlas de manera indiscriminada, podría llevar a malos diseños, o diseños que requieran modificaciones al protocolo. La *RFC* 3735 [7], proporciona una guía que permite, avanzando en grado de complejidad, escoger un mecanismo de extensión, o descartar *EPP* como la solución buscada. Dicha guía, se resume a continuación.

1. *Extensión comando-respuesta.* Esta es la manera más simple para extender el protocolo, y, por lo tanto, debería ser considerada primero. El usuario deberá contestar la pregunta *¿se puede desarrollar la tarea, extendiendo un mapeo de objeto existente, o modificándolo levemente?* Si la respuesta fuese afirmativa, usar el mecanismo de extensión de *comando-respuesta*, es lo más adecuado.

2. *Extensión de objetos.* Si no existiera ningún mapeo de objetos, cuya modificación sirviera para completar los requerimientos, se debe considerar el mecanismo de extensión de objetos, que implica agregar nuevos objetos. La utilización de este mecanismo, será resultado de una respuesta afirmativa a la pregunta *¿se puede cumplir la tarea utilizando los comandos y estructuras de respuesta de EPP, aplicados a un nuevo objeto?* Ahora, el usuario debería definir un nuevo tipo de objeto, lo cual implica, además, definir cómo actúan los comandos sobre él[4].

3. *Extensión del protocolo.* Si ninguna de las alternativas anteriores fuera de utilidad, se debería considerar extender el protocolo, agregando nuevos comandos, respuestas u otros tipos de estructuras, para ser utilizados con objetos nuevos o preexistentes. Previo a la utilización de

[4]de aquí, que se hable de *"mapeo de objetos"*, ya que los objetos se mapean a los comandos del protocolo

este mecanismo, se deberá contestar afirmativamente si *¿se puede completar la tarea, agregando a EPP nuevos comandos, respuestas u otras estructuras, aplicados a objetos nuevos o existentes?*

Finalmente, si la respuesta a la última pregunta es negativa, se puede afirmar que *EPP no es la respuesta al problema planteado.*

4.4. Utilizando *EPP*

EPP puede ser utilizado en diversos ámbitos, donde sea necesario un contexto de provisión de objetos contenidos en un repositorio central. En esta sección, se presentan a manera introductoria las extensiones existentes para el manejo de nombres de dominio de Internet, analizando la información manejada por el protocolo, a través de los comandos de transformación y consulta (ver Sección 4.2.1), aplicados a contactos, *hosts* y nombres de dominios. El detalle de las extensiones se puede consultar en las *RFC* respectivas (4933[28], 4932[29] y 4931[30]).

Las extensiones planteadas en dichos documentos, hacen referencia a detalles sobre los tipos que pueden tener los atributos de cada objeto. Dicho detalle, está fuera del alcance de esta introducción, pero resulta interesante analizar lo relativo al estado de los objetos.

Los objetos, en general, pueden tomar algunos de los estados que se describen a continuación.

clientDeleteProhibited, serverDeleteProhibited Cualquier pedido para borrar el objeto debe ser ignorado.

clientUpdateProhibited, serverUpdateProhibited Para estos estados, todo intento de actualizar el objeto, salvo cambiar este estado, debe ser rechazado.

ok Este es el estado normal de un objeto que no tiene operaciones pendientes o prohibiciones. Es indicado o borrado por el servidor a medida que otros estados son indicados o borrados.

pendingXX Un comando de transformación XX (*create, update, transfer, renew* o *delete*) fue efectuado sobre el objeto, pero su acción fue postergada por el servidor. La postergación de ciertas acciones, es asunto del servidor, y puede deberse a diversas acciones, como la de revisiones humanas o acciones de terceros. En el caso de los *hosts*, un estado pendiente, también, puede deberse a una transferencia del nombre de dominio que lo subordina.

Los objetos de tipo *host* y *contacto* también pueden tener asociado el estado *linked*, el cual indica si el objeto tiene al menos una relación activa con algún otro objeto, como un nombre de dominio. Por otro lado, *hosts* y *dominios*, pueden tener asociado los estados *clientTransferProhibited* y *serverTransferProhibited*, en cuyo caso, el objeto así marcado, no puede ser transferido, ya sea por el cliente o por el servidor, respectivamente.

Finalmente, los objetos de tipo *nombre de dominio*, pueden ser etiquetados con alguno de los siguientes estados:

clientHold, serverHold La información de delegación del servicio *DNS*, no debe ser publicada para el objeto.

clientRenewProhibited, serverRenewProhibited Las solicitudes para renovar la validez del objeto deben ser rechazadas.

inactive Aún no se ha asociado información de delegación al objeto.

Ciertas combinaciones de estados, no están permitidas:

- *ok* solo se puede mezclar con *linked* para hosts y contactos. Para nombres de dominio, no se puede combinar con ningún otro estado.
- *pendingDelete* no se puede combinar ni con *clientDeleteProhibited*, ni con *serverRenewProhibited*.
- *pendingTransfer* no se puede combinar ni con *clientTransferProhibited*, ni con *serverTransferProhibited* (contactos y nombres de dominio).
- *pendingUpdate* no se puede combinar ni con *clientUpdateProhibited*, ni con *serverUpdateProhibited*.
- *linked* se puede combinar con cualquier estado.
- *pendingRenew* no se puede combinar ni con *clientRenewProhibited*, ni con *serverRenewProhibited*.

Por último, los estados *pendingCreate*, *pendingDelete*, *pendingRenew*, *pendingTransfer*, y *pendingUpdate*, no deben ser combinados entre sí.

4.4.1. Contactos

Los *contactos* representan a personas asociadas a los dominios provistos por el repositorio central. Un objeto de este tipo, por lo tanto, contendrá información relativa a personas, como ser información postal y social sobre dicha persona. Ejemplo de esta información, son el nombre del contacto, su teléfono, su correo electrónico, la organización a la cual pertenece, y su dirección. La dirección, se desglosa en sus diversos componentes: calle y número, ciudad, país, código postal, etc.

Opcionalmente, se puede indicar a los clientes qué información requiere procesamiento especial del servidor, antes de ser divulgada a terceros.Esta indicación, se realiza a través del elemento `<contact:disclose>`, indicando si se prefiere permitir o denegar la divulgación de un determinado atributo, y si se trata de la versión local o internacionalizada del mismo. Todas las solicitudes en este sentido, deben ir en acuerdo con las políticas de la colección de datos del servidor, anunciadas en el *greeting* (ver Sección 4.2.3).

Este tipo de objeto, tiene la propiedad de ser transferible entre los clientes de un repositorio, y, por lo tanto, le es aplicable el comando *transfer* (ver Sección 4.2). Cuando un cliente desee transferir un contacto, deberá proporcionar su identificador y la información de autenticación necesaria, y podrá, luego, consultar los mensajes originados por el servidor durante el proceso de transferencia, ya sea con el mismo comando *transfer*, o con el comando *poll*.

4.4.2. *Hosts*

Los *hosts* son objetos que existen para la delegación de dominios, y, por lo tanto, contienen la información necesaria a los efectos de delegación, esto es, nombre y dirección *IP*. Existen, además, otros datos que permiten trazar el ciclo de vida del objeto, a través de fechas e identificación de los clientes que intervinieron en dicho ciclo de vida.

Un dato importante sobre los objetos de este tipo, es el estado, el cual ya fue descrito con detalle anteriormente, en esta misma sección. También es importante destacar que se almacena quién es el cliente patrocinador del objeto, o sea, quién lo puede administrar o transformar.

Existen varios tipos de *hosts*: internos/externos y los que están subordinados, o no, a algún dominio. Un *host* es externo o interno relativo al repositorio en que se encuentra almacenado, mientras que un *host* estará subordinado o no, si es usado para la delegación de algún dominio. Pueden existir *hosts* que no estén subordinados a ningún dominio, y pueden haber dominios, dentro del repositorio, que utilicen *hosts* externos, es decir, en otro repositorio.

4.4.3. Nombres de dominio de Internet

Un *nombre de dominio de Internet* tiene asociado su *FQDN*, los *hosts* a los que se delegan los servicios de resolución del dominio, diversos contactos, el cliente patrocinador (un *registry EPP*), e información sobre su período de validez. Los *hosts* asociados, deben existir previo al registro del nombre, y se pueden especificar con el *FQDN* de un *host* registrado, o indicando éste y la dirección *IP* asociada.

Un nombre de dominio, puede tener asociados contactos de varios tipos (técnicos, administrativos, de pago, etc.) y tiene asociado un contacto para representar a al persona que registra el nombre (el titular de dicho nombre). El período de validez indica un intervalo de tiempo durante el cual el nombre es válido, y este puede ser renovado con el comando *renew*.

Los nombres de dominio, pueden ser transferidos con el comando *transfer*. Cuando un nombre de dominio es transferido, se puede extender su período de validez, y todos los objetos de tipo *host* subordinados, deben ser incluidos como parte del proceso de transferencia.

DNS habilita la definición de diversos tipos de registros asociados a un dominio, pero este mapeo sólo define los necesarios para la delegación de servicios y resolución, es decir *NS*, *A* y *AAAA*. Funcionalidades de otros tipos de registros *DNS*, como por ejemplo *MX* y *CNAME*, podrían ser desarrolladas a través de la extensión de *EPP*.

Hasta aquí se presentaron las descripciones de los objetos involucrados en el negocio de registro de nombres de dominio de Internet. En la Figura 4.3, se puede apreciar un diagrama que muestra cómo se asocian entre sí.

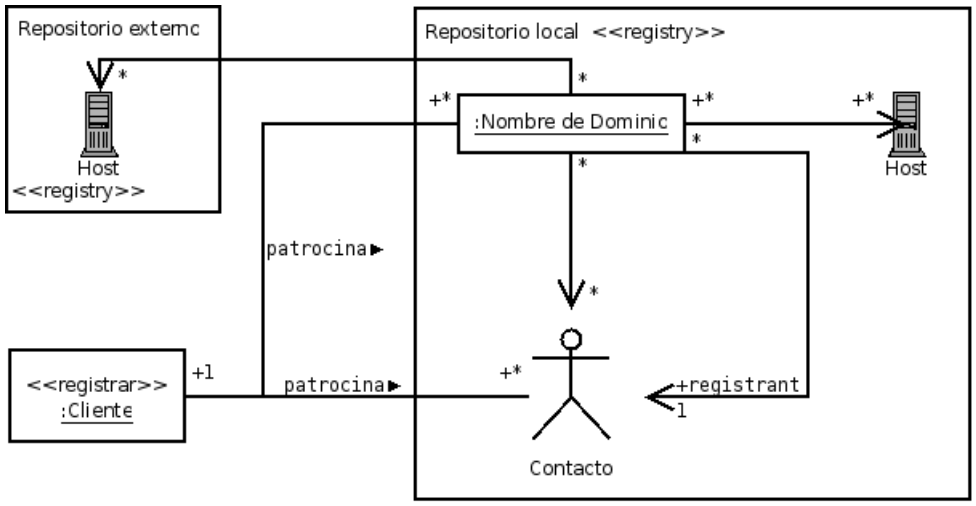


Figura 4.3: *EPP* aplicado a la provisión de Nombres de Dominio de Internet

4.5. Resumen del capítulo

EPP es un protocolo diseñado para comunicar un registro (*registry*) de objetos, con los interesados en registrar (almacenar) objetos en este repositorio (*registrars*). Está basado en *XML*, y su forma extensible lo hace apto para almacenar diversos tipos de objetos. Los objetos almacenados pueden ser manejados y consultados a través de comandos de transformación (como alta, baja y modificación) y consulta.

Los mecanismos de extensión, permiten adaptar el protocolo genérico (ver [1]) a las necesidades de almacenamiento de objetos del interesado. Existen 3 formas de realizar extensiones (ver [7]). La primera es la extensión de comandos y respuestas, que consiste en modificar levemente objetos existentes y extender el mapeo de los comandos a estos objetos, para adaptarlos al uso requerido. En segunda instancia, se puede recurrir a las extensiones de objetos, que consiste en agregar nuevos objetos y definir cómo actúan los comandos de *EPP* sobre los mismos. Por último, se puede extender el protocolo agregando nuevos comandos. Si ninguno de estos procedimientos ayuda a resolver el problema, se puede afirmar que *EPP* no es una solución para el mismo.

El uso principal de *EPP*, y principal razón para su desarrollo, es el almacenamiento y gestión de nombres de dominios de Internet, para su posterior publicación en el Sistema de Nombres de Dominio (*DNS*). A tales efectos, existen extensiones para el mapeo del protocolo a los nombres de dominio, a los *hosts* (que cumplen funciones de servidores de nombres para los dominios), y a los contactos, que son las personas relacionadas al dominio, como puede ser un dueño, un contacto para gestiones administrativas, etc.

Capítulo 5

Extensiones existentes de EPP

5.1. Introducción

EPP posee diversos mecanismos de extensión que permiten adaptar el protocolo a las necesidades específicas de los agentes interesados en brindar un registro de objetos centralizados utilizando dicho protocolo. Dichos mecanismos, abarcan la extensión de las interacciones comando respuesta, las extensiones a comandos, y las extensiones al propio protocolo. El primero, agrega elementos a las solicitudes del cliente y a las respuestas del servidor, sobre los comandos existentes en el protocolo, aplicados a los objetos ya definidos. La segunda posibilidad, implica agregar nuevos objetos al protocolo y definir como actúan los comandos existentes sobre dichos objetos. La última posibilidad a considerar, es la de extender el protocolo propiamente dicho, agregando nuevos comandos.

En las siguientes secciones, se presentan los resultados de una investigación basada en Internet sobre extensiones existentes al protocolo *EPP* y sus mapeos a los objetos implicados en el negocio del registro de nombres de dominio. La misma, se realizó utilizando el buscador de Google, y contó con asesoramiento de los especialistas de SeCIU en el asunto.

5.2. Extensiones para el dominio .COOP

El *TLD .COOP*, es un dominio restringido destinado a organizaciones cooperativas[1]. Como tal, cada nombre registrado bajo éste, debe sufrir un proceso de verificación para asegurar que el nombre está realmente asociado a una organización de este tipo. A tales efectos, se introduce el concepto de *registrant*, el cual no es más que un contacto de *EPP* que ha sido verificado como perteneciente a una organización cooperativa. Cualquier contacto verificado como *registrant* del dominio *.COOP*, puede registrar nombres dentro de éste. En la figura 5.1, puede verse el proceso seguido para comprobar la elegibilidad de un contacto como *registrant*.

El proceso de verificación, se inicia cuando un contacto es utilizado como *registrant* por primera vez. Dentro del mismo, se distinguen dos entidades participantes: el socio de verificación (*VP* por sus siglas en Inglés) y el sponsor. El *VP* es una persona u organización externa que acuerda ayudar a *dotCoop*[2] para confirmar la elegibilidad de un *registrant*. Un *sponsor*, es otra cooperativa, u organización de cooperativas, que puede proveer información acerca de la elegibilidad del patrocinado.

[1] http://www.nic.coop

[2] Organización encargada de administrar el *TLD .COOP*

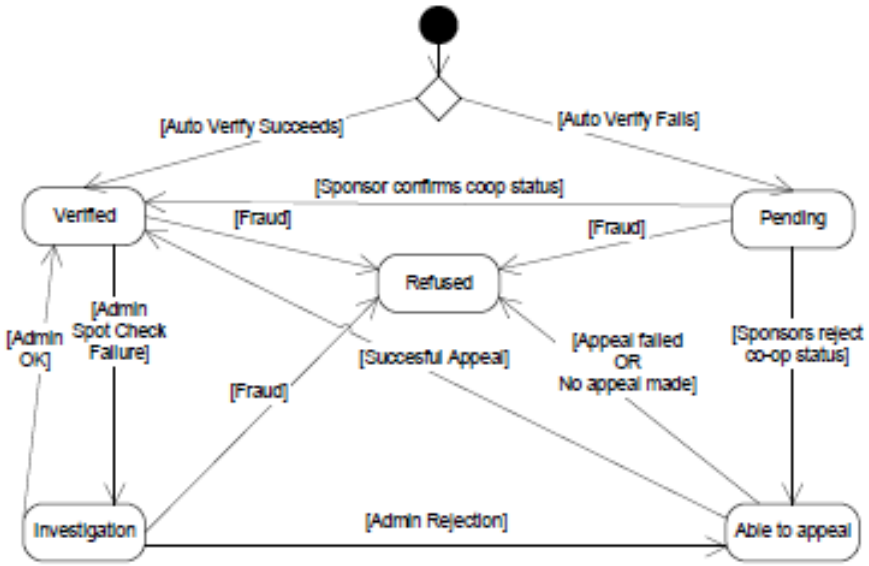


Figura 5.1: Proceso de verificación para *registrants* del TLD .COOP

Cuando un *registrant* es confirmado como elegible, su estado es *verified*. Si el *registrant* no puede ser verificado automáticamente, pasa a un estado pendiente, donde puede ser aceptado (pasaría a verificado) o rechazado. Si es rechazado, tiene un plazo para apelar. Si las apelaciones en plazo no tienen efecto, o si el plazo se cumple, se rechaza la solicitud.

Algunos *registrants* verificados, pueden ser elegidos para ser auditados. Cuando un *registrant* es auditado, puede resultar que se confirme su elegibilidad o que sea rechazado.

Los dominios registrados por un *registrant* que no está verificado, no pueden ser activados para el sistema de *DNS*.

Para soportar este proceso, se extiende el contacto *EPP* agregando una lista de patrocinadores, un estado de verificación, una preferencia de lenguaje, y una preferencia de lista de correo. Los patrocinadores son contactos, pero, como en el *EPP* estándar sólo se pueden relacionar contactos con dominios, es necesaria esta extensión para poder relacionar al contacto patrocinado con sus sponsor. La preferencia de lenguaje, es para permitir a los registros, comunicarse con los contactos en su lenguaje, y la lista de correo sirve para que el registro envíe información a los *registrants* [31].

5.3. Extensiones para *IDN*

IDN (*Internationalized Domain Name*, o Nombre de Dominio Internacionalizado), es un nombre de dominio que puede tener caracteres fuera del conjunto *ASCII*. Ejemplo de éstos, son los nombres en idiomas con lenguajes ideográficos, como el Chino o el Japonés, o los nombres en lenguajes como el Francés o el Español, que pueden tener acentos, y, en el caso del Español, la letra eñe.

Debido a que los sistemas *DNS* fueron diseñados basándose en caracteres *ASCII*, se hace necesaria una manera de poder soportar nombres *no-ASCII*, basándose exclusivamente en caracteres *ASCII*. Esta manera define una codificación para los nombres *no-ASCII*, y dos operaciones para realizar las traducciones en uno y otro sentido [32].

Las aplicaciones *IDN* deben soportar los siguientes estándares: *Nameprep* [33], *Stringprep* [34], y *Punycode* [35]. Los dos primeros, indican la manera en que se deben preparar las cadenas de caracteres para las transferencias de nombres *IDN*, y el último, define cómo se codifican dichos nombres. La codificación implica la utilización de un prefijo conocido como *ACE* (*ASCII Compatible Encoding*, o Codificación Compatible con *ASCII*), el cual marca que una etiqueta está internacionalizada. Este prefijo, que es la cadena `xn--`, no puede ser utilizado como parte del nombre de ningún dominio. Además, la codificación, implica guiones en ciertas posiciones de la cadena de caracteres, lo cual hace que tampoco se puedan usar estos caracteres en dichas posiciones del nombre.

La utilización de nombres *IDN*, introduce varias dificultades. La más importante de todas, es el tamaño del repertorio de caracteres posibles, lo cual hace que la tarea de codificación-decodificación no sea sencilla. Esta dificultad, se ataca restringiendo el conjunto de caracteres que soporta un registro, lo cual obliga a que el registro indique el conjunto de caracteres soportados.

Otra dificultad, más sutil, es la de los caracteres confundibles: caracteres de diferentes alfabetos, que a la vista son muy similares, pero que significan distintas cosas.

Además, surgen problemas cuando se tratan nombres equivalentes. Por ejemplo: ¿*república* es lo mismo que *republica*?, ¿qué se debe hacer con los nombres equivalentes que se utilizan cuando no está disponible *IDN* (como peña, penha, penia, etc.)? Es necesario incluir políticas para dirimir estas cuestiones.

En cuanto al manejo de los nombres por parte del sistema *DNS*, no existen problemas: a nivel de dicho protocolo, se manejan todos los nombres con la codificación *punycode*, que no incluye caracteres *no-ASCII* y puede ser manejada por los servidores de nombres. Se deja todo el trabajo de

codificación-decodificación a las aplicaciones interesadas en resolver nombres. Por lo tanto, los *registries*, antes de generar los archivos de zona, deben asegurarse de realizar éste trabajo. Para esto, se pueden tomar dos enfoques: que el *registrar* envíe el nombre en formato *punycode*, o que lo envíe en formato *IDN* y que el registro se encargue de la traducción. Si el enfoque es el primero, el *registry* permanece sin cambios, mientras que si es el segundo, es necesario agregar elementos que posibiliten no sólo la traducción, si no, también, el manejo de errores que pueden ocurrir al intentar traducir caracteres no soportados.

Por estos motivos, no sería necesario modificar el protocolo *EPP*, salvo que se quiera agregar el soporte en el registro. Si fuera esta la opción, sería necesario agregar extensiones para indicar la lista de idiomas y caracteres soportados, y, eventualmente, las políticas que se toman para atacar las dificultades antes expuestas. En [36] se define una extensión a *EPP* para indicar, mediante una extensión a los nombres de dominio que muestra, mediante una etiqueta, el idioma usado en el registro. En [37], se indican los procedimientos para registros *IDN* en el *TLD* .PL., asociado a Polonia, cuyo alfabeto incluye caracteres *no-ASCII*. Allí se puede ver cómo se definen los caracteres permitidos, y cómo se permite el registro *IDN* sin modificar el *EPP* estándar.

5.4. Extensiones para *DNSSEC*

SeCIU tiene planes de agregar soporte para las extensiones de seguridad *DNSSEC* a su servicio de *DNS*. Como fue explicado en la Sección 2.3, *DNSSEC* agrega cuatro nuevos tipos de registros de recursos a *DNS*. De ellos, los registros *DNSKEY*, *RRSIG* y *NSEC*, son agregados al archivo de zona durante el proceso de firmado, por lo que no se requiere ninguna interacción con *EPP*. Sin embargo, no ocurre lo mismo con el registro *DS*. Es necesario que ante un cambio de clave en un dominio firmado, se notifique al *registry* actualizando el *digest* contenido en el registro *DS*. Para realizar dicha actualización, se utiliza *EPP*.

Por lo tanto, la única interacción entre *DNSSEC* y *EPP*, es que el primero utiliza al segundo como mecanismo de gestión de los registros de tipo *DS*, para que éstos puedan ser publicados desde el repositorio central, lo cual puede verse en la figura 5.2.

En la *RFC* 4310 [38] se plantean las extensiones necesarias a los comandos *info*, *create* y *update* de *EPP* para dar soporte para que el *registry* pueda gestionar y publicar los registros *DS*, necesarios para el correcto funcionamiento de *DNSSEC*. Además, se contemplan los casos de actualizaciones de los registros *DS* debido a cambios de claves *KSK*, tanto planificados como

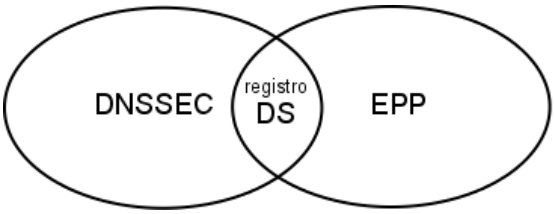


Figura 5.2: Relación entre *DNSSEC*, *EPP* y registros DS

casos de emergencia.

A continuación, se presenta un análisis de las extensiones definidas en la *RFC* mencionada.

Extensión de la respuesta del comando *info* aplicado a Dominios Esta extensión consiste en que la respuesta del comando *info* contenga la información de delegación del dominio, la cual fue aportada por el *registrar* en una invocación previa de un comando *create* o *update*.

Luego de la Sección `<resData>` de *EPP*, la cual contiene la información del dominio, se agrega la Sección `<secDNS:infData>` la cual contendrá una o más etiquetas `<secDNS:dsData>`, cada una de las cuales describe la información de delegación proporcionada. Cada uno de los elementos `<secDNS:dsData>` contendrá los nodos listados a continuación.

`<secDNS:keyTag>` Este elemento contiene el valor de *tag* de la clave utilizada para generar el *digest*.

`<secDNS:alg>` Contiene el código numérico asignado al algoritmo utilizado para generar las claves criptográficas.

`<secDNS:digestType>` Contiene el código numérico que se corresponde con el algoritmo utilizado para generar el *digest*.

`<secDNS:digest>` Contiene el *digest* de la clave *DNSKEY*.

`<secDNS:maxSigLife>` Este nodo es opcional, en caso de estar presente indica una preferencia para el número de segundos de expiración a partir de la generación de la firma del registro *DS*. En caso de existir varios registros `<secDNS:dsData>` para un dominio, los valores de `<secDNS:maxSigLife>` deberían ser iguales.

`<secDNS:keyData>` Este nodo también es opcional, en caso de estar presente indica los valores de entradas de datos utilizados para el cálculo del digesto. Contiene:

- `<secDNS:flags>`: banderas que indican el tipo de clave.
- `<secDNS:protocol>`: indica el código del protocolo utilizado.
- `<secDNS:alg>`: número de algoritmo utilizado para la clave.
- `<secDNS:pubKey>`: clave pública utilizada.

Los detalles de los valores permitidos para cada uno de estos nodos se encuentran en la *RFC* 4034 [10].

En caso de cualquier tipo de error al obtener esta información, se debe retornar una respuesta de error *EPP*.

En los Cuadros 5.1 y 5.2 se muestran ejemplos de respuestas exitosas al comando *info*, cuando un dominio tiene asociada información de seguridad. El segundo ejemplo, contiene los nodos opcionales. Cada ejemplo, asume la parte estándar de *EPP*, mostrando únicamente la información agregada.

```
<extension>
  <secDNS:infData
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:secDNS-1.0
    secDNS-1.0.xsd">
    <secDNS:dsData>
      <secDNS:keyTag>12345</secDNS:keyTag>
      <secDNS:alg>3</secDNS:alg>
      <secDNS:digestType>1</secDNS:digestType>
      <secDNS:digest>49FD46E6C4B45C55D4AC</secDNS:digest>
    </secDNS:dsData>
  </secDNS:infData>
</extension>
```

Cuadro 5.1: Respuesta al comando *info* para un nombre con información *DNSSEC*

Extensión del comando *create* aplicado a Dominios Bajo el nodo `<extension>` se agrega un nodo `<secDNS:create>` el cual identifica la extensión. Este a su vez contendrá uno o más nodos `<secDNS:dsData>`, los cuales tienen la información de delegación, y ya están detallados en la sección anterior para el comando *info* (ver Sección 5.4).

El servidor deberá responder con un código de error *EPP* en caso que los valores recibidos no sean apropiados, ya sea por un problema sintáctico como por razones políticas del *registry*.

```
<extension>
  <secDNS:infData
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:
    secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:dsData>
      <secDNS:keyTag>12345</secDNS:keyTag>
      <secDNS:alg>3</secDNS:alg>
      <secDNS:digestType>1</secDNS:digestType>
      <secDNS:digest>49FD46E6C4B45C55D4AC</secDNS:digest>
      <secDNS:maxSigLife>604800</secDNS:maxSigLife>
      <secDNS:keyData>
        <secDNS:flags>256</secDNS:flags>
        <secDNS:protocol>3</secDNS:protocol>
        <secDNS:alg>1</secDNS:alg>
        <secDNS:pubKey>AQPJ////4Q==</secDNS:pubKey>
      </secDNS:keyData>
    </secDNS:dsData>
  </secDNS:infData>
</extension>
```

Cuadro 5.2: Respuesta al comando *info* para un nombre con información *DNSSEC* y nodos opcionales.

Los Cuadros 5.3 y 5.4 muestran la información que se agrega al comando *create* para satisfacer los requerimientos de *DNSSEC*.

```
<extension>
  <secDNS:create
      xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
      xsi:schemaLocation="urn:ietf:params:xml:ns:
      secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:dsData>
      <secDNS:keyTag>12345</secDNS:keyTag>
      <secDNS:alg>3</secDNS:alg>
      <secDNS:digestType>1</secDNS:digestType>
      <secDNS:digest>
        49FD46E6C4B45C55D4AC
      </secDNS:digest>
    </secDNS:dsData>
  </secDNS:create>
</extension>
```

Cuadro 5.3: Comando *create* para un nombre con información *DNSSEC*

Extensión del comando *update* aplicado a Dominios Para satisfacer los requerimientos de información de *DNSSEC* en el *registry*, bajo el nodo `<extension>`, deberá existir un nodo `<secDNS:update>` el cual identifica la extensión. Contendrá 3 secciones:

`<secDNS:add>` Especifica información a agregar. Contiene uno o más nodos `<secDNS:dsData>`.

`<secDNS:rem>` Especifica información a quitar. Contiene uno o más nodos `<secDNS:keyTag>` que son usados para identificar la información a quitar.

`<secDNS:chg>` Especifica información a cambiar. Contiene uno o más elementos `<secDNS:dsData>` con los que se sobreescribirá la información existente para el dominio.

Además, el nodo `<secDNS:update>` podrá contener el atributo *urgent* como mecanismo para solicitar, al *registry*, el procesamiento prioritario de su solicitud. Si esta actualización no puede ser completada con alta prioridad, se deberá retornar el código de error *EPP* 2306, *parameter value policy error*.

```
<extension>
  <secDNS:create
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:
    secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:dsData>
      <secDNS:keyTag>12345</secDNS:keyTag>
      <secDNS:alg>3</secDNS:alg>
      <secDNS:digestType>1</secDNS:digestType>
      <secDNS:digest>
        49FD46E6C4B45C55D4AC
      </secDNS:digest>
      <secDNS:maxSigLife>604800</secDNS:maxSigLife>
      <secDNS:keyData>
        <secDNS:flags>256</secDNS:flags>
        <secDNS:protocol>3</secDNS:protocol>
        <secDNS:alg>1</secDNS:alg>
        <secDNS:pubKey>AQPJ////4Q==</secDNS:pubKey>
      </secDNS:keyData>
    </secDNS:dsData>
  </secDNS:create>
</extension>
```

Cuadro 5.4: Comando *create* para un nombre con información *DNSSEC* y nodos opcionales

Los Cuadros 5.5, 5.6, 5.6, 5.7 y 5.8, muestran la información extendida para diversas actualizaciones, que incluyen información de seguridad. El primero, muestra un ejemplo de invocación al comando *update* para agregar información de delegación, mientras que, en el segundo, dicha información está siendo quitada. En el tercer Cuadro, se está realizando una modificación urgente, tal como lo muestra el atributo *urgent* con valor 1 dentro del nodo `<secDNS:update>`. Finalmente, en el Cuadro 5.8 se muestra un ejemplo de una actualización incluyendo la información opcional.

```
<extension>
  <secDNS:update
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:
    secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:add>
      <secDNS:dsData>
        <secDNS:keyTag>12346</secDNS:keyTag>
        <secDNS:alg>3</secDNS:alg>
        <secDNS:digestType>1</secDNS:digestType>
        <secDNS:digest>
          38EC35D5B3A34B44C39B
        </secDNS:digest>
      </secDNS:dsData>
    </secDNS:add>
  </secDNS:update>
</extension>
```

Cuadro 5.5: Extensión al comando *update* para agregar información *DNSSEC*

5.5. Extensiones para el dominio .PL

Los nombres de dominio polacos, son registrados dentro del *TLD* .PL. Las extensiones más interesantes para este dominio, son las que se refieren a la reserva de nombres de dominio, para lo cual, se extiende el comando *create* aplicado al objeto dominio, agregando el elemento `<book>`, y la creación del objeto *future*. También resulta interesante la creación del comando *report* para obtener diversas listas relativas a los objetos registrados.

El elemento `<book>`, fue agregado al comando *create* estándar para indicar que el nombre de dominio involucrado en dicho comando, va a ser reservado,

```
<extension>
  <secDNS:update
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:
    secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:rem>
      <secDNS:keyTag>12345</secDNS:keyTag>
    </secDNS:rem>
  </secDNS:update>
</extension>
```

Cuadro 5.6: Extensión al comando *update* para quitar información *DNSSEC*

```
<extension>
  <secDNS:update urgent="1"
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:
    secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:chg>
      <secDNS:dsData>
        <secDNS:keyTag>12345</secDNS:keyTag>
        <secDNS:alg>3</secDNS:alg>
        <secDNS:digestType>1</secDNS:digestType>
        <secDNS:digest>
          49FD46E6C4B45C55D4AC
        </secDNS:digest>
      </secDNS:dsData>
    </secDNS:chg>
  </secDNS:update>
</extension>
```

Cuadro 5.7: Extensión al comando *update* para modificar información *DNSSEC* de manera urgente.

```
<extension>
  <secDNS:update
    xmlns:secDNS="urn:ietf:params:xml:ns:secDNS-1.0"
    xsi:schemaLocation="urn:ietf:params:xml:ns:
    secDNS-1.0 secDNS-1.0.xsd">
    <secDNS:chg>
      <secDNS:dsData>
        <secDNS:keyTag>12345</secDNS:keyTag>
        <secDNS:alg>3</secDNS:alg>
        <secDNS:digestType>1</secDNS:digestType>
        <secDNS:digest>
          49FD46E6C4B45C55D4AC
        </secDNS:digest>
        <secDNS:maxSigLife>604800</secDNS:maxSigLife>
        <secDNS:keyData>
          <secDNS:flags>256</secDNS:flags>
          <secDNS:protocol>3</secDNS:protocol>
          <secDNS:alg>1</secDNS:alg>
          <secDNS:pubKey>AQPJ////4Q==</secDNS:pubKey>
        </secDNS:keyData>
      </secDNS:dsData>
    </secDNS:chg>
  </secDNS:update>
</extension>
```

Cuadro 5.8: Extensión al comando *update* para modificar información *DNSSEC* de manera urgente con información opcional.

mas no registrado. Cuando se omite dicho elemento, el dominio es registrado de manera normal, mientras que en su presencia, el nombre es reservado por un determinado período de tiempo. Durante este período, la única acción posible es el registro por parte del cliente que lo reservó. Una vez transcurrido el tiempo de reserva, si no se ha confirmado, el registro del nombre en cuestión es bloqueado por otro período de tiempo, luego del cual el nombre queda disponible para reserva o registro. Es relevante notar que no es obligatoria la reserva de un nombre previo a su registro: el paso de reserva es opcional.

El fin de la creación del objeto *future* es similar a la extensión `<book>`, en cuanto a que se refieren a la reserva de nombres. La diferencia, es que el elemento `<book>` es utilizado para reservar nombres que no están registrados, mientras que, el objeto *future* se utiliza para manifestar el interés de un cliente en registrar un nombre de dominio ya registrado, cuando el período de expiración del mismo se cumpla sin que exista intención, del propietario del nombre, de renovarlo. Una vez culminada la validez de un dominio, al no haber sido renovado, el sistema chequea la existencia de un objeto *future* con dicho nombre (el del dominio expirado). En caso de no existir tal objeto, el dominio queda, nuevamente, disponible para su reserva o registro. Si existiera un *future* para el dominio, automáticamente se crea una reserva para dicho nombre (con la extensión `<book>`) asociada al cliente que creó el *future*. Una vez creada la reserva, el objeto es eliminado. El objeto *future*, puede ser manejado utilizando los comandos de *EPP* [39].

Para obtener listados sobre los objetos registrados en el repositorio, se introduce el comando *report*. Para su utilización, el cliente debe indicar sobre qué tipo de objeto desea realizar la consulta (dominio, contacto, host, o future), utilizando, en cada caso, un elemento adecuado (contenido dentro del comando) con los datos para el criterio de búsqueda. El sistema, responderá con una lista de los objetos pertenecientes al usuario, que cumplan con el criterio de búsqueda [40].

También, se dispone de una extensión que permite, al cliente, ocultar determinada información a los sistemas de *WHOIS*, que permiten hacer consultas sobre la información almacenada en los sistemas de registros de nombres (*consentForPublishing*), y otra que sirve para identificar un determinado contacto como una persona física, en contraposición a organizaciones (*individual*).

5.6. Extensiones para el dominio .BR

El registro de dominios en Brasil, realiza una extensión al mapeo de contactos de *EPP*. Dicha extensión se utiliza para representar a organizaciones

que registran dominios, como podría ser, por ejemplo, un banco o cualquier empresa [41]. Esto, a diferencia del contacto *EPP*, que representa personas individuales, ya sean físicas o jurídicas, diluyéndose el significado de organización. Puede verse a las organizaciones, entendidas en este contexto, como un grupo de contactos "normales" que comparten determinada información social. A su vez, las organizaciones que no dispongan de oficinas locales (en Brasil), pueden recurrir a una organización apoderada (*proxy*) que los represente. La definición de este tipo de organizaciones, se realiza *off-line*.

Más interesantes resultan las extensiones relativas al registro de nombres de dominio. Se define un *ticket*, que es un identificador secuencial asociado al registro de un Nombre de Dominio, y que garantiza el principio de *Primero en Llegar, Primero en ser Servido*, o *FCFS* (*First Come, First Served*), en las solicitudes sobre el nombre.

Luego del borrado o expiración de un nombre de dominio, se lo somete a un período de liberación durante el cual, los interesados en dicho nombre, pueden pedir su registro. Si hay suficientes solicitudes, en orden por el *ticket* suministrado, se elige a quien tenga derecho sobre el nombre y se le asigna. En caso contrario, el nombre puede pasar al espacio disponible de nombres o a una lista de nombres a ser ofrecidos en un próximo período de liberación [42].

Los pedidos de registro de nombres, pueden tener su estado en *pendingCreate*, lo cual significa que el registro del nombre está latente, en espera de procedimientos fuera de línea, ya sea por parte del registrante, el registrador (o cliente patrocinador) o el registro. Se diferencian tres tipos diferentes de esperas:

Documentation Pending. Dependiendo de la categoría del Nombre de Dominio, puede ser necesario que el registrante tenga que presentar algún documento para probar que puede utilizar dicho nombre. En este caso, se asocia un *doc pending* y la lista de los documentos correspondientes.

DNS Pending. Todo dominio debe tener un conjunto de servidores de nombre que respondan de manera autoritativa. Si dichos servidores no pueden responder por el dominio, se asocia este *pendiente* hasta que el problema se resuelva.

Release Proccess Pending. Este tipo de *pendiente*, indica el final y el resultado del proceso de liberación.

Un punto a destacar, es el algoritmo de equivalencia, el cual es muy sencillo. Primero, se define que los caracteres acentuados son equivalentes a su versión sin acentuar, y que la ce con cedilla (c es equivalente a ç y C a Ç),

y luego, se define que dos nombres de dominio son equivalentes, si de la aplicación del algoritmo sobre éstos se obtiene el mismo resultado. Desafortunadamente, la documentación consultada, no define ninguna política sobre nombres equivalentes. En un área relacionada, el registro .BR ofrece la posibilidad de nombres IDN, para lo cual define una tabla de caracteres válidos [43].

5.7. Extensiones para brindar un período de gracia

Durante 2002, la *ICANN*, desarrolló una propuesta para dotar a los nombres de dominio *DNS*, de un período de gracia, durante el cual las acciones tomadas sobre dicho nombre en un contexto de registro central compartido, tal como el que busca ser *EPP*, pudieran ser revertidas o revocadas. En particular, se desarrolló el denominado *Redemption Grace Period* (*RGP*, Período de Gracia para Redención), que permite revisar el borrado de nombres del registro central, antes de que estos sean eliminados definitivamente del repositorio [44]. En la figura 5.3, puede verse un diagrama de estados, donde se muestra el ciclo de un nombre al cual se le aplica un comando de borrado, y el proceso seguido durante el *RGP*.

Al principio del proceso, un dominio está en el estado *ok* de *EPP*, o cualquier otro que permita el procesamiento de un comando *delete* de *EPP*. El *RGP* comienza una vez que se recibe dicho comando. Si se desea restaurar el dominio, se debe enviar una orden *restore* utilizando una versión extendida del comando *update*. Cuando se procesa exitosamente dicho pedido, el *registry* espera que el *registrar* envíe un reporte de restauración. Si el reporte es satisfactorio, el dominio pasa nuevamente al estado inicial. Si no se recibe el reporte, se vuelve a esperar un *restore*. Cuando pasa un determinado tiempo sin recibir el *restore* , el dominio pasa a esperar su eliminación, la cual es realizada una vez que se termina el período durante el cual el dominio puede permanecer en estado *pendingDelete*. Finalizado el borrado, el dominio queda disponible para ser registrado nuevamente.

Como se menciona en el párrafo anterior, existe una extensión a *EPP* a través del comando *update*, pero ésta no es la única. Otras extensiones destinadas a dar soporte a este requerimiento, son el agregado de un estado de *RGP* al objeto dominio, y la correspondiente extensión para revisar el estado *RGP* de un dominio, así como la modificación del comando *update* para soportar, además de las solicitudes de restauración, los correspondientes reportes.

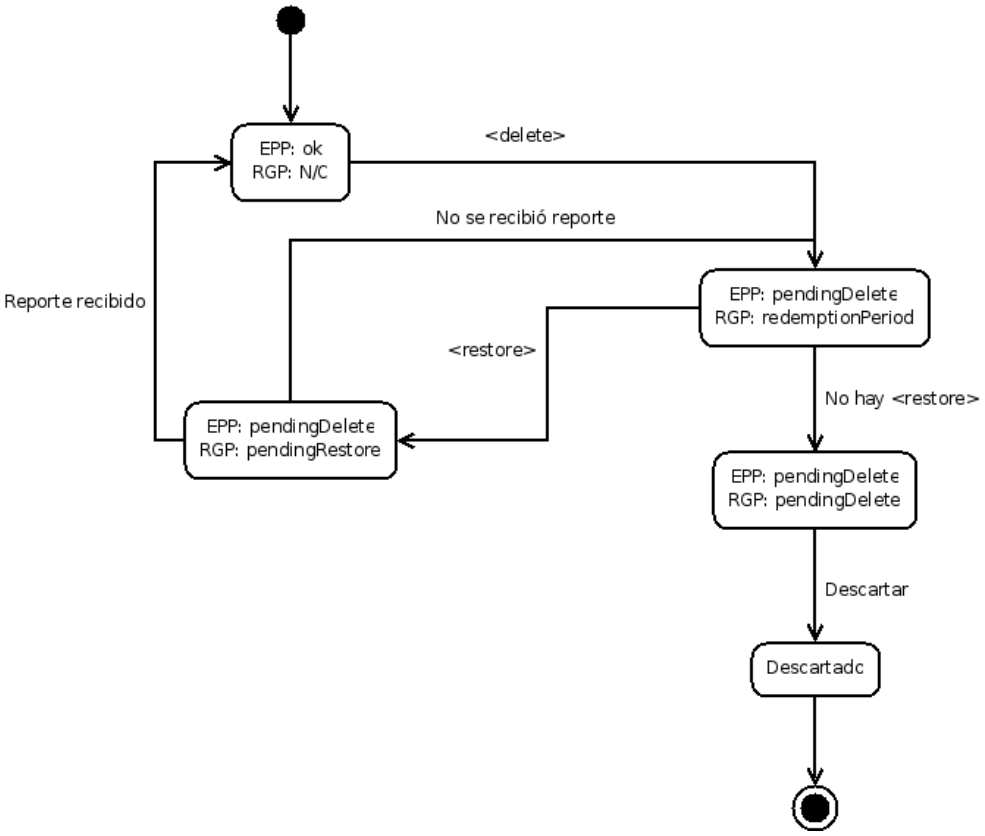


Figura 5.3: Diagrama de estados del *RGP*

Además del *RGP*, existen otros tipos de períodos de gracia:

Add Grace Period. Es un período que ocurre luego del registro inicial, durante el cual, si el dominio es borrado por el *registrar*, el *registry* otorga un crédito al *registrar* por el costo del registro.

Auto-renew Grace Period. Es similar al anterior, ya que el *registry* otorga un crédito al *registrar* si el dominio es borrado durante este período. Este período ocurre luego de que el dominio termina su período de validez y es renovado automáticamente por el *registry*, y el crédito es por el costo de la renovación.

Renew Grace Period. Es idéntico al anterior, pero comienza cuando el dominio es renovado explícitamente por el *registrar*.

Existen algunas controversias respecto a la utilización de Períodos de Gracia. Por ejemplo, el *AGP*, ha sido utilizado por *cybersquatters*[3] para determinar cuáles son los dominios más rentables. Éstos, registran los nombres, y durante el *AGP*, aplican técnicas para determinar si el dominio es rentable. Si no lo fuera, el dominio es borrado, y el *squatter* recobra el dinero invertido en el registro [45].

5.8. Resumen del capítulo

Existen diversas extensiones que diferentes interesados han realizado al *EPP* estándar para adaptarlo a sus necesidades.

El *gTLD .COOP*, destinado a cooperativas, incorpora un proceso de validación para determinar si determinada persona u organización puede registrar dominios en esta zona. El *registry* recurre a los *Validation Partner*, quienes ayudan a confirmar la elegibilidad del contacto como *registrant* de *.COOP*, y los *sponsors*, que son otras cooperativas, u organizaciones de cooperativas, que pueden proporcionar información sobre el mismo. La incorporación del proceso, se realiza agregando la información al objeto contacto, y extendiendo los comandos de manera acorde.

Los Nombres de Dominio Internacionalizados, son nombres de dominio que contienen caracteres fuera del repertorio *ASCII*, por ejemplo, *pingüinos.org.uy*. El *DNS* es capaz de manejar estos nombres si son almacenados

[3]El *domain tasting* es la práctica destinada a obtener provecho económico, registrando nombres de dominio que tienen alta probabilidad de ser registrados por otras organizaciones, y es realizada por los *cybersquatters*. El *cybersquatter*, registra el nombre, para luego vendérselo al interesado por un costo mayor al usual.

en formato *punycode*, definido por los estándares *Stringprep* y *Nameprep*. Como este código sólo contiene caracteres *ASCII*, es posible proveer nombres de este tipo sin modificar *EPP*, siempre y cuando los *registrars* los registren en el formato correcto. Puede ser útil modificar los comandos de consulta y de creación para tomar en cuenta los nombres internacionalizados, y modificar el objeto dominio para que éste indique el idioma utilizado, y extender los comandos de consulta y transformación para que puedan manejar este dato.

La incorporación de *DNSSEC* implica que el *registry* deba proveer el registro *DS* asociado a un dominio. Para esto, es necesario agregar un atributo al objeto dominio y extender los comandos de transformación (para manejar la asociación) y los de consulta (para mostrarla).

En el *ccTLD* de Polonia, es posible reservar nombres y manifestar interés en nombres registrados para cuando éstos queden disponibles. Con la extensión *book* al comando *create* de dominios es posible realizar la reserva, y mediante el agregado del objeto *future* es posible manifestar la intención de adquirir un nombre. Finalmente, el agregado del comando *report*, permite que los *registrars* obtengan listados de los objetos que tienen registrados.

Dentro del registro brasileño, se destacan la introducción de un *ticket* asociado al registro de un nombre, y la diferenciación del estado *pendingCreate* en un dominio. El *ticket* permite garantizar el principio *FCFS* y conocer el estado de la solicitud de registro. Mediante la diferenciación de pendientes, es posible determinar el motivo por el cual está retrasada una acción. Por último, está definido un algoritmo de similaridad entre nombres *IDN* y nombres *ASCII*, con el cual es posible determinar si son equivalentes.

Los Períodos de Gracia definidos por la *ICANN*, buscan habilitar un período durante el cual es posible revertir una acción luego de realizada. El más importante es el *Redemption Grace Period*, que permite deshacer el borrado de un nombre. Esto se logra modificando el comportamiento del comando *delete* y modificando el objeto dominio para indicar el estado del mismo. Existen otros cuyo uso es controversial. En particular, el *Add Grace Period* (que ocurre luego de la creación de un dominio) permite ataques de *Domain Tasting*.

Capítulo 6

Software de Repositorio para Registro de Nombres de Dominio

6.1. Introducción

En este capítulo se analizan diversos *software* que implementan protocolos *registry-registrar*, poniendo especial énfasis en *OpenReg*, que es el utilizado por SeCIU. También, se hace un análisis de librerías que permiten a los desarrolladores, producir aplicaciones que se conecten a servidores *EPP* para enviar comandos, y se analizan algunas herramientas existentes que hacen esta tarea.

Las soluciones analizadas, fueron evaluadas, en primera instancia, por el protocolo que implementaran, y en segunda instancia por el sistema de licenciamiento ofrecido. Con este criterio se relegaron opciones que no implementaran *EPP*, o que no existieran versiones libres de las mismas.

6.2. *OpenReg*

OpenReg[1], es el sistema *EPP* utilizado por SeCIU como repositorio de objetos. Además de Uruguay, es utilizado también por el *ccTLD* Venezuela. Fue desarrollado por *ISC*[2], y originalmente estaba pensado para gestionar la zona *org*. Se encuentra desactualizado, ya que su última versión data del año 2003. Soporta las RFC 3730 a 3734 ([46], [47], [48], [49], [50], y [7]), que definen el protocolo *EPP* y sus mapeos a nombres de dominio, contactos y *hosts*, así como el transporte subyacente al protocolo y las guías para extenderlo, cuya última versión se especifica en la *RFC* 5730 [1]. Ofrece una interfaz *EPP* y un servicio de *WHOIS*. Está desarrollado en *Perl* y utiliza *PostgreSQL* como motor de base de datos.

En la figura 6.1 puede verse los componentes de la aplicación.

Éstos son:

EPP Front-end. Implementa el protocolo *EPP*. Este demonio es el encargado de recibir los pedidos de los diferentes *registrars* y devolver sus respectivas respuestas. Decodifica los *XML* de los pedidos *EPP* y envía las acciones al demonio *xaction* a través del canal *MsgBus*. El demonio *xaction* valida el pedido según el *registrar* que lo envía, y realiza las operaciones necesarias en la base de datos, para luego enviar la respuesta al *EPP Front-end* a través, nuevamente, del canal *MsgBus*.

Protocolo *MsgBus*. Este protocolo es implementado por el demonio *msgq* y proporciona un canal de comunicación entre procesos que pueden

[1] https://www.isc.org/software/openreg/

[2] https://www.isc.org

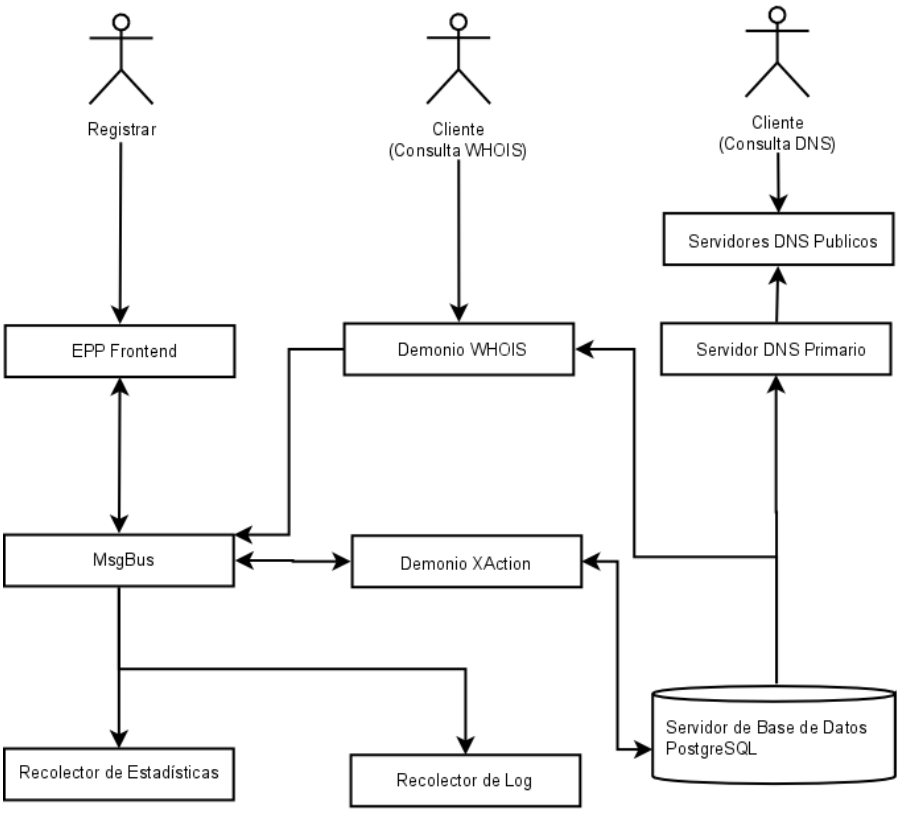


Figura 6.1: Módulos *OpenReg*

estar corriendo en una o más máquinas. Ofrece un servicio mediante el cual los procesos se pueden registrar para recibir información en varios canales. Cuando el demonio recibe un mensaje, lo coloca en el canal adecuado y lo reciben todos los procesos suscritos a éste. Actualmente, el transporte de mensajes, se realiza sobre *TCP*.

Demonio *WHOIS*. Este componente se encarga de realizar las consultas de base de datos necesarias para satisfacer los pedidos de información del servicio de *WHOIS*, cuya definición se encuentra en la *RFC* 3912 [51]. El demonio soporta búsquedas para dominios, contactos y, hosts individuales, y, adicionalmente a estas consultas básicas, ofrece un comando para reportar la información completa de un dominio dado.

Demonio *xaction*. El demonio *xaction* realiza operaciones de bases de datos en nombre del *EPP Front-end*. Esto incluye el alta, baja y modificación de dominios, contactos y servidores de nombres. La comunicación entre el *front-end* y el demonio *xaction* se realiza mediante el intercambio de estructuras de datos a través de un canal en el *MsgBus*. Antes de intentar cualquier operación en la base de datos, el demonio chequea la validez de los datos recibidos para asegurarse que toda la información necesaria fue proporcionada. También chequea los permisos del *registrar* que pidió la operación, para asegurarse que dicha operación le es permitida.

Base de Datos *SRS*. La base de datos *SRS* es una base de datos relacional, implementada en *PostgreSQL*, que contiene toda la información sobre los dominios, contactos y servidores de nombres, así como de las relaciones entre éstos, para un registro de dominios (*registry*) dado. Esta base, es tanto consultada como modificada por el demonio *xaction* según sea solicitado, y es tratada como una fuente de datos de "sólo-lectura" por el demonio *WHOIS*, y por los diversos generadores de reportes incluidos, incluyendo al *script* de generación de archivos de zona, quien se encarga de procesar los datos de la base para generar la información de la zona y publicarla en el servicio de *DNS*.

Además de estos componentes, existen otros que se encargan del registro de información en diversos puntos de la aplicación, y otros más que realizan reportes basados tanto en la base *SRS*, como en los diversos *logs* del sistema. Por ejemplo, los *generadores de reportes srs*, son una colección de *scripts* que consultan la base *SRS*, y elaboran diversas estadísticas sobre el registro, presentándolas en reportes fácilmente entendibles. Lamentablemente, varios de estos componentes *no* están implementados.

OpenReg tiene varias limitaciones importantes en su implementación. Las principales son:

Operaciones modo "pendiente" No hay soporte para aquellas operaciones que necesiten completarse de manera diferida.

Comandos no implementados Los comandos *poll* y *transfer* no están implementados.

Manejo incompleto de los objetos *AuthInfo* No se corrobora que sean válidos al realizar cambios sobre objetos

Soporte incompleto para estados No se realizan los controles necesarios para los cambios de estados de los objetos, ni se manejan adecuadamente los conflictos.

Portabilidad Su implementación tiene problemas con la manera en que las versiones actuales de *Perl* manejan las cadenas *UTF-8*, por lo que no funciona con las versiones más nuevas de *Perl*, desde la 5.10 en adelante. Esto hace que su uso sea problemático en versiones actuales de GNU/Linux.

6.3. *CoCCA*

CoCCA[3], es un sistema *EPP* completo que tiene funcionalidades de *registry* y de *registrar*. Para cumplir estas funciones cuenta con interfaces web de gestión, además de la interfaz *EPP*.

Está desarrollado en *Java*, utilizando el servidor de aplicaciones *RESIN*, y el motor de base de datos *PostgreSQL*, e implementa la versión estándar de *EPP*.

Posee un diseño escalable que separa el servidor *EPP* de la aplicación de gestión web en dos aplicaciones independientes. Es posible, incluso, ejecutar varias instancias del servidor *EPP* para repartir la carga. Respecto a su performance, ha sido utilizado exitosamente en *registries* con 1.25 millones de dominios. En la Figura 6.2 se muestra un ejemplo de como podría ser desplegado en un entorno de alta disponibilidad.

La administración se realiza vía web, y los *registrars* pueden utilizar las interfaces *EPP* y web. Soporta *IDN* y *ENUM*, y tiene características de seguridad avanzadas. Por ejemplo, realiza control de *Domain Tasting*. Posee

[3]http://www.cocca.cx/

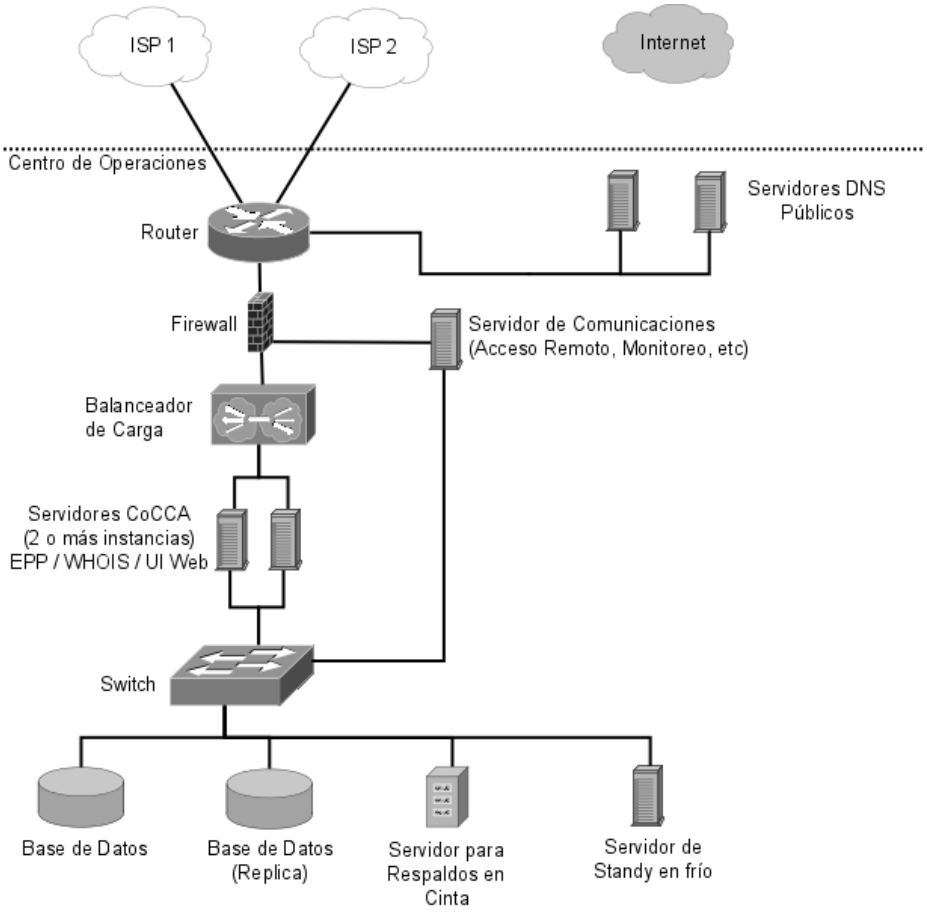


Figura 6.2: Ejemplo de entorno *CoCCA*

funcionalidades de valor agregado para el *registry*, como facturación automática en *PDF*, *WHOIS* personalizable, renovaciones automáticas de dominios, transferencias de dominios, restricciones configurables de nombres (tanto palabras ofensivas como nombres genéricos) y de políticas del *registry*.

Algunos de los países que utilizan este Software son Afganistán (*.af*), Christmas Island (*.cx*), Dominica (*.dm*), etc.

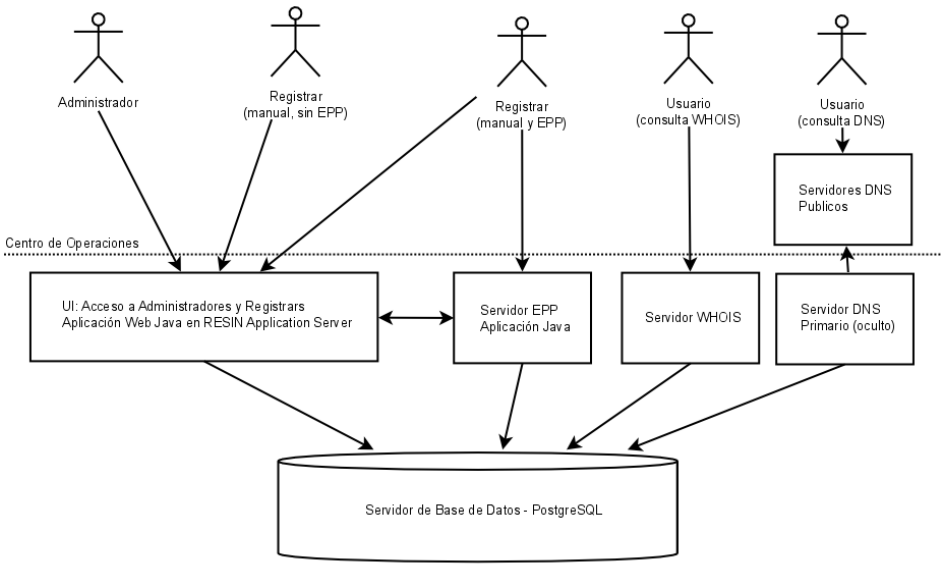


Figura 6.3: Módulos *CoCCA*

En la Figura 6.3 pueden observarse los componentes principales de este sistema, y como interactúan con los diferentes tipos de usuarios (administradores, *registrars* y usuarios). Los dos componentes a más importantes son la Aplicación de Gestión Web, que utiliza el servidor de aplicaciones *RESIN*, y el servidor *EPP*, que es una aplicación independiente basada en *QuickServer*[4].

Debido a la doble funcionalidad de este software, para el *registry* y para los *registrars*, a nivel de datos se mantienen bases de datos independientes. Internamente utiliza una tercera base de datos para la generación de las zonas a publicar.

En cuanto a su distribución, hasta la versión 2.29 se encuentra disponible en *SourceForge* bajo licencia *GNU* (con fecha 18/11/2008)¸ siendo esa la versión analizada en esta sección. Existen versiones más nuevas, en particular la última en producción (a la fecha) es la 2.52 del 6/11/2009, pero no son libres, estando disponible sólo para los miembros de la organización. Sin

[4]librería para Java para el desarrollo de servidores TCP

embargo, existe una nota en el sitio oficial diciendo que están dispuestos a enviar la última versión a los *ccTLD* que lo soliciten sin costo alguno.

CoCCA ofrece, además, el servicio de soporte, instalación y migración de datos por 10.000 dólares neozelandeses anuales (aproximadamente 7100 dólares americanos al momento de realizado este relevamiento).

6.4. *FRED*

FRED[5] es desarrollado por el *NIC* de República Checa, y cuenta con un servidor *EPP*, un cliente *EPP* y una aplicación de administración web. Es un proyecto activo, ya que la última versión disponible es de enero 2010. Tiene una licencia *GPLv2* y se basa en componentes libres o de código abierto, como *Apache*, *PostgreSQL*, *Python*, *omniORB*, etc. Está en producción en los registros de República Checa y Angola, y se está probando en los de Tanzania, Costa Rica, Ruanda y Senegal. Ofrece funcionalidades avanzadas como facturación (con interfaz bancaria), envío de e-mail a partir de plantillas, verificaciones técnicas de los *nameservers*, soporte *IPv6*, *IDN*, *DNSSEC* y *ENUM*, historia completa de cambios con posibilidad de *rollback*, período de gracia, etc.

Brinda una interfaz de administración web y otra de consultas a *WHOIS*. Además tiene una interfaz *EPP* (sobre *SSL*) para *registrars*, para la cual se proporcionan dos clientes: uno para consola y otro gráfico (recientemente añadido). Estas tres interfaces de *frontend*, corren bajo *Apache*, y para la comunicación con el *backend* (llamado Registro Central) utilizan *omniORB/ORBit*.

Este último componente, acepta las peticiones de las operaciones relativas a los dominios, los contactos u otros objetos en el registro, envía las consultas a la base de datos y devuelve los resultados a través de la interfaz correspondiente. Realiza además las tareas de mantenimiento regular, como la verificación de los registros (vencimiento del dominio, control técnico de los datos de dominio, etc) e invocación de las acciones pertinentes, basándose en los resultados de estos controles (información sobre los contactos del dominio, borrado de dominios de una zona, etc.). El registro central también, periódicamente, genera los correspondientes archivos para las zonas administradas. Estos componentes pueden verse en la figura 6.4.

El modelo *EPP* no es el estándar. Los contactos tienen algunas modificaciones menores y la manera en que se manejan los *hosts* es diferente: se crea una nueva entidad, el *NSSet*, que agrupa *hosts*. De esta manera se agrega un nivel más de indirección, ya que, mientras en *EPP* estándar un dominio tiene

[5] http://fred.nic.cz/

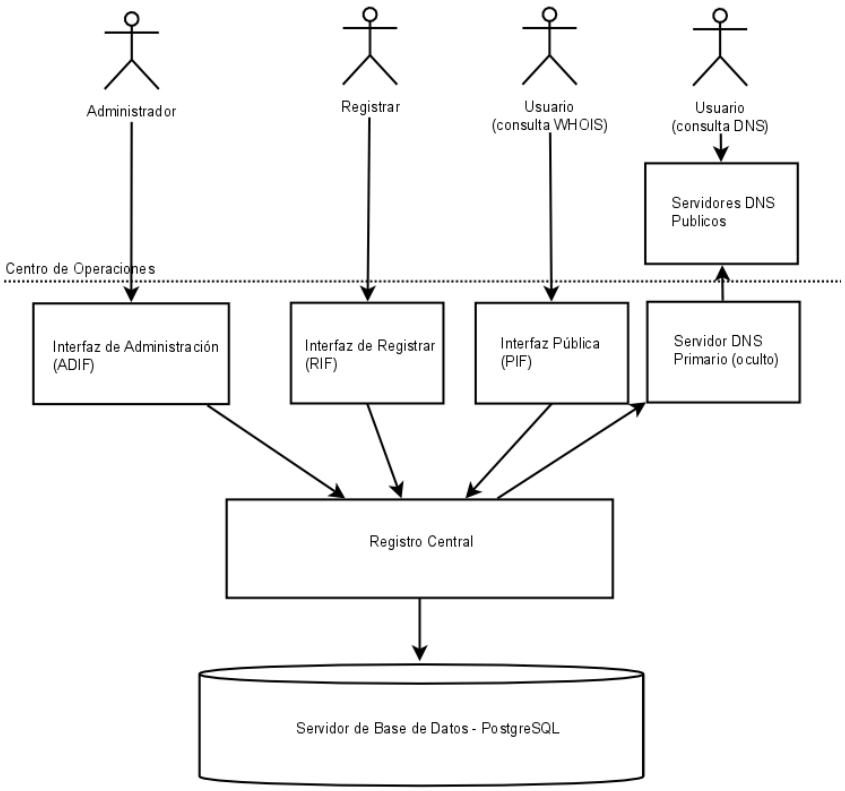


Figura 6.4: Módulos *FRED*

asociado *hosts*, en *FRED* un dominio tiene asociado un *NSSet* que agrupa *hosts*. Los contactos técnicos ya no se asocian al dominio directamente, sino que se asocian al *NSSet*. Esta extensión es realizada para simplificar la manera en que se modelan los dominios que comparten los mismos *hosts*, ya que, de este modo, estos dominios comparten el mismo *handle* de *NSSet*.

Figura 6.5: *EPP* utilizado por *FRED*

6.5. Otros sistemas

Durante el relevamiento realizado, se relevaron otros sistemas de registro centralizado que no poseen una interfaz *EPP*. Debido a esta carencia, y que el foco de este trabajo se encuentra en *EPP*, no se realizó un análisis profundo de los mismos. Dichos sistemas son:

Codev-NIC. Es un proyecto desarrollado en conjunto por los *NIC* de Francia, Madagascar y Costa de Marfil[6]. Está orientado a *registries* pequeños, y una de sus principales virtudes es la posibilidad de configurarlo según las políticas de cada *registry*.

Su desarrollo está basado fuertemente en plantillas. Por ejemplo, se utilizan plantillas para que las políticas configuradas generen las tablas correspondientes en la base de datos. Para el acceso de los *registrars* cuenta con una interfaz *XML-RPC*, además de acceso directo a la base de datos. Esta interfaz *XML-RPC* no cumple con el estándar *EPP*.

DNRS. Está desarrollado en Nueva Zelanda[7], y brinda un software cliente, que es utilizado por los *registrars*, y otro que actúa como servidor, y es

[6]http://codev-nic.generic-nic.net/
[7]http://www.dnrs.net.nz/

utilizado por los *registries*. Es un proyecto activo, siendo su última versión de marzo de 2010. Se distribuye bajo licencia *GPL*. Originalmente utilizaba *EPP* como protocolo de comunicación, pero actualmente migró al protocolo *NZ-SRS* [52]. Este es un protocolo basado en *XML* similar a *EPP*, siendo la principal diferencia entre ellos que *NZ-SRS* está orientado a *registries* más gruesos, y que por lo tanto incluye otra clase de funcionalidades (por ejemplo datos de facturación) que no están contempladas en *EPP*.

mod-epp. En este caso no se trata de un servidor *EPP* completo[8], sino que se trata de un módulo para *Apache* y de algunos scripts *CGI* que realizan las funciones de *registry*.

tinyReg. Es un sistema de registro de dominio que tiene por objetivo ser mínimo, funcional, y robusto a la vez[9]. Está desarrollado en *Perl* y sus interfaces son web. No soporta *EPP*.

6.6. Librerías *EPP*

6.6.1. *Universal Registry/Registrar Toolkit*

El proyecto *Universal Registry/Registrar Toolkit*[10], también conocido como *EPP-RTK*, brinda una solución *Java* para interactuar con un servidor *EPP*. A continuación listan las clases principales y su distribución:

Clases generales *EPP*

A continuación se detallan las principales clases genéricas.

***EPPClient*:** clase que encapsula la conexión y comunicación con el Servidor *EPP*.

***EPPGreeting*:** representa el *greeting* enviado por el servidor *EPP*, y permite conocer detalles del servidor, como, por ejemplo, las extensiones soportadas.

***EPPLogin*:** permite que el *registrar* establezca una conexión.

***EPPLogout*:** usada para finalizar la conexión con el servidor.

[8] http://sourceforge.net/projects/aepps/
[9] http://www.nsrc.org/tinyReg/
[10] http://sourceforge.net/projects/epp-rtk/

***EPPPoll*:** permite recuperar mensajes enviados por el servidor. El *registrar*, debe enviar periódicamente un comando *poll* por dos motivos: mantener activa la conexión y recibir notificaciones pendientes.

***RTKBase*:** es una clase abstracta, que es la clase base de las demás clases *RTK* del proyecto. Además define el formato de fecha y varias restricciones.

***EPPXMLBase*:** clase abstracta, base de la decodificación *XML*, y superclase de nuevas clases que se agreguen por extensiones al protocolo.

***EPPTransportXX*:** es una familia de clases cuyos nombres comienzan con *EPPTransport* y corresponden a los manejadores de la capa de transporte en el protocolo *EPP*. La clase base es *EPPTransportBase*. Actualmente, están implementados los transportes por *TCP, TCP/TLS* y *SMTP*.

***EPPStatus*:** es responsable del manejo de estados.

Contactos *EPP*

Este grupo de clases permiten ejecutar acciones relativas a los *contactos EPP*.

***EPPContactCheck*:** se utiliza para ejecutar comandos *check* que involucren contactos.

***EPPContactInfo*:** se utiliza para ejecutar comandos *info* sobre contactos.

***EPPContactCreate*:** se utiliza para ejecutar comandos *create* para crear nuevos contactos patrocinados por el *registrar*.

***EPPContactUpdate*:** se utiliza para ejecutar comandos *update* sobre contactos patrocinados por el *registrar*.

***EPPContactDelete*:** se utiliza para ejecutar comandos *delete* sobre contactos patrocinados por el *registrar*. Dichos contactos no deben estar asociados a ningún dominio.

***EPPContactTransfer*:** se utiliza para ejecutar comandos *transfer* sobre contactos.

Análogamente, existen clases para el manejo de dominios y *hosts*.

EPP-RTK, utiliza un conjunto de módulos *IDL*[11] basados en los objetos y comandos *EPP*. Estos módulos, sirven para cubrir las diferencias entre los esquemas *XML* y *EPP* utilizado como una *API*. Especifican los datos usados y las interfaces que se deben realizar. También, se usan para asegurar que diferentes implementaciones (de extensiones distintas) sigan una estructura similar. Son distribuidos a modo de referencia, ya que desde el software se utilizan compilados como clases *Java*. Están distribuidos en los siguientes paquetes:

- *org.openrtk.idl.epp*
- *org.openrtk.idl.epp.contact*
- *org.openrtk.idl.epp.domain*
- *org.openrtk.idl.epp.host*

Las clases *IDL* (todas comienzan con *epp_*) son usadas, principalmente, como contenedores de datos para los pedidos y las respuestas. Las clases en el paquete *com.tucows.oxrs.epp.rtk.xml* (todas comienzan con *EPP*) son usadas para convertir los datos definidos en el *IDL* desde y hacia el *XML* utilizado en la comunicación con el registro *EPP*.

Para cualquier comando *EPP*, se va a usar un trio de clases: una para los datos del pedido, otra para los datos de la respuesta, y una tercera para la conversión a *XML*. Por ejemplo, en el comando *check* de dominios, las tres clases involucradas son *epp_DomainCheckReq*, *epp_DomainCheckRsp* y *EPPDomainCheck* respectivamente, además de las clases involucradas en el manejo de la conexión al servidor. La curva de aprendizaje para su utilización, es relativamente rápida debido a que existen guías con buenos ejemplos de código.

Las extensiones *EPP* soportadas son las definidas para los dominios *.biz, .us, .name, .org* y *.info*, y existe un complemento, que puede ser descargado, para agregar soporte a las extensiones del *registry Afilias*[12].

La última versión disponible es la 0.96, con fecha 25/08/2009, y está publicada con licencia *LGPL*.

[11] Interface Definition Language

[12] http://www.afilias.info/global-registry-services

6.6.2. *Perl EPP Library (Net::EPP)*

El proyecto *Net:EPP*[13], ofrece una serie de módulos de *Perl* que implementan diversas funciones relacionadas con el protocolo *EPP*. Los principales módulos son los siguientes:

***Net::EPP::Protocol*:** Es una implementación de bajo nivel del protocolo. Brinda dos funciones: *get_frame* y *send_frame*.

Net::EPP::Client Es un cliente de bajo nivel que permite enviar los comandos en formato *XML* directamente al servidor.

Net::EPP::Simple A diferencia de *Net::EPP::Client*, este cliente brinda la posibilidad de realizar las tareas más comunes, por medio de una *API*.

Net::EPP::Frame Es un constructor de *frames EPP* basado en *XML::LibXML* que simplifica las tareas de creación de los *XML* correspondientes a los comandos *EPP*. Se usa en conjunto con *Net::EPP::Client*.

Net::EPP::Proxy Se trata de un servidor *proxy* que actúa de intermediario entre múltiples clientes *EPP* y un servidor *EPP*, para que todos los clientes utilicen una única conexión con el servidor, en lugar de que cada uno deba crear la suya.

Net::EPP::ResponseCodes Es una librería para exportar los valores de los códigos de respuesta definidos en el protocolo *EPP* como constantes *Perl*. Por ejemplo el código 2001 se exporta como la constante *SYNTAX_ERROR*.

Estos módulos fueron creados, y son mantenidos, por *CentralNic* para uso de sus propios registradores, pero, desde su liberación original, han sido ampliamente utilizados por registradores y registros de todo tipo. Posteriormente, *CentralNic* optó por publicar los fuentes en un proyecto público (alojado por Google) para permitir que los interesados contribuyan en desarrollo de estas bibliotecas. Se distribuye mediante *CPAN*, el repositorio de módulos *Perl*.

6.6.3. *PHP EPP Library (Net_EPP_Client)*

Net_EPP_Client[14] es una clase que implementa un cliente *EPP* de bajo nivel para aplicaciones *PHP*. Gestiona las conexiones y el manejo de *frames*

[13]https://www.centralnic.com/company/labs/perl
[14]https://www.centralnic.com/company/labs/php

en una manera fácil de usar. El diseño de la clase *Net_EPP_Client* es muy similar a la del módulo *Perl Net::EPP::Client*, que también es desarrollado por *CentralNic*. La versión actual es la 0.0.3, del 25 de mayo de 2007, y se distribuye bajo licencia *GNU GPLv2* o posteriores. Está diseñada como una clase única, llamada *Client.php*, que depende del módulo *PHP PEAR*, utilizado para el manejo de errores.

La clase, brinda al programador los siguientes métodos:

connect: Acepta como parámetros *host*, puerto (por defecto 700), *timeout* (en segundos, por defecto 1) y *ssl*, para indicar si utilizar *SSL* o no. Retorna la cadena correspondiente al *greeting* en caso de éxito o un error *PEAR* en caso contrario.

getFrame: Es bloqueante, retorna la cadena con el *XML* de la respuesta (o un error *PEAR*).

sendFrame: Envía el *XML* recibido por parámetro al servidor.

request: Es un *wrapper* para las funciones *getFrame* y *setFrame*, simplificando la comunicación. Recibe como argumento la cadena correspondiente al *XML* a enviar, y retorna la correspondiente al *XML* de la respuesta.

disconnect: Cierra la conexión con el servidor.

6.6.4. *NeuLevel Registrar Toolkit*

NeuLevel Registrar Toolkit[15] es una *API* de alto nivel para *EPP*, disponible para *Java* y *C++*. Proporciona un modelo de objetos que hace sencillo conectarse a un servidor y enviarle comandos. Soporta las extensiones necesarias para comunicarse con el *registrar NeuLevel*[16], en particular las definidas para los dominios *.biz* y *.us*. Está compuesta por los siguientes paquetes:

com.neulevel.biz. Define e implementa las clases relacionadas con los dominios *.BIZ*

com.neulevel.epp.core. Define e implementa los componentes centrales del protocolo *EPP*.

com.neulevel.epp.core.command. Implementa varios comandos relacionados con el protocolo *EPP*.

[15]http://sourceforge.net/projects/epp-ver-04/
[16]http://www.neustar.com/

com.neulevel.epp.core.response. Implementa varias respuestas y componentes relacionados del protocolo *EPP*.

com.neulevel.epp.transport. Define e implementa varios mecanismos de transporte para el protocolo *EPP*.

com.neulevel.epp.transport.tcp. Implementa el mecanismo de transporte *TCP* para *EPP* usando *TLS* para seguridad.

com.neulevel.us. Define e implementa las clases relacionadas el registro de dominios *.US*.

6.6.5. *Net::DRI*

Net::DRI[17], es un conjunto de módulos *Perl*, orientados a objetos, que proporcionan una interfaz abstracta y uniforme para conectarse a los proveedores de nombres de dominio. Tiene licencia *GPL*, e implementa los siguientes protocolos:

1. *RRP*: definido en las *RFC* 2832 [53] y 3632 [54], es el utilizado por *.COM* y *.NET* antes de pasar a *EPP*, y sigue siendo utilizado por algunos *ccTLDs*.

2. *EPP*: usado por *.COM/.NET*, todos los nuevos *gTLDs* (*.INFO, .BIZ, .PRO*, etc...) y muchos *ccTLDs*.

3. *ENUM*: *EPP E.164* (*RFC* 4114 [55]) y *EPP ENUM Validation* (*RFC* 5076 [56]).

4. *DNSSEC*: *EPP SecDNS* (*RFC* 4310 [38])

5. Extensiones para varios *registries* específicos.

6. *WHOIS*: decodificación de información whois para dominios en *.COM, .NET, .ORG, .BIZ, .INFO, .AERO*, etc.

7. *DAS*: *Domain Availability Service* para *.BE, .EU, .TC, .VG, .GD, .AU* y *.NL + BookMyName + AdamsNames*.

8. *Web Services*: *AFNIC, BookMyName, Gandi, OVH* y *AdamsNames*.

9. *RRI*: usado por *DENIC*, *NIC* alemán.

[17] http://www.dotandco.com/services/software/Net-DRI/

10. *XCP*: usado por *OpenSRS*.

11. *IRIS*: IRIS (*RFC* 3981 [57] y asociadas).

Brinda también *Net::DRI::Shell*, una interfaz por línea de comandos que se puede utilizar para hacer todas las operaciones a través de *Net::DRI* sin escribir ninguna línea de código. Está plenamente documentada y permite procesamiento por lotes, con *logging* y estadísticas.

6.6.6. *EppClient API*

EppClient API[18] es una librería comercial para utilizar *EPP* desde la plataforma *.NET*. Soporta los registros de *.NO* y *.SE*. Brinda, además, un cliente web. Su costo es de €1200 para la versión de un único servidor, y €3500 para la versión sin restricciones. Además, tiene un costo de mantenimiento anual del 10 % del monto anterior. La licencia del cliente web, para 3 usuarios, tiene un costo anual de €600.

6.7. Clientes *EPP* genéricos

La lista de más abajo, describe algunos clientes que permiten interactuar con diversos registros.

Preppi [19] Es un cliente gráfico *EPP*, desarrollado por *CentralNic* para sistemas **NIX*, que permite conectarse a un servidor *EPP* y enviarle comandos en formato *XML*. Está escrito en *Perl* y se basa en la librería *Net::EPP* nombrada anteriormente. Se distribuye bajo licencia *GPL*.

EPPInterpreter [20] es un *frontend* gráfico para la librería *EPP-RTK* escrito en *Java*. Soporta *scripting* en varios lenguajes (*Python, Perl*, etc). Su última versión es de 2001.

EPP Testing Tool [21] Es un cliente que puede ser utilizado desde línea de comandos o desde una interfaz web. Su objetivo es simplificar las pruebas a servidores *EPP*, permitiendo ejecutar lotes de comandos *EPP*.

[18] http://eppclient.com/api/
[19] https://www.centralnic.com/company/labs/preppi
[20] http://eppinterpreter.sourceforge.net/eppinterpreter/docs/
[21] http://sourceforge.net/projects/epptt/

FRED EPP Client *FRED*, el servidor *EPP* Checo, distribuye, además del software servidor, dos variantes de su cliente: uno por línea de comandos y otro gráfico. Implementa las extensiones específicas de *FRED* (*NSSET*, etc), por lo que sólo puede ser utilizado para acceder a ese servidor y no a un servidor *EPP* estándar.

6.8. Clientes *EPP* específicos

Los clientes que se describen en esta sección, permiten trabajar con *registries* específicos.

***.IT*:** *EPP Ceglia Tools* es un conjunto de librerías *PHP* para realizar registros de dominios en el *ccTLD .IT*[22].

***AusRegistry*:** librerías cliente *EPP* para *AusRegistry*[23].

***Nominet*:** proyecto *PHP* para permitir la integración con *Nominet (.UK)*[24].

***SIDN*:** implementación de un cliente en *PHP5* para interactuar con los servidores *EPP* de *SIDN*[25].

***Enom*:** es otro proyecto *PHP*, en este caso para la integración con *Enom*[26].

***.NO*:** *Draupne EPP Client*, es un cliente gráfico *EPP* escrito en *Java*, diseñado para el registro del *ccTLD .NO*[27].

***.BR*:** *LibEPP-NICBR-CGI* y *PHPEPP-NICBR*, son librerías *C++* y clases *PHP* para la comunicación con el *NIC* de Brasil[28].

6.9. Extendiendo *OpenReg*

En esta sección se investiga la capacidad de extensión de *OpenReg*, dado que es el *software* de registro utilizado por SeCIU.

Gracias a la modularidad de *OpenReg*, los cambios requeridos para realizar una extensión, están centralizados en los módulos *EPP Front-end* y

[22] http://sourceforge.net/projects/eppcegliaclient/
[23] http://sourceforge.net/projects/ar-epp-clnt-lib/
[24] http://sourceforge.net/projects/nominetepp/
[25] http://sourceforge.net/projects/php5-sidnepp/
[26] http://sourceforge.net/projects/enomeppapi/
[27] http://sourceforge.net/projects/draupnegui/
[28] http://sourceforge.net/projects/epp-brnic-cgi/ y
http://sourceforge.net/projects/phpepp-nicbr/

xaction. El módulo *MsgBus* no se tiene que modificar, ya que su única intención es ser un medio de comunicación entre los procesos, y, por lo tanto, es transparente a los comandos *EPP* o sus argumentos. El resto de módulos (*Whois*, *Estadísticas*, *WriteZone*) no deberían ser modificados, salvo que la extensión a implementar necesite una modificación en alguno estos servicios.

La base *SRS*, debe actualizarse para agregar tablas nuevas, o campos a las tablas existentes, para poder almacenar los datos asociados a las extensiones agregadas.

Como ya fue analizado, existen tres mecanismos de extensión para *EPP*: modificando un objeto, modificando un comando (o su respuesta),y modificando el protocolo. Desde el punto de vista del desarrollo, estos mecanismos, se implementan realizando al menos uno de los siguientes cambios:

- Modificación del *XML* de entrada de un comando *EPP*.
- Modificación del *XML* de salida de un comando *EPP*.
- Creación de un nuevo comando *EPP*.

Por ejemplo, si se desea extender un objeto *EPP*, seguramente sea necesario modificar la entrada de los comandos *update* y *create* para recibir los nuevos atributos, y, además, una modificación de la salida del comando *info* para poder mostrar los atributos.

Las modificaciones al protocolo, son las que implican la creación de comandos nuevos.

A continuación, se describen los mecanismos de modificación enumerados.

6.9.1. Modificación del *XML* de entrada de un comando *EPP*

La primera parte de la modificación consiste en agregar la decodificación de los nuevos nodos. Esta decodificación, se realiza desde el fuente `EPPHandler.pm`, el cual está basado en el decodificador *SAX*. Básicamente, para cada nodo nuevo se debe insertar un nuevo ítem a la tabla `action`. Esta tabla contiene los manejadores a invocar para los diferentes eventos de la decodificación.

El nuevo ítem, deberá tener como clave el nombre del nodo esperado, y, como valor, un *array* con punteros a tres funciones:

1. la que se invoca al encontrar la etiqueta inicial
2. la que se invoca al detectar el contenido del nodo

3. una que se utiliza para realizar un post-procesamiento, por ejemplo, para obtener el valor deseado a partir de datos temporales guardados por las anteriores funciones.

En la mayoría de los casos, no es necesario utilizar las 3 funciones y puede indicarse un puntero nulo para omitir alguno de los eventos.

El decodificador brinda algunas variables que pueden ser utilizadas dentro de estas funciones

- la variable `heritage` permite obtener la lista de nodos padre del nodo actual
- los argumentos `text` o `attr` se utilizan para obtener el valor del nodo o los atributos respectivamente
- la variable `work` se usa como almacén temporal de datos para los eventos posteriores
- la variable `req` para retornar un valor una vez finalizado el procesamiento.

Existen dos funciones predefinidas que son de ayuda para casos simples: `_set_command` y `_set_params`. Al utilizar la primera, se carga la variable correspondiente al comando a partir de la etiqueta actual, y, al utilizar la segunda, se guarda una variable a retornar a partir del contenido del nodo. Por lo tanto, estas dos funciones simplifican la decodificación, ya que no es necesario implementar una función para una etiqueta, salvo que se requiera un comportamiento atípico, como validaciones particulares o calcular el valor a retornar a partir de nodos previos. Sólo con modificar este fuente de decodificación, *OpenReg* va a tomar los campos decodificados y se encargará de que se envíen al módulo *xaction*, que es quien ejecuta las acciones en la base de datos.

La segunda parte de la modificación, es hacer que el manejador correspondiente al comando ejecutado, contemple los nuevos elementos obtenidos. Cada uno de los manejadores de los diferentes comandos dentro de *xaction*, filtra, de todos los valores recibidos desde *EPP Front-end*, sólo aquellos que son de su interés. Por lo tanto, se deben agregar los nuevos nodos que se hubieran detectado para que no sean filtrados. Con el *hash* resultante, se invoca al fuente `DB.pm` donde están codificadas las modificaciones a Base de Datos necesarias. Todos los cambios de comportamiento, como por ejemplo nuevos campos a grabar en la base de datos, deben ser implementados en éste lugar.

6.9.2. Modificación del *XML* de salida de un comando *EPP*

En este caso, lo primero a modificar es el fuente `DB.pm` para codificar la extracción de la base de datos de los nuevos nodos a devolver. El caso más simple es cuando sólo se desea agregar un nuevo código de retorno, el cual debe ser agregado a `EPPResultCode.pm`, fuente donde se ubican las constantes para todos los códigos de retorno.

Si es necesario retornar nuevos datos, se debe codificar en el fuente `EPPWriter.pm`, agregando, para la función correspondiente al comando actual, los nodos requeridos. Se debe invocar a las funciones `startTag`, `dataElement`, y `endTag` para insertar la etiqueta inicial, el contenido, y la etiqueta final respectivamente.

6.9.3. Creación de un nuevo comando *EPP*

El caso de un nuevo comando *EPP*, incluye a los dos casos anteriores, pero además se debe tener en cuenta varios cambios adicionales.

En lo que respecta a la decodificación del *XML*, se necesita crear una nueva función para el comando y agregar un nuevo ítem al mapa de comandos, llamado *cmdmap*. También es necesario agregar una nueva función en el fuente `EPPRegistrar.pm` para que el comando sea enviado al *MsgBus*. Luego, en los fuentes `xaction.pl` y `DB.pm`, es necesario crear una función por cada nuevo comando, donde se codificará el comportamiento requerido.

Al momento de generar el *XML* a retornar en el fuente `EPPWriter.pm`, se debe tener en cuenta que es necesario invocar las funciones `_preamble` y `_postamble` para generar los nodos comunes a todos los comandos. Gracias a esto, sólo es necesario codificar la inserción de los nodos que corresponden específicamente al comando creado, teniendo la precaución de que se realice únicamente para el caso que la ejecución del comando fue exitosa, ya que en caso de un error no es necesario codificar el retorno del código correspondiente, porque esto ya está implementado dentro de las funciones anteriormente citadas.

6.9.4. Modificación de generación de archivo de zona

Los archivos de zona son generados por el módulo `write-zone.PL`, por lo tanto, cualquier cambio en la estructura del archivo debe ser incorporado a dicho módulo. Por ejemplo, una razón para hacer este tipo de cambios, es incluir los registros *DS* en el archivo de zona a publicar, necesarios para el funcionamiento de *DNSSEC*.

Este fuente está estructurado como una serie de consultas a la base de datos, donde cada una obtiene un tipo de registro de recurso a publicar. Por cada una de ellas, se formatean los registros devueltos según la estructura de dicho tipo, y se los escribe en el archivo correspondiente.

Dependiendo de la modificación requerida, podría ser necesario cambiar alguna de las consultas existentes, cambiar el formato de lo que se escribe en el archivo de zona, o crear nuevas consultas.

Para el caso del ejemplo (generación de registros *DS*) es necesario crear una nueva consulta que retorne los registros *DS* en condiciones de ser publicados, y para cada uno de ellos, formatearlos y escribirlos.

6.9.5. Corrección de problemas de portabilidad

Existen problemas, causados por errores en tiempo de ejecución, para la comunicación entre los diferentes procesos que componen *OpenReg*, que provocan que no sea posible loguearse para realizar tarea alguna. Dichos errores están causados por cambios en el manejo de las cadenas de caracteres *UTF-8* en *Perl*, ocurridos entre la versión 5.8 (por ejemplo, incluida en *Debian GNU/Linux 4.0 Etch*) y 5.10 (incluida en *Debian GNU/Linux 5.0 Lenny*).

Para la comunicación entre los diferentes procesos que forman *OpenReg* (ver Sección 6.2), se utiliza el componente *MsgBus*. Éste define un formato especial para los mensajes, que a cada *string* a enviar, antepone el largo del mismo. Este largo se indica con el caracter, cuya posición en la tabla *ASCII* se corresponde con el largo del *string*. Con esta codificación, el destinatario puede conocer la cantidad de caracteres que debe leer hasta encontrar el fin del mensaje.

Cuando los valores a codificar son chicos, no ocurre ningún problema. Sin embargo, cuando los valores aumentan hasta que el caracter tiene una representación en *UTF-8* de más de un *byte*, se produce el error citado. Las versiones 5.8 (y anteriores) de *Perl* retornan la cadena resultante de la concatenación del largo con el mensaje sin hacer conversión alguna. Sin embargo, las versiones 5.10 en adelante, reconocen que se está concatenando un caracter (correspondiente al largo) que debería codificarse con más de un *byte*, por lo que automáticamente agregan los *bytes* necesarios para convertirlo en una cadena *UTF-8* válida. Cuando ésta es recibida por el destinatario, al intentar decodificarla encuentra caracteres sobrantes no esperados, que fueron agregados automáticamente por el intérprete *Perl*, y al no poder realizar la decodificación exitosamente, el mensaje es descartado.

Para solucionarlo, existen dos enfoques posibles. El primero de ellos consiste en modificar los fuentes *Perl* que realizan la codificación de los mensajes, eliminando los *bytes* que fueron agregados durante la conversión automática

a una cadena *UTF-8* válida. El segundo, consiste en modificar la decodificación de los mensajes agregando la lógica necesaria para poder interpretar correctamente mensajes codificados como *UTF-8*.

El segundo enfoque es claramente más complejo que el primero, debido a que la decodificación de los mensajes es relativamente complicada, pero brinda mayores garantías para solucionar el problema de fondo. Al momento de implementar la solución, debe tomarse en cuenta que hay una variante de ese error, que afecta al momento de intentar enviar caracteres latinos (por ejemplo en los datos de un contacto), por lo que debe procurarse resolver ambos problemas a la vez.

6.10. Conclusiones

Luego de realizado este análisis, se puede concluir que *OpenReg* está relegado respecto a otras soluciones debido a su falta de soporte y actualización. De las alternativas analizadas, las más interesantes son *CoCCA* y *FRED*.

El software *FRED* utiliza un modelo de datos diferente al de *OpenReg* (que es el utilizado por SeCIU), mientras que *CoCCA*, si bien implementa el mismo modelo, ofrece dificultades debido a sus características de licenciamiento. También difieren en aspectos funcionales: *CoCCA* dispone de más características orientadas a los *registrars*, y *FRED* a los *registries*.

Se detectaron varias librerías que permiten desarrollar rutinas que envíen comandos *EPP*, entre las cuales destacan *EPP-RTK*, escrita en Java, y *Net::DRI*, desarrollada para *Perl*. Asimismo, se identificaron los clientes *Preppi*, que es un cliente gráfico para enviar solicitudes en *XML* y visualizar sus respuestas, y *EPP Tesing Tool*, que permite ejecutar lotes de comandos.

Estos, y otros aspectos, son analizados en profundidad en la Sección 10.1.1 del capítulo de conclusiones.

Parte II

Desarrollo

Capítulo 7

Definición de Extensiones para el Registro de Dominios .UY

7.1. Introducción

EPP, es un protocolo diseñado para proveer un registro de objetos en un escenario donde existen *registries* (*registros*, encargados de guardar objetos y proveer servicios sobre los mismos), y *registrars* (*registradores*, quienes están interesados en registrar objetos y obtener acceso a ellos). Así, las operaciones del *registry*, permiten que los *registrars* guarden objetos nuevos, o que consulten, modifiquen o borren los que ya registraron. Otros comportamientos, que no tengan esta característica, deben ser manejados por medios alternativos.

Si bien el protocolo está diseñado de manera genérica, fue concebido con el objetivo de mantener el conjunto de nombres de dominios de Internet, asociados a diferentes zonas. Dicho de otra manera, fue diseñado, como primer uso, como un repositorio de nombres de dominio de Internet, y de otros objetos asociados, como personas responsables del dominio, y los *hosts* encargados de resolver las consultas de *DNS* sobre el mismo. A pesar de esto, su generalidad permite utilizarlo para registrar otro tipo de objetos totalmente distintos a los relacionados al negocio de nombres de dominio, así como permite extender los objetos de este negocio, para adaptarlo (a *EPP*) a las particularidades del registro de determinadas zonas.

En este capítulo, se presenta el resultado de un análisis de las necesidades particulares del registro de nombres de dominio de Internet en la zona .UY, dando un conjunto de extensiones y modificaciones a los mapeos de *EPP* relativos a éstos, contactos asociados a los mismos, y a los servidores de nombres, que permiten modelar las características que tiene el negocio en Uruguay.

Las extensiones aquí presentadas, se encuentran basadas en investigaciones realizadas sobre extensiones que otros registros han realizado a los objetos (ver Capítulo 5) y en entrevistas con los responsables del registro de nombres en Uruguay. Para cada una, se presenta un análisis de la necesidad contemplada, se analiza y define una solución, y se presentan algunos lineamientos para su implementación.

La presentación de las mismas, está realizada con un enfoque *ausente-excluido*, es decir, que si no se plantea extender un comando, es porque el mismo no se extiende. A pesar de esto, algunos comandos no extendidos, son presentados para justificar su *no-extensión*.

Se trabajó en 7 extensiones:

- Manejo de Dominios Críticos
- Registro de Nombres *IDN*
- Extensiones *DNSSEC*

- Validación de acciones de transformación de objetos del repositorio
- Listado de objetos
- Período de gracia
- Auditoría de acciones

Cabe destacar que un aspecto importante tomado en cuenta para definir estas extensiones, fue el hecho de la futura unificación del registro de nombres de dominio en la zona .UY, y la apertura a diversas entidades registradoras (ver Capítulo 3), lo cual introduce dificultades que deben ser contempladas.

7.2. Manejo de Dominios Críticos

7.2.1. Problema

Algunos dominios registrados por SeCIU en .UY, deben estar siempre activos. Para evitar que estos dominios sean borrados, se define como política central del *registry* la calidad de *crítico* (ver Sección 3.2). Actualmente, dicha calidad se maneja a nivel del único *registrar* existente: SeCIU. Ante la posibilidad de admitir más *registrars*, se hace necesario manejar dicha calidad en el *registry*.

Por motivos de seguridad, la modificación de esta calidad debe ser realizada con métodos *off-line* por caminos alternativos a *EPP*, para evitar que cualquier *registrar* decida si sus dominios son o no críticos.

7.2.2. Solución

Extensión a la respuesta del comando *delete* aplicado a los nombres de dominio

Para evitar el borrado de dominios críticos se agrega un atributo *booleano* al objeto *dominio* para indicar si el dominio representado es o no *crítico*. Dicho atributo permanecerá oculto a todos los comandos y será manejado con procedimientos *off-line*. Su única utilidad, es avisarle al sistema que debe bloquear las solicitudes de borrado de los dominios que lo tengan en *verdadero*. Los códigos de respuesta existentes no resultan adecuados para indicar el hecho de que el dominio asociado a la solicitud no puede ser borrado por su calidad de crítico [1], por lo tanto, se agrega un nuevo código de respuesta para responder a los comandos de borrado. Siguiendo el criterio existente para los códigos de respuesta de *EPP*, se define que dicho código sea el 2309, cuya descripción es *Domain is not deletable*.

7.2.3. Implementación

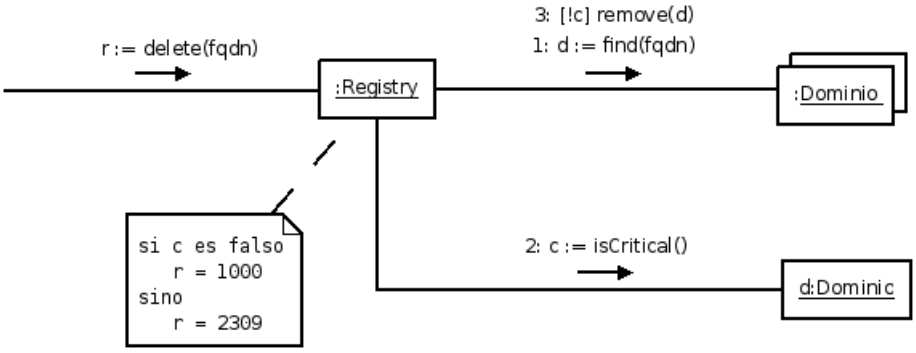


Figura 7.1: Implementación del Manejo de Dominios Críticos

En la Figura 7.1 se aprecian las interacciones necesarias para llevar a cabo el requerimiento. El *registry* recibe la solicitud de borrado y busca el dominio. Si el dominio es crítico, se responde con el código 2309. En caso de que no sea crítico, se procede al borrado del dominio siguiendo el proceso normal, el cual puede llevar al borrado exitoso (respuesta con código 1000) o a algún error debido a las validaciones que se realicen.

7.3. Registro de nombres *IDN*

7.3.1. Problema

El idioma oficial de Uruguay, Español, admite ciertos caracteres especiales que no son contemplados por el código *ASCII* que se utiliza en las transacciones del servicio de *DNS*. Estos caracteres (vocales con acento, eñe y u con diéresis), eventualmente, podrían aparecer en algún nombre de un dominio a registrar en .UY. Por ejemplo, si existiese una organización denominada *"Salven a los Pingüinos"*, ésta podría requerir el nombre *pingüinos.org.uy*. Para posibilitar el registro de dichos nombres, se definen estándares sobre los nombres *IDN*, tal como se describe en la Sección 5.3. Estos estándares definen una codificación de las cadenas *IDN* basada en la notación *punycode*, denominada codificación *ACE*, que sólo posee caracteres *ASCII*, y que por lo tanto, puede ser manejada correctamente por el sistema *DNS* [35] [32].

Los *resolvers* podrán consultar cualquier nombre conteniendo caracteres especiales siempre y cuando, antes de realizar la consulta correspondiente, traduzcan las cadenas a la codificación antedicha.

Existen varios enfoques que atacan este problema, partiendo de la especificación, resolviendo diversas dificultades que surgen en la implementación de

un *registry IDN*. De ellos, el más sencillo deja incambiadas las operaciones de registro, obligando a que los *registrars* envíen los nombres en formato *ACE*, para que al volcarlos a los archivos de zona para el *DNS*, éstos puedan ser resueltos correctamente.

A pesar de no tener que modificar las operaciones de registro, el *registry*, debe indicar los conjuntos de caracteres soportados, y puede bloquear los registros de nombres que contengan caracteres que no están en dichos conjuntos. También puede ser de utilidad, mantener el nombre en su formato original, para posteriores consultas.

Además, se introduce el problema de la similitud, mencionado en la sección 5.6, en las extensiones del *registry* .BR, y previamente analizado en 5.3.

7.3.2. Solución

Extensión del comando *create* aplicado a Nombres de Dominio

Para realizar un registro IDN, el cliente deberá hacerlo utilizando el nombre *ACE* del dominio. Adicionalmente a los datos estándares de *EPP*, se deberá indicar el lenguaje utilizado y el nombre *IDN*. Este agregado busca dejar disponible el nombre *IDN* para futuras consultas, y evitar confusiones en los nombres por los caracteres *no-ASCII*, restringiendo el conjunto de caracteres posibles para el nombre a los usados en el idioma nativo del *registry*.

En el Cuadro 7.1 puede verse una solicitud del registro del nombre *soñadores.net.uy*. Dentro del nodo `<domain:create>`, está el nombre del dominio en notación *ACE*, de la misma manera que lo están los dos *hosts* asociados. La extensión `<idnext:create>` se usa para indicar el nombre *IDN* del dominio registrado y el lenguaje del registro (en este caso, Español de Uruguay). La respuesta es la estándar de *EPP*.

El registro de nombres *IDN* puede fallar por varios motivos:

1. Se está realizando un registro de nombre *IDN* y no se indican los datos para la extensión `<idnext:create>`.

2. El nombre contiene caracteres no soportados por el *registry*.

3. El nombre es similar a otro ya existente en el *registry*.

El conjunto de caracteres soportados y los criterios para la similitud de nombres, están definidos en la Sección 7.3.2. Cuando alguno de estos criterios falla, se debe retornar un código de error adecuado. Para el primer caso, se puede usar el código de error 2003, que indica la ausencia de un parámetro requerido. En el segundo, el código más adecuado es el 2306, utilizado cuando

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <create>
      <domain:create
           xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>xn--soadores-e3a.org.uy</domain:name>
        <domain:period unit="y">2</domain:period>
        <domain:ns>
          <domain:hostObj>
            ns1.xn--soadores-e3a.org.uy
          </domain:hostObj>
          <domain:hostObj>
            ns1.xn--soadores-e3a.org.uy
          </domain:hostObj>
        </domain:ns>
        <domain:registrant>jd1234</domain:registrant>
        <domain:contact type="admin">sh8013</domain:contact>
        <domain:contact type="tech">sh8013</domain:contact>
        <domain:authInfo>
          <domain:pw>2fooBAR</domain:pw>
        </domain:authInfo>
      </domain:create>
    </create>
    <extension>
        <idnext:create
            xmlns:idnext="urn:seciu:params:xml:ns:
            idnext-1.0">
            <idnext:name>soñadores.org.uy</idnext:name>
            <idnext:lang>es_UY</idnext:lang>
        </idnext:create>
    </extension>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.1: Pedido de registro nombre IDN

un valor de un parámetro está sintácticamente bien formado, pero políticas internas del servidor no permiten usarlo. El elemento `<value>` de la respuesta, contendrá el nombre *IDN* que tiene los valores incorrectos. Estos códigos se resumen en el Cuadro 7.2

Código	Descripción	Causa
2003	Required parameter missing	Se está registrando un nombre *IDN* (una cadena *ASCII* que comienza con el prefijo *ACE*) pero no se indica la información extendida
2306	Parameter value policy error	El nombre indicado es sintácticamente correcto pero contiene caracteres no soportados por el registro

Cuadro 7.2: Errores en el registro de un nombre *IDN*

En el tercer caso, lo más adecuado es dejar el registro en estado *pendingCreate* e indicar que el dominio está en conflicto con otro registrado. Luego, mediante procedimientos *off-line*, se deberá determinar si el registro aplica y el dominio puede ser activado. El código de respuesta para la creación de un dominio que entre en conflicto con otro registrado, debe ser 1001, que indica una operación exitosa, pero en estado pendiente. La comunicación de la resolución del conflicto, sea positiva o negativa, será a través de mensajes de servicio.

Extensión de otros comandos aplicados a dominios

Este nuevo contexto que permite el registro de nombres internacionalizados, abre la posibilidad de otra forma para identificar dominios: el propio nombre internacionalizado. Por estos motivos, es deseable que los comandos *EPP* que reciben como parámetro un nombre de dominio, puedan ejecutarse tanto cuando se indica el nombre *ACE*, como cuando indica el nombre internacionalizado. De esta manera, los comandos *info*, *check*, *update*, *renew*, *delete* y *transfer*, deben poder recibir como parámetro el nombre internacionalizado.

En el caso de los comandos de consulta, si ésta es exitosa, se debe retornar la información de registro *IDN* en un nodo extendido. En los Cuadros 7.3 y 7.4 puede verse una transacción *check* exitosa para un nombre *IDN*.

En estos Cuadros, se puede apreciar cómo el parámetro pasado al comando *check* es el nombre internacionalizado. En la respuesta, se responde

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <command>
    <check>
      <domain:check
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:name>soñadores.org.uy</domain:name>
      </domain:check>
    </check>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.3: Solicitud de chequeo de nombre internacionalizado

la disponibilidad del nombre en notación *ACE*, y se entrega un elemento de extensión que contiene el nombre internacionalizado y la etiqueta de lenguaje.

Conjunto de caracteres soportados

El conjunto de caracteres soportados es un subconjunto del alfabeto latino[1]. Dicho subconjunto, incluye los caracteres permitidos en los nombres de dominio de Internet, y agrega los caracteres *no-ASCII* propios del idioma español que se muestran en el Cuadro 7.5, junto a su correspondiente código *UNICODE*.

Similitud de nombres

Un nombre de dominio es similar a otro, si, aplicados los criterios de equivalencia al primero, se obtiene uno igual al segundo. Los criterios de equivalencia se listan en el Cuadro 7.6.

En el Cuadro 7.7 se muestra el algoritmo para determinar la similitud.

7.3.3. Implementación

En la Figura 7.2 se muestra (parcialmente) el modelo de dominio necesario para implementar la extensión que permita registrar nombres *IDN*, evitando los problemas que pueden surgir como resultado del registro de tales nombres.

[1] *Unicode 5.2 Character Code Charts*, http://www.unicode.org/charts/, último acceso abril de 2010

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <domain:chkData
       xmlns:domain="urn:ietf:params:xml:ns:domain-1.0">
        <domain:cd>
          <domain:name avail="1">
            xn--soadores-e3a.org.uy
          </domain:name>
        </domain:cd>
      </domain:chkData>
    </resData>
    <extension>
        <idnext:chkData
           xmlns:idnext="urn:seciu:params:xml:ns:
           idnext-1.0">
           <idnext:aceName>soñadores.org.uy</idnext:name>
           <idnext:lang>es_UY</idnext:lang>
       </idnext:chkData>
    </extension>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54322-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Cuadro 7.4: Respuesta a chequeo de nombre internacionalizado

Código	Caracter	Descripción
U+00C1	Á	A mayúscula con acento agudo
U+00C9	É	E mayúscula con acento agudo
U+00CD	Í	I mayúscula con acento agudo
U+00D1	Ñ	Eñe mayúscula
U+00D3	Ó	O mayúscula con acento agudo
U+00DA	Ú	U mayúscula con acento agudo
U+00DC	Ü	U mayúscula con diéresis
U+00E1	á	A minúscula con acento agudo
U+00E9	é	E minúscula con acento agudo
U+00ED	í	I minúscula con acento agudo
U+00F1	ñ	Eñe minúscula
U+00F3	ó	O minúscula con acento agudo
U+00FA	ú	U minúscula con acento agudo
U+00FC	ü	U minúscula con diéresis

Cuadro 7.5: Caracteres *no-ASCII* soportados por el registro .UY

1. Las vocales acentuadas son equivalentes a su correspondiente sin acentuar.
2. La u con diéresis, es equivalente a la u sin diéresis.
3. La eñe es equivalente a la ene.
4. La eñe es equivalente al dígrafo *nh*
5. La eñe es equivalente al dígrafo *ni*

Cuadro 7.6: Criterios de equivalencia

```
function similar(nombre a, nombre b): bool
    nombre aux = sustituir_criterio(a)
    if(aux == b)
        similar = true
    else
        similar = false
end similar
```

Cuadro 7.7: Algoritmo de similitud

Los *dominios*, almacenan su nombre en el atributo *name* en formato *ACE*, si es que se trata de un nombre internacionalizado. En caso de tratarse de uno de estos nombres, debe tener asociado exactamente un registro *IDN* (*RegIDN*) que contiene el nombre en su formato original y una etiqueta que identifica el idioma de origen del mismo. Si el nombre no fuese internacionalizado, no se asocia registro *IDN* alguno.

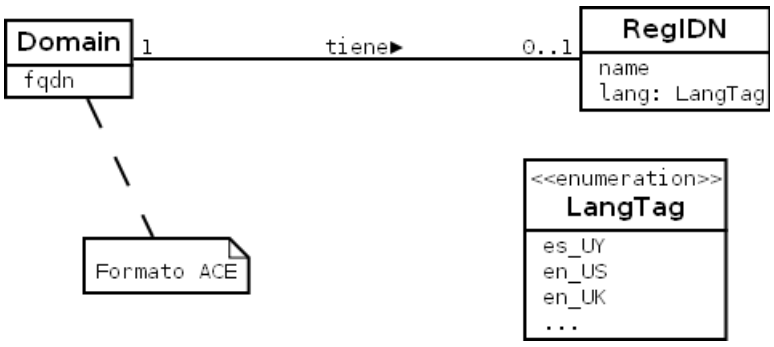


Figura 7.2: Modelo de dominio para registro IDN

Cuando el *registry* recibe un registro de nombre, primero se fija si es un registro *IDN*. Si no lo es, registra el nombre por el procedimiento normal, pero si lo es, comprueba que se esté proporcionando la información de *IDN*. Con dicha información, se valida el registro contra el conjunto de caracteres soportados primero, y contra los nombres ya registrados, según el criterio de similitud (ver Cuadro 7.6).

Si el nombre es soportado por el *registry* y no está en conflicto con ninguno existente, se realiza el registro y se activa el dominio. Si ocurre un conflicto, la operación queda pendiente y se registra el incidente. Si el nombre no es soportado, se rechaza el registro.

El Cuadro 7.8 muestra un pseudocódigo para el registro *IDN*.

```
function Registrar::registro(dominio, reg_idn)
  si nombre no es IDN
    registro = registrar(dominio)
  fin si
  si es IDN
    si reg_idn != null
      si check_charset(dominio, reg_idn)
        dominio_conflicto = check_simil(dominio, reg_idn)
        si dominio_conflicto != null
          insertar_conflicto(dominio, dominio_conflicto)
          registro = registrar(dominio, reg_idn, pendingCreate)
        si no
          registro = registrar(dominio, reg_idn, ok)
        fin si
      si no
        registro = error(2306, reg_idn)
      fin si
    si no
      registro = error(2003)
    fin si
  fin si
end registro
```

Cuadro 7.8: Algoritmo de registro de nombres

7.4. Extensiones *DNSSEC*

7.4.1. Solución

Como se analizó en la Sección 2.3, extendiendo los comandos *info*, *create* y *update*, es posible dar soporte a la provisión y manejo de las extensiones de seguridad para *DNS*, *DNSSEC*. Para poder proveer estas extensiones, es necesario realizar los cambios que se describen a continuación.

7.4.2. Implementación

En la Figura 7.3 se muestran los cambios necesarios al modelo de dominio de *EPP*.

Además de las correspondientes modificaciones al modelo de datos, se deben modificar los *handlers* de los comandos *create*, *update* e *info* para

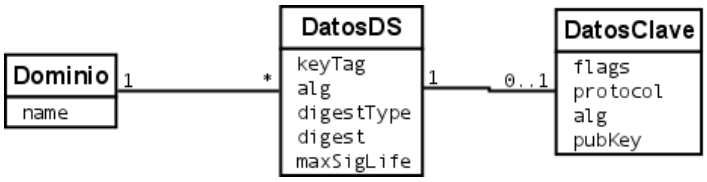


Figura 7.3: Extensión del objeto dominio para la gestión de información *DNSSEC*

insertar, actualizar y mostrar la información *DNSSEC* almacenada.

Por otro lado, se debe considerar que los *registrars* deben contar tanto con un *front-end* que permita el ingreso de registros *DS*, como con una librería capaz de enviar comandos *EPP* extendidos.

Finalmente, el *registry* tiene que contemplar los cambios necesarios para poder publicar las zonas firmadas. Básicamente, estos cambios incluyen la generación periódica y almacenamiento de las claves privadas, así como contar con una infraestructura para el firmado de las zonas.

En la Sección 10.1.2 se realiza un análisis más profundo del impacto de la implementación de estas extensiones.

7.5. Validación de acciones de transformación de objetos del repositorio

7.5.1. Problema

Debido a políticas locales del registro, la creación o modificación (actualización, borrado, etc.) de ciertos dominios se debe retrasar para poder realizar ciertos procedimientos fuera de línea antes de autorizar la acción. Tal es el caso de los dominios, por ejemplo, gubernamentales (*.gub.uy*), para los cuales, antes de dar el *ok* definitivo, se espera una nota oficial que avale el registro del mismo.

Actualmente, para la zona .UY, la definición y ejecución de estas políticas, está dividida entre los dos *registries* existentes: ANTEL para *.com.uy*, y SeCIU para el resto. En un escenario de registro central compartido, como al que se apunta, donde haya un único *registry* para toda la zona, existiendo multitud de *registrars* que lo utilizan, plantea el problema de que cada uno defina su política particular para los dominios que registre.

El problema introducido, debe ser atacado centralizando las políticas de autorizaciones en el registro compartido. Tiene que ser el único *registry* existente, quien defina qué y cómo validar, y no cada *registrar* por separado.

Asimismo, la definición de políticas (y su ejecución) debe ser lo suficientemente flexible para poder introducir cambios, y, eventualmente, acceder a terceros que ayuden en este proceso.

7.5.2. Solución

Incorporación del proceso de validación al Registro

La solución, parte de incorporar el proceso de validación al *registry*. Para esto, es necesario que cada dominio, además de su estado definido por el protocolo *EPP*, tenga asociado un estado que indique, en caso de existir, el estado de validación en que se encuentra.

Relacionado a cada estado de validación, hay una lista de tareas pendientes a realizar para salir de dicho estado. Estas tareas, pueden ser cosas tales como esperar la carta oficial para los dominios gubernamentales, o realizar una llamada telefónica al titular del dominio para que éste autorice un cambio.

Las tareas se pueden agrupar en las siguientes clases:

Pendiente por documentación. Para realizar la transición al siguiente estado de validación, se está esperando que llegue determinada documentación.

Pendiente por habilitación. Para realizar la transición al siguiente estado de validación, se está esperando que una entidad externa la habilite. Por ejemplo, podría requerirse la habilitación del Banco Central del Uruguay para registrar dominios en un posible dominio de segundo nivel *.fin.uy*, destinado a entidades financieras.

Pendiente por DNS. Por ejemplo, un cambio en los *hosts* que se encargan de responder por el dominio, podría necesitar que se corrobore que los servidores asociados puedan responder por el dominio, como puede suceder con los dominios críticos. Un dominio que está esperando esta verificación, se encuentra *pendiente por DNS*.

Pendiente por verificación de registro DS. En un escenario donde se implementan las extensiones relativas a *DNSSEC*, es necesario realizar algunas verificaciones cuando se actualiza la información relativa a este aspecto. Un dominio que está esperando alguna de estas verificaciones, se encuentra *pendiente por verificación de registro DS*.

Todas las solicitudes de comandos de transformación, deben ser filtradas por el *registry* para evaluar si dicha solicitud necesita autorización previo a ser ejecutada. La tarea de definir si validar o no, debe ser realizada por un

delegado que para responder a esta pregunta puede conectarse a un tercero, dando más flexibilidad para agregar entidades externas que se encarguen o ayuden en el proceso. Por los mismos motivos, la definición de los pasos a seguir (estados, transiciones y tareas pendientes), también deben ser delegada a uno o varios terceros.

Las transiciones entre estados, no pueden ser manejadas mediante *EPP*, ya que éste es un protocolo para gestionar la provisión de objetos y los cambios en éstos. El almacenamiento de los datos de verificación en el registro, es únicamente al efecto de proporcionar, a quienes registran, un seguimiento de sus solicitudes. Las modificaciones de estos datos, corren por cuenta de terceros que tienen disponible una interfaz, distinta a la de *EPP*, para realizarlas. Todos los cambios durante este proceso, deben ser notificados al *registrar* mediante mensajes de servicio, cuya lectura, será responsabilidad exclusiva de éste, utilizando el comando *poll* de *EPP* o cualquier otro mecanismo asincrónico.

Actores del proceso

En el proceso de validación de acciones, se distinguen cuatro actores principales. Estos son:

Registrar es quien realiza una solicitud para transformar un objeto almacenado en el repositorio.

Registry es donde se almacenan los datos de los objetos, incluyendo los datos de validación de acciones. Se vale del *agente legislador* para definir y ejecutar las políticas. Actúa consultando al *agente legislador* sobre los pasos a seguir ante acciones del *agente ejecutor*. Actualiza el estado de validación de los objetos en base a estas consultas. Debe notificar al *registrar* todas las acciones que se realicen por mecanismos distintos a *EPP*.

Agente legislador se encarga de definir cuáles son las acciones a validar, sobre qué objetos se aplican, y cómo son los procesos de validación. Esto implica definir los estados, con su lista de tareas asociadas, y cómo son las transiciones entre éstos. Responde, al *registry*, la pregunta *¿debo validar la acción A sobre el objeto Y?*

Agente ejecutor su función es contestar las tareas asignadas por el *registry* mediante una interfaz que éste define.

Proceso de validación de la solicitud

Durante el proceso de validación, los actores mencionados colaboran de la manera que se muestra en los diagramas que aparecen a continuación.

En la Figura 7.4, el *registry* recibe la solicitud de transformar un objeto ya existente, o de crear uno nuevo, y consulta al *agente legislador* si dicha solicitud puede ser completada inmediatamente o debe ser autorizada. Si no debe ser autorizada, la acción solicitada se lleva a cabo de manera "normal", mientras que si debe ser autorizada, el *agente legislador*, indica al *registry* los pasos a seguir y éste notifica al *registrar* que la acción fue retrasada. Como resultado de esta invocación, cuya acción se verá retrasada, el cliente recibe un *ticket* que le permitirá identificar la transacción para su posterior seguimiento.

El *registry* debe asignar las tareas a realizar a los *agentes ejecutores* adecuados y, de ser necesario, deberá almacenar datos temporales, los cuales serán descartados y/o impactados en el registro una vez resuelta la solicitud.

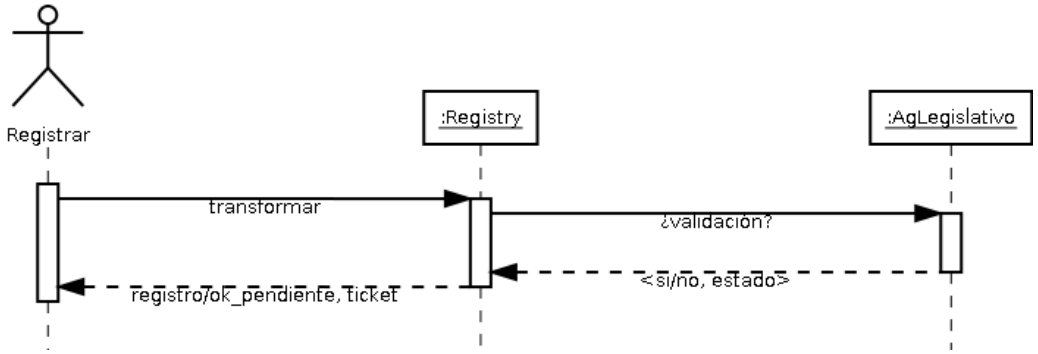


Figura 7.4: Chequeo de una acción de transformación

Cuando el *agente ejecutor* cumple un paso de la verificación, se lo indica al *registry*. Como se muestra en la Figura 7.5, dicha indicación es realizada de manera asíncrona, por lo que, de la llamada, el *agente ejecutor* retorna inmediatamente y puede seguir con otras tareas.

Cuando el *registry* recibe la llamada, chequea si los pasos a cumplir fueron completados. Si fueron completados, consulta al *agente legislador* los pasos a seguir. Si hubieran pasos, se actualiza el estado del objeto. En caso que no restaran pasos, la acción se ejecuta.

El *agente ejecutor*, tiene la responsabilidad de consultar al *registry* qué tareas le fueron asignadas y debe cumplir, como se muestra en la Figura 7.6.

Todos los pasos que modifican el estado de la solicitud del *registrar*, le deben ser notificados mediante mensajes de servicio. Cuando se realiza una

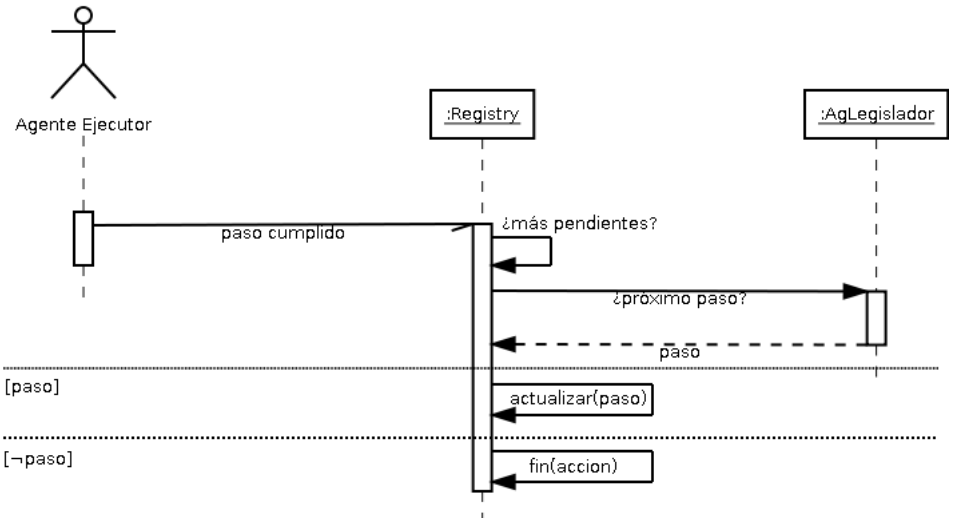


Figura 7.5: El *agente ejecutor* cumple una tarea pendiente.

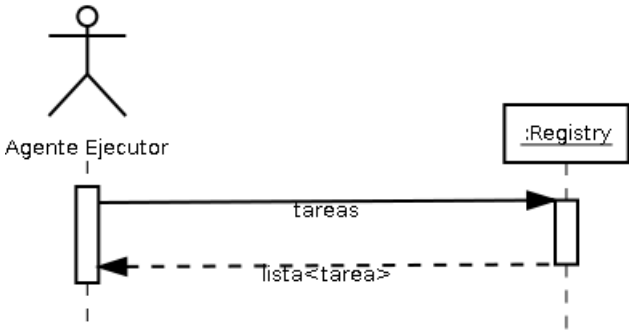


Figura 7.6: El *Agente ejecutor* debe consultar al *registry* sus tareas pendientes.

acción que debe ser retrasada, se debe indicar en la respuesta inmediata a la acción, el retraso y sus motivos. En ambos casos, el *registrar* dispondrá de un *ticket* para saber a qué transacción se refiere un determinado mensaje que le envió el *registry*.

Extensión de los comandos *EPP* para solicitudes de transformación

El proceso de verificación, se desencadena cuando se recibe una solicitud de transformación, como *create* o *update*, para un dominio por parte de un *registrar*. Cuando esto sucede, se realiza una llamada al *agente legislador*, quien, dado el dominio y el tipo de modificación solicitada, determinará si debe ser retrasada o no.

En caso de determinarse que la acción no necesita ser demorada, se procede a la ejecución normal de la misma. Si la solicitud recibida necesita ser autorizada, el *registry* deberá almacenar los datos de la solicitud, y responderle al registrar con el código *OK_PENDING*, en lugar de *OK*, y entregarle un *ticket* para que pueda identificar la transacción durante el proceso. De esta manera el *registrar* se enterará que su solicitud fue retrasada para ser autorizada. El *agente legislador*, también se encargará de indicar el estado inicial de validación, así como la lista de pendientes que deben asignarse al dominio a autorizar, datos que también deberán almacenarse.

En el Cuadro 7.9, puede verse un ejemplo de respuesta a una solicitud que debe ser retrasada. Allí, se muestra sólo la parte que se agrega como extensión.

El nodo `<validation:resData>`, tiene un hijo que contiene el identificador del *ticket* creado. Los nodos `<validation:status>` y `<validation:pendingList>`, muestran el estado de verificación y la lista de pendientes actual de la solicitud identificada por el *ticket*.

Extensión de los comandos *EPP* durante el período de validación

Si se reciben nuevas solicitudes de transformación para un dominio que se encuentra en proceso de validación, el *registry* deberá rechazarlas, indicándole al *registrar* que debe esperar a que termine el proceso en curso antes de poder solicitar una transformación nueva. La razón para rechazarlas, es que en esta nueva solicitud podrían estarse cambiando datos sensibles que, posiblemente, condicionen la aceptación de la solicitud. Además, existe la posibilidad de que algunas etapas de la validación se hubieran realizado con la información original, por lo que quedaría invalidada con la nueva solicitud. En otros casos, la modificación posterior puede depender de la solicitud inicial (por ejemplo un comando *update* luego de un comando *create*), por lo

```
<extension>
  <validation:ticket
    xmlns:validation="urn:seciu:params:xml:ns:
    validation-1.0">
    <validation:idTicket>
      12345678
    </validation:idTicket>
    <validation:status>
      PENDING_STATUS
    </validation:status>
    <validation:pendingList>
      <validation:pending>
        PENDING_1
      </validation:pending>
      <validation:pending>
        PENDING_2
      </validation:pending>
      <validation:pending>
        PENDING_3
      </validation:pending>
    </validation:pendingList>
  </validation:ticket>
</extension>
```

Cuadro 7.9: Respuesta extendida a un comando transformador, que incluye un *ticket* y una lista de pendientes

que, la solicitud original, debe poder ser completada antes de llevar a cabo la nueva transformación. En caso que surjan casos excepcionales, deberán ser manejados de manera manual.

Durante el período de validación, es deseable que el *registrar* pueda realizar un seguimiento de sus solicitudes. Para esto, se debe extender la respuesta del comando *info* de la manera mostrada en el Cuadro 7.9. Allí, se puede ver el identificador del *ticket*, que permite referirse a la transacción que desencadenó el proceso, y los datos relativos al estado de validación actual de la solicitud asociada.

Para informar de los hitos ocurridos durante el proceso de validación, el *registry*, creará mensajes de servicio que muestren dichos cambios al *registrar* interesado. Los mensajes pueden verse en la Sección 7.5.2, y es responsabilidad exclusiva de sus destinatarios, el leerlos y enterarse de la información en ellos contenida.

Extensión de los comandos *EPP* para la resolución de una transformación

Una vez finalizado el proceso de validación, si la resolución fue afirmativa, deben impactarse los cambios en la base de datos, mientras que, si fue denegada, deberán eliminarse los datos temporales. No se realizan extensiones al protocolo en este caso, ya que no existe una comunicación entre el *registry* y el *registrar*. La resolución, es decidida por el *agente ejecutor*. Éste, informa la resolución al *registrar* a través de una interfaz que este último expone a tales efectos. Dicha interfaz, es disjunta a la de *EPP*.

El *registrar*, puede conocer el resultado del proceso de validación leyendo los mensajes de servicio que el *registry* va creando durante el mismo.

Creación de mensajes de servicio

Durante el proceso, es necesaria la creación de varios mensajes de servicio para informar al *registrar* de los cambios en el estado de validación de sus solicitudes, y de la resolución final del proceso. Como se puede leer en las secciones anteriores, cada vez que se cumple un elemento de la lista de pendientes o se produce un cambio de estado, el *registry* debe insertar un nuevo mensaje en la cola de mensajes del *registrar* solicitante, para informar de esta situación.

En el Cuadro 7.10 se muestra el mensaje creado por el servidor cuando se completa un paso de la validación de una solicitud. El nodo `<msgQ:msg>` dice que se ha cumplido alguna tarea de la lista de pendientes. Las tareas cumplidas, se listan en el nodo `<ticket:completed>` de los datos de la respuesta.

Además, se incluyen los demás datos del *ticket*, que permiten identificar a qué transacción está asociado y en qué estado se encuentra (el estado de verificación y la lista de pendientes).

Un mensaje similar se envía cuando la solicitud cambia de estado. El texto del mensaje, debe indicar que se completaron las tareas necesarias para el estado anterior, el nuevo estado y la lista de tareas pendientes.

La notificación de que el proceso de verificación culminó, se realiza con un mensaje como el del Cuadro 7.11. En el mensaje se puede ver la indicación de que el proceso de validación ha culminado. El identificador del *ticket* en los datos de la respuesta, permite identificar la transacción cuya validación ha culminado. Finalmente, el nodo `<validation:approved>` indica si la solicitud fue aprobada (*true*) o denegada (*false*). Durante el proceso de validación, pueden surgir observaciones, las cuales pueden ser indicadas en la lista `<validation:observations>`.

7.5.3. Implementación

La Figura 7.7 muestra las extensiones al modelo de datos, necesarias para incorporar el proceso de validación al registro. Los dominios que están en un proceso de validación, tienen asociado un *ticket* que guarda cuál es la operación retrasada y un número de *ticket* que lo identifica. El estado de validación actual del dominio y la lista de tareas pendientes relativas a este estado, se encuentran asociadas al *ticket*. Cada tarea pendiente debe ser realizada por un agente. La operación que originó el proceso, que es almacenada en el *ticket*, puede tener datos de entrada, los cuales se registran asociados al *ticket* respectivo.

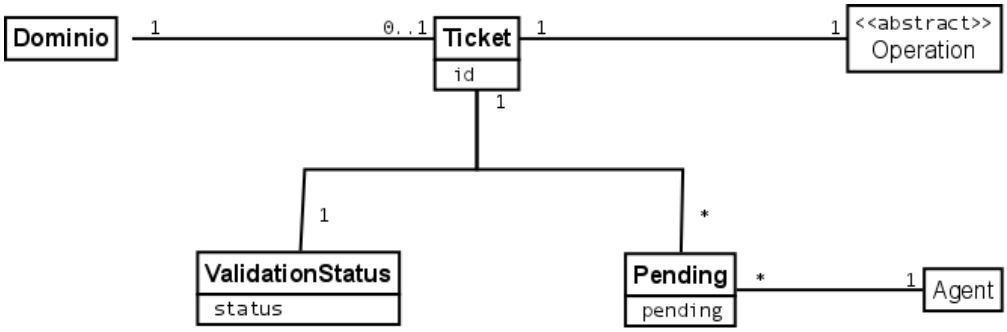


Figura 7.7: Modelo de datos del *registry* extendido para poder completar las tareas de verificación

Como se muestra en la Figura 7.8, el *registry* espera que el *agente legislador* realice la interfaz *LegislationAgent*. La misma, exige a dicho agente

```
<msgQ count="5" id="12345">
  <qDate>2000-06-08T22:00:00.0Z</qDate>
  <msg>Validation task completed for request</msg>
</msgQ>
<resData>
  <validation:ticket
    xmlns:validation="urn:seciu:params:xml:ns:
    validation-1.0">
    <validation:id>12345678</validation:id>
    <validation:completed>
      <validation:pending>
        PENDING_1
      </validation:pending>
    </validation:completed>
    <validation:validationStatus>
      VALIDATION_STATUS
    </validation:validationStatus>
    <validation:pendingList>
      <validation:pending>
        PENDING_1
      </validation:pending>
      <validation:pending>
        PENDING_2
      </validation:pending>
      <validation:pending>
        PENDING_3
      </validation:pending>
    </validation:pendingList>
  </validation:ticket>
</resData>
```

Cuadro 7.10: Mensaje enviado al *registrar* cuando se completa un paso del proceso de verificación de una solicitud (sólo se incluyen los nodos más relevantes

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg>
        Command completed successfully; ack to dequeue
      </msg>
    </result>
    <msgQ count="5" id="12345">
      <qDate>2000-06-08T22:00:00.0Z</qDate>
      <msg>
        Validation proccess completed for request
      </msg>
    </msgQ>
    <resData>
        <validation:ticket
           xmlns:validation="urn:seciu:params:xml:ns:
           validation-1.0">
           <validation:idTicket>
             12345678
           </validation:idTicket>
           <validation:approved>
             BOOL_VALUE
           </validation:approved>
           <validation:observations>
               <validation:observation>
                   Service observation
               </validation:observation>
           </validation:observations>
       </validation:ticket>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Cuadro 7.11: Mensaje enviado al *registrar* cuando culmina el proceso de verificación de una solicitud

que pueda contestar si un dominio debe ser validado, y, en caso afirmativo, debe poder entregar los datos relativos al proceso de validación (estados de validación, tareas pendientes y transiciones). A través de la interfaz *ExecutionAgent*, los *agentes ejecutores* pueden consultar las tareas que le son asignadas, y pueden indicar una resolución para cada una de ellas. La interfaz de *EPP*, a través de la cual los *registrars* se conectan con el *registry*, es totalmente independiente de las interfaces para los agentes, y éstas, a su vez, son independientes entre sí.

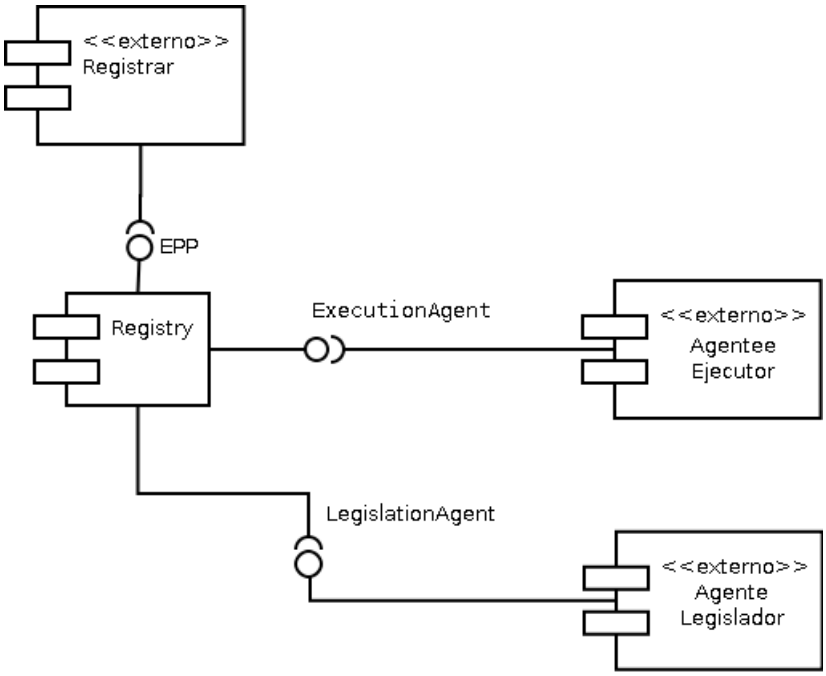


Figura 7.8: Componentes involucrados en el proceso de validación de solicitudes

7.6. Listados de objetos

7.6.1. Problema

El comando *info* de *EPP* permite hacer consultas sobre la información relativa a los objetos de manera individual. No existe forma de obtener reportes o consultas sobre grupos de objetos. Por esta razón se hace necesario extender *EPP* agregando un nuevo comando con este fin. El nuevo comando se denomina *report*.

7.6.2. Solución

Para dar flexibilidad a la solución, se plantea poder realizar consultas de 3 maneras distintas, dependiendo del tipo de búsqueda deseada. Éstas son:

Listado de objetos según condiciones. En este caso de debe especificar que tipo de objetos se desea consultar, y qué condiciones deben cumplir los atributos de los incluidos en la respuesta.

Listado de objetos según consulta nombrada. A partir de una lista de consultas nombradas previamente cargadas, es decir, una consulta y un nombre que la representa, se podrá invocar la consulta especificando el nombre de ésta y el valor para sus argumentos

Listado de objetos según consulta libre. Esta es la opción más flexible de las 3, ya que a partir del modelo conceptual de *EPP* se podrá especificar con completa libertad (condiciones, *joins*, etc.) qué condiciones deben cumplir los objetos a retornar.

En todos los casos se podrá especificar qué atributos de los objetos encontrados se desea obtener.

Un ejemplo de llamada al comando *report* puede verse en el Cuadro 7.12. En él se aprecia la estructura general del *XML* de entrada, la cual contiene el nodo raíz `<epp>`, un nodo `<command>`, y a continuación un nodo `<report>` que identifica al comando. Este último podrá tener dos nodos hijos: el primero de la forma `<report:xxx>` para identificar cual es el tipo de búsqueda seleccionada y, opcionalmente, un nodo `<report:filter>` para limitar qué columnas se incluirán en la respuesta. En caso de no incluirse el nodo `<report:filter>`, la respuesta contendrá todos los atributos definidos para la consulta.

La respuesta podrá contener un código de error acorde a la causa por la que no pudo ser ejecutada la consulta, o el código 1000 para indicar que fue ejecutada exitosamente. En este último caso, contendrá un nodo `<resData>` con los datos de la respuesta.

La respuesta estará organizada en filas y columnas, esto es, dentro del nodo `<resData>`, existirá un nodo hijo `<report:object>` por cada fila, el cual a su vez tendrá un nodo `<report:attribute>` por cada par $< atributo, valor >$ del resultado de la consulta.

A continuación, se describen en profundidad los tres tipos de consulta definidos.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
     epp-1.0.xsd">
  <command>
    <report>
      <!-- Datos específicos de la consulta  -->
      <report:filter>
        <filter:filterCol>FOO_COL</filter:filterCol>
      </report:filter>
    </report>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.12: Ejemplo de solicitud de *report*

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1000">
      <msg>Command completed successfully</msg>
    </result>
    <resData>
      <report:object>
        <report:attribute meta-attribute="value">
          ATTRIBUTE_VALUE
        </report:attribute>
      </report:object>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Cuadro 7.13: Respuesta exitosa a un comando *report*

Listado de objetos según condiciones

Utilizando este modo de consulta se pueden listar contactos, *hosts* o dominios, filtrando por el valor de sus atributos. La estructura general de este comando puede verse en el Cuadro 7.14.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
     epp-1.0.xsd">
  <command>
    <report>
      <report:object>
        <!-- Elementos específicos del objeto -->
      </report:object>
    </report>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.14: Solicitud de reporte de objetos según condiciones

Bajo el nodo `<report>`, se deberá especificar la clase de objeto a buscar (`<report:contact>`, `<report:host>` o `<report:domain>`). A continuación se deberá indicar la lista de los valores de los atributos en el formato `<report:attribute>value</report:attribute>` (por ejemplo, `<report:name>%FOO%</report:name>`). Si se especifica más de un atributo, en el resultado se incluirán aquellos objetos que cumplan con todas las condiciones simultáneamente. La lista de atributos posibles para cada tipo de objeto son los definidos en *EPP*. A modo de ejemplo, en el Cuadro 7.15 puede verse un caso de búsqueda de dominios por nombre.

Listado de objetos según consulta nombrada

Las consultas nombradas permiten ejecutar de manera simple aquellas consultas que se realizan con frecuencia. Previamente, un administrador debe cargar la consulta en el sistema (una sentencia estilo *SQL*) y asignarle un nombre. Luego, los usuarios de *EPP* podrán ejecutarla a partir de su nombre y la lista de argumentos que reciba. Este tipo de consulta exige que el usuario conozca de antemano el nombre de la consulta y los argumentos que recibe, además de su semántica.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
     epp-1.0.xsd">
  <command>
    <report>
      <report:domain
       xmlns:domain="urn:seciu:params:xml:ns:report-1.0"
       xsi:schemaLocation="urn:seciu:params:xml:ns:
       report-1.0.xsd">
        <report:name>%FOO%</report:name>
      </report:domain>
    </report>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.15: Ejemplo de solicitud de reporte de objetos según condiciones para dominios

En el Cuadro 7.16, puede verse una consulta nombrada. Este tipo de consulta se identifica con un nodo `<report:named>` bajo el nodo `<report>`. Este nodo podrá contener dos nodos hijos: `<report:name>` para indicar el nombre de la consulta, y, opcionalmente, `<report:parameters>` con la lista de parámetros que recibe la consulta. Por cada uno de los parámetros se deberá incluir un nodo `<report:parameter>` con dos nodos dentro: `<report:name>` (para indicar el nombre del parámetro) y `<report:value>` (para indicar su valor).

En caso de que no exista una consulta con el nombre indicado, o que la lista de argumentos recibida no coincida con los parámetros esperados, se deberá retornar un código de error adecuado.

Listado de objetos según consulta libre

Las consultas libres son el tipo más flexible de las consultas definidas, y están basadas fuertemente en el concepto de consulta *SQL*. El usuario debe especificar una lista de objetos a consultar (opcionalmente podrá definir alias para cada uno) y las condiciones que deben cumplir, pudiendo especificar condiciones de *join*.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
     epp-1.0.xsd">
  <command>
    <report>
      <report:named
       xmlns:domain="urn:seciu:params:xml:ns:report-1.0"
       xsi:schemaLocation="urn:seciu:params:xml:ns:
       report-1.0.xsd">
        <report:name>FOO_QUERY</report:name>
        <report:parameters>
          <report:parameter>
            <report:name>FOO_NAME</report:name>
            <report:value>FOO_VALUE</report:value>
          </report:parameter>
        </report:parameters>
      </report:named>
    </report>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.16: Solicitud de reporte de tipo consulta nombrada

Los datos de entrada necesarios para este tipo de consulta, son los siguientes:

Entidades Se debe especificar la lista de entidades a consultar, las cuales podrán ser *hosts*, contactos o dominios. En caso de ser necesario, se puede repetir una misma entidad varias veces en el listado a consultar. Opcionalmente se pueden definir alias para las entidades listadas de modo que las condiciones de la consulta sean más legibles. Las entidades se indican bajo el nodo `<report:fromlist>`. Por cada una, deberá existir un nodo `<report:from>`, que a su vez contendrá un `<report:object>` que especifica la entidad, y, opcionalmente, otro nodo `<report:alias>` para indicar un nombre con el cual referirse posteriormente a la entidad.

Condiciones Para la lista de entidades, se deben indicar las condiciones que deben cumplir los objetos a incluir en el resultado. Estas se pueden especificar usando los nombres de las entidades o los alias asignados. Se pueden utilizar los operadores lógicos usuales, como `=`, `<>`, `<`, `<=`, `>`, `>=`, `AND` y `OR`, además de los paréntesis para agrupar. Estas condiciones se deben incluir bajo el nodo `<report:wherelist>`, incluyendo un nodo `<report:where>` por cada una. La condición se indica dentro de un nodo `<report:condition>`.

Campos a devolver La especificación de los campos a devolver se realiza de la misma manera que en los otros tipos de consulta, o sea, en un nodo `<report:filter>`.

Un ejemplo de este tipo de consulta puede verse en el Cuadro 7.17.

7.7. Período de Gracia

7.7.1. Problema

El período de gracia aquí citado, se refiere al período de tiempo existente entre la baja automática de un dominio y la baja definitiva del mismo, durante el cual dicho dominio permanece en estado *pendingDelete* antes de ser borrado definitivamente.

Actualmente, SeCIU maneja este período a nivel de *registrar*. Es decir: el conteo del período de gracia lo realizan las aplicaciones encargadas de "limpiar" la base de datos (ver Sección 3.2.2).

Nuevamente, debido a la posibilidad de introducir nuevos *registrars*, dicho período debe ser manejado por el *registry* para asegurarse el cumplimiento de las políticas centrales que existan al respecto.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="urn:ietf:params:xml:ns:epp-1.0
     epp-1.0.xsd">
  <command>
    <report>
      <report:select
       xmlns:domain="urn:seciu:params:xml:ns:report-1.0"
       xsi:schemaLocation="urn:seciu:params:xml:ns:
       report-1.0.xsd">
        <report:fromlist>
          <report:from>
            <report:object>FOO_OBJ</report:object>
            <report:alias>FOO_ALIAS</report:alias>
          </report:from>
        </report:fromlist>
        <report:wherelist>
          <report:where>
            <report:condition>
              FOO_CONDITION
            </report:condition>
          <report:where>
        </report:wherelist>
      </report:select>
    </report>
    <clTRID>ABC-12345</clTRID>
  </command>
</epp>
```

Cuadro 7.17: Solicitud de reporte de tipo consulta libre

El *registry*, debe encargarse de procesar el conjunto de dominios para ver cuáles tienen causal para la baja automática e indicarlo con el estado *pendingDelete*, y de realizar el borrado una vez que el período de gracia ha expirado.

Asimismo, el *registry* deberá tomar en cuenta el momento del período de gracia en el cual se encuentre el dominio para procesar los comandos, si esto es necesario, y no deberá cargar al sistema de *DNS* dominios que estén pendientes de baja.

Finalmente, todas las acciones llevadas a cabo sobre un dominio por iniciativa del *registry*, deberán ser notificadas al correspondiente *registrar*.

Muchas de las ideas presentadas en esta sección, son extraídas de la *RFC* 3915 [44], relativa al mapeo del período de gracia en el registro de dominios.

7.7.2. Solución

Gestión del Período de Gracia en el Registro

El período de gracia puede ser modelado con la máquina de estados de la Figura 7.9.

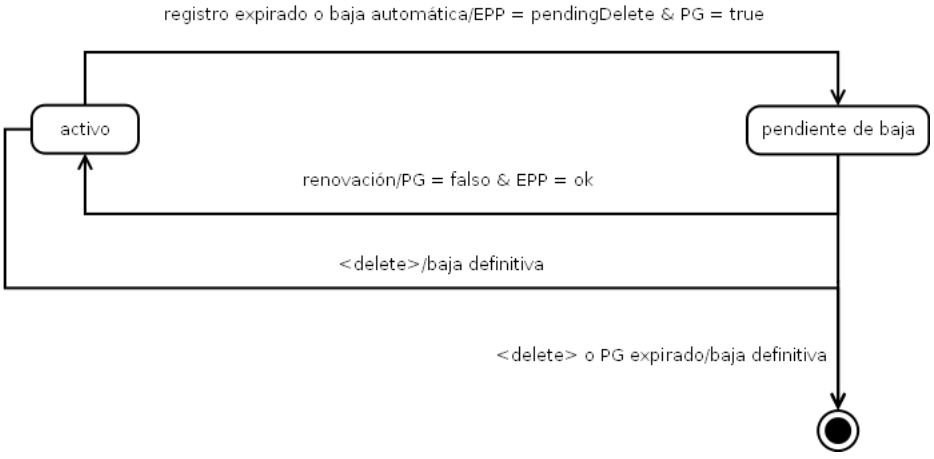


Figura 7.9: Gestión del Período de Gracia

Cualquier dominio activo, capaz de recibir un comando *delete*, luego de ser dado de baja porque expiró su validez y no se realizó la correspondiente renovación, o porque hubo alguna razón para su baja automática, pasa al estado de *EPP pendingDelete*, y se indica que el dominio está dentro del período de gracia. Durante el mismo, las únicas acciones que se pueden llevar a cabo, son consultas, baja definitiva del dominio, o renovación.

Todas las transiciones entre estos estados, sobre todo la baja definitiva (automática), deben ser notificadas al correspondiente *registrar* mediante mensajes de servicio.

Los dominios que se encuentren en el período de gracia con estado en *pendingDelete*, no deben ser volcados a los correspondientes archivos de zona del *DNS*.

A continuación, se mencionan los comandos que se ven afectados por el Período de Gracia. También se presentan algunos comandos que no son extendidos, porque no se ven afectados, pero que se podría llegar a pensar en su extensión.

Extensión de los comandos *EPP*

El estado de período de gracia de un dominio puede ser consultado a través del comando *info*. Adicionalmente a los datos normales, el *registry* indicará dicho estado mediante una extensión como la que se muestra en el Cuadro 7.18. Si el dominio no estuviese en el período, dicha extensión no se muestra.

```
<extension>
  <gpext:resData
    xmlns:gpext="urn:seciu:params:xml:ns:gpext-1.0">
    <gpext:graceStart>
      2000-06-08T22:00:00.0Z
    </gpext:graceStart>
  </gpext:resData>
</extension>
```

Cuadro 7.18: Extensión para mostrar el período de gracia en el comando `<info>`

El efecto del comando *delete*, siempre es el mismo (la baja definitiva) y no depende de si el dominio está o no dentro del período de gracia. A diferencia de la baja definitiva (automática), estas bajas no generan mensajes de servicio para el *registrar*, ya que es este último el único que puede ejecutar un *delete* sobre un dominio de su propiedad, y va a saber en qué momento está dando de baja el dominio. Con estos argumentos a la vista, no es necesario extender este comando para realizar el manejo del período de gracia.

A pesar de que un dominio se encuentre en el período de gracia, sigue estando registrado, por lo que su nombre no está disponible. Por lo tanto, ya

que el comando *check* está destinado a informar la disponibilidad nombres, no es necesario extenderlo.

El comando *transfer* permite la transferencia de nombres de dominio entre *registrars*. Durante el período de gracia, las únicas acciones de transformación posibles sobre un nombre de dominio son la baja definitiva o la renovación, operaciones que deben ser llevadas a cabo por el mismo *registrar* que patrocina el registro. Por lo tanto, las solicitudes de transferencia de dominios en período de gracia deben ser bloqueadas.

Tampoco, como se menciona más arriba, es posible realizar transformación alguna, salvo baja definitiva o renovación, de un dominio en período de gracia. Los cambios a un dominio tal, deben ser bloqueados.

Para ambos comandos, el código de respuesta estándar 2304, *Object Status Prohibits Operation*, es decir, que el estado del objeto no permite la operación, es adecuado para indicar el bloqueo mencionado. Adicionalmente, se podrá indicar en la misma respuesta, mediante una extensión, que el dominio está en el período de gracia.

Cuando un dominio se encuentra en el período de gracia, puede ser renovado usando el comando *renew*. A nivel de interacción con el cliente, no es necesario realizar ninguna extensión. La implementación del comando, debe agregar el desmarcado del período de gracia.

Creación de mensajes de servicio

En el Cuadro 7.19 se muestra un ejemplo de mensaje recibido por un *registrar* cuando el *registry* dio de baja un dominio porque la fecha de expiración pasó sin que se procesara la renovación.

El texto del mensaje indica que el período de validez de un dominio concluyó. Los datos de la respuesta muestran que el dominio afectado es *ejemplo.com* y que su estado actual es *pendingDelete*, e indican la fecha en que el dominio fue colocado en dicho estado (*pendingDeleteDate*) y la fecha en que termina el Período de Gracia (*expGracePeriod*).

El mensaje mostrado en el Cuadro 7.20 es el que el servicio crea cuando un dominio que se encontraba en el período de gracia, fue descartado automáticamente.

El texto del mensaje dice que el Período de Gracia concluyó y que el dominio fue borrado. Los datos de respuesta indican que el dominio borrado es *ejemplo.com* y la fecha en que se procesó la baja.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg>
        Command completed successfully; ack to dequeue
      </msg>
    </result>
    <msgQ count="5" id="12345">
      <qDate>2000-06-08T22:00:00.0Z</qDate>
      <msg>Validity period expired</msg>
    </msgQ>
    <resData>
      <object:statusData
       xmlns:object="urn:ietf:params:xml:ns:object-1.0">
        <object:name>ejemplo.com</object:name>
        <object:status>pendingDelete</object:status>
        <object:pendingDeleteDate>
          2010-07-08T22:00:00.0Z
        </object:pendingDeleteDate>
        <object:expGracePeriod>
          2010-07-23T22:00:00.0Z
        </object:expGracePeriod>
      </object:statusData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Cuadro 7.19: Mensaje enviado al *registrar* cuando se da la baja automática de un dominio

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<epp xmlns="urn:ietf:params:xml:ns:epp-1.0">
  <response>
    <result code="1301">
      <msg>
        Command completed successfully; ack to dequeue
      </msg>
    </result>
    <msgQ count="5" id="12345">
      <qDate>2000-06-08T22:00:00.0Z</qDate>
      <msg>
        Grace Period expired, domain automatically deleted
      </msg>
    </msgQ>
    <resData>
      <object:deleteData
       xmlns:object="urn:ietf:params:xml:ns:object-1.0">
        <object:name>ejemplo.com</object:name>
        <object:deleted>
          2010-07-23T22:00:00.0Z
        </object:deleted>
      </object:statusData>
    </resData>
    <trID>
      <clTRID>ABC-12345</clTRID>
      <svTRID>54321-XYZ</svTRID>
    </trID>
  </response>
</epp>
```

Cuadro 7.20: Mensaje enviado al *registrar* cuando se da la baja definitiva de un dominio

7.7.3. Implementación

La gestión del Período de Gracia en el registro, requiere que este último monitoree constantemente los dominios registrados para ver cuáles deben entrar al período de gracia y cuáles deben ser dados de baja automáticamente. En la Figura 7.10 se muestra cómo se agrega un demonio[2] que realiza dicha tarea.

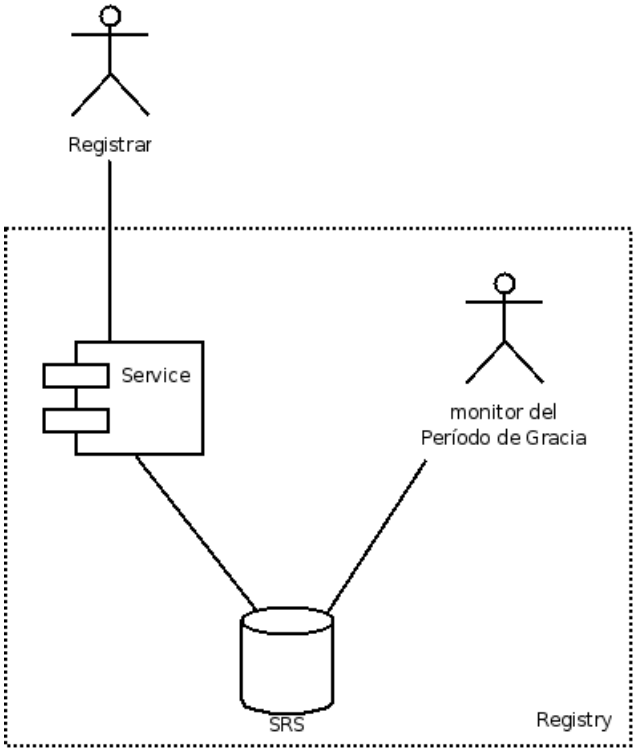


Figura 7.10: Implementación de la gestión del Período de Gracia

El *monitor del Período de Gracia* consulta periódicamente la base de datos del *registry*, buscando dominios no críticos, activos, que no hayan sido renovados antes del fin de su período de validez, o que estén en el período de gracia con estado *pendingDelete*.

Los primeros son colocados dentro del período de gracia con la fecha adecuada, y un mensaje como el visto en la Sección 7.7.2 es encolado para el *registrar* que patrocina el dominio.

[2]un demonio, o *daemon* en inglés, por las siglas de Disk and Execution Monitor, es un proceso informático que se ejecuta en segundo plano a la espera de peticiones, eventos, o haciendo *polling* sobre alguna entidad del sistema

Cuando el monitor detecta que un dominio que está en el período de gracia, pendiente de ser borrado, y que el tiempo transcurrido desde que entró en dicho estado excede al estipulado por *registry*, el dominio es borrado definitivamente, y la notificación correspondiente es creada.

Los mensajes creados por el monitor, deben ser leídos por el *registrar* consultando al *registry*. Es su responsabilidad el leerlos si desea enterarse de los cambios en los dominios que patrocina.

Para indicar que un dominio está dentro del período de gracia, se deben agregar dos atributos al objeto dominio: un *booleano* que indique que está en el período de gracia y una fecha que dice cuando se accedió a dicho período. La fecha coincide con la fecha en que se marca el estado *pendingDelete*, por lo tanto, sólo es necesario el booleano, como se muestra en la figura 7.11.

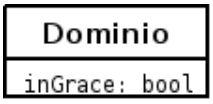


Figura 7.11: Extensión del objeto dominio para la gestión del Período de Gracia

7.8. Auditoría de acciones

7.8.1. Problema

Se desea llevar un registro detallado de las acciones que los *registrars* realizan sobre los objetos que patrocinan, para poder realizar diferentes auditorías. Para esto, el *registry* debe llevar un registro de todas las transformaciones ejecutadas sobre dichos objetos.

7.8.2. Solución

Esta funcionalidad, no requiere modificar el protocolo, ya que basta con que el servidor *EPP* registre cada comando recibido. Por lo tanto, este requerimiento no es estrictamente una extensión, sino una modificación al *software* que actúa como servidor.

La política respecto a cuales objetos deben ser registrados, es decisión del *registry*, por lo que es transparente respecto a la manera de almacenar los datos y podría basarse en factores diversos. Estos factores escapan del alcance de este análisis, estudiándose únicamente los datos a almacenar y la manera de realizarlo.

En la siguiente sección, se presentan las modificaciones necesarias para cumplir el requerimiento.

7.8.3. Implementación

Los datos que interesa almacenar, son los siguientes:

- Fecha de la modificación.
- *Registrar* que la realiza.
- Objeto afectado por la modificación.
- Tipo de acción realizada.
- Argumentos de la acción, valores recibidos.
- Resultado (código *EPP*).
- Mensaje de error, en caso de que corresponda.

Los argumentos de la acción tienen la particularidad de que la estructura de su contenido no es fija, ya que depende de la acción y del tipo de objeto. Esto hace que la cantidad de combinaciones posibles sea alta, lo cual, sumado a que es un dato informativo y no se van a realizar búsquedas a partir de su contenido, indica que lo más adecuado es guardarlos en un único campo de texto libre. Éste, a su vez, debe ser legible para los usuarios, por lo que se propone una estructura de tabla, donde se guarden pares $< argumento, valor >$.

Para almacenar los campos necesarios, se necesita un modelo similar al del Cuadro 7.21.

audit_id	Identificador del registro
date	Fecha de modificación
registrar_id	*registrar* que lo realiza
object_type	Tipo de objeto asociado: dominio, contacto, etc.
object	Identificador del objeto asociado: *FQDN* para dominios, *handle* para contactos, etc.
command	Comando ejecutado: *create, update, delete*
input	Argumentos de la modificación
return_code	Código de retorno *EPP*
return_msg	Mensaje de error asociado a la respuesta

Cuadro 7.21: Posible modelo de datos

Capítulo 8

Repositorio Extendido de Dominios

8.1. Introducción

El *Repositorio Extendido de Dominios (D-REX)*, es un desarrollo realizado sobre la base del sistema *EPP* empleado en SeCIU, *OpenReg*, agregándole extensiones al protocolo. Las extensiones agregadas son varias de las definidas el Capítulo 7. Éstas, introducen determinados procesos asociados que deben ser realizados por fuera de la interfaz *EPP* del sistema. Dichos procesos de gestión, son implementados por una aplicación nueva, denominada *ARCO*: *Administración del Repositorio Central de Openreg.*

El alcance del sistema, abarca las siguientes extensiones:

- *Manejo de Dominios Críticos*
- *Registro de Nombres IDN* Es posible registrar dominios *IDN*, y se chequea similitud cuando se registran. La similitud no es chequeada cuando el nombre no es *IDN*.
- *Extensiones DNSSEC*
- *Validación de Acciones de Transformación de Objetos del Repositorio.* Se implementó la validación del *create* de dominios, utilizando en el proceso, un Agente Legislador que toma, como pasibles de validación, a los dominios militares y gubernamentales.
- *Listados de Objetos del Repositorio.* Es posible realizar listados de dominios según condiciones sobre sus atributos.
- *Auditoría de Acciones*

En lo que sigue, este capítulo se estructura de la siguiente manera:

Modelo de Casos de Uso. Se presentan los actores involucrados y los casos de uso de la aplicación en formato resumido.

Casos de uso críticos. El Modelo de Casos de Uso permite identificar algunos casos de uso que son críticos para la aplicación, debido a los componentes que involucran y a las comunicaciones que se llevan a cabo. En esta sección se presentan diagramas de secuencia que los ilustran.

Modelo de Arquitectura. Se muestra la arquitectura de *software* diseñada para dar soporte a los requerimientos, prestando especial atención a los casos de uso críticos.

Modelo de Implementación. Se detallan los aspectos de implementación del sistema, su despliegue en nodos físicos, y un resumen de las tecnologías utilizadas.

8.2. Modelo de Casos de Uso

8.2.1. Actores

En la Figura 8.1, se pueden ver los actores involucrados en el uso del sistema de registros y las relaciones que existen entre éstos. Los papeles que cada uno cumple se detallan a continuación:

Usuario Web. Es un usuario que accede al sistema por la interfaz gráfica (web) del usuario.

Usuario Administrativo. Es un usuario web capaz de realizar tareas administrativas y de gestión del repositorio central.

Usuario Registrador. Es un usuario web que puede consultar datos sobre su actividad como *registrar* del sistema *EPP*.

Superusuario. Es un usuario web, que posee privilegios tanto de usuario administrativo, como de usuario registrador. En algunos casos, el nivel de privilegio es mayor que el heredado.

Usuario Agente Ejecutor. Representa a un *agente ejecutor*, que contesta tareas sobre la validación de acciones de transformación en los dominios. Accede al sistema a través de una interfaz de *Web Services XML*.

Todos estos actores, son usuarios registrados en el sistema *ARCO*. El único actor que puede registrar nuevos usuarios, es el *superusuario*. Todos los usuarios, incluyendo a los que sean agentes ejecutores, pueden editar la información de perfil que el superusuario les asigna al momento de su registro, accediendo al sistema a través de la interfaz web.

Además, existe una entidad externa, denominada *Agente Legislador*, que se encarga de definir las acciones a validar, y qué pasos se deben cumplir para realizar las validaciones. Éste, a su vez, puede realizar su trabajo de la manera que guste, por ejemplo, delegándolo a otras entidades.

8.2.2. Casos de uso

Usuario Web

La Figura 8.2, presenta el único caso disponible para los usuarios web. El mismo se refiere a la gestión de su perfil como usuario del sistema *ARCO*. Los usuarios son registrados únicamente por el *superusuario* y, por lo tanto, la primer acción disponible en este caso de uso, es la validación de la creación del

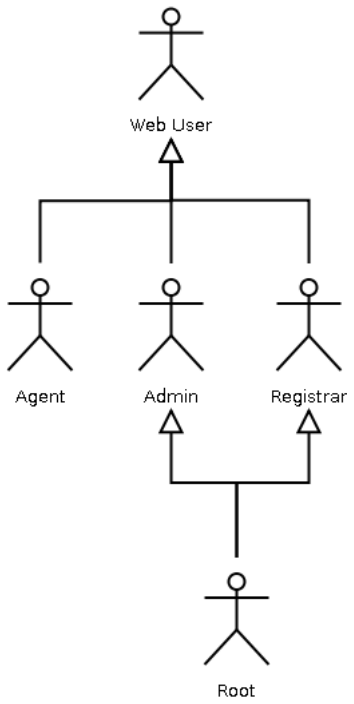


Figura 8.1: Actores de los Casos de Uso

usuario, para evitar abusos y suplantaciones de identidad. Una vez validada la creación del perfil, el usuario podrá gestionar los datos asociados a su perfil, así como cambiar su contraseña de acceso.

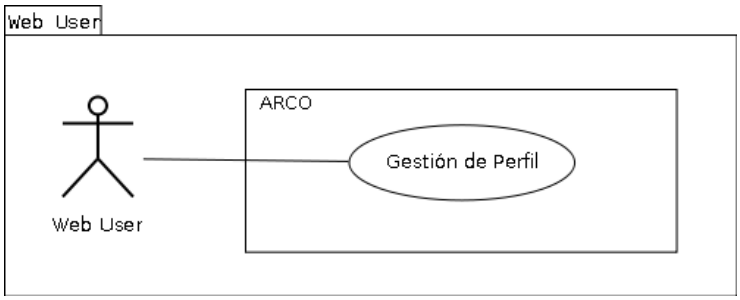


Figura 8.2: Casos de Uso para usuario Web

Usuario Administrativo

Los usuarios administrativos del sitio, son aquellos capacitados para resolver acciones administrativas sobre el repositorio. Dentro de estas acciones,

y para el alcance de este proyecto, se definen dos: la gestión de excepciones por similitud de registro de nombres *IDN*, y la autorización de cambios de los registros de seguridad asociados por el protocolo *DNSSEC*.

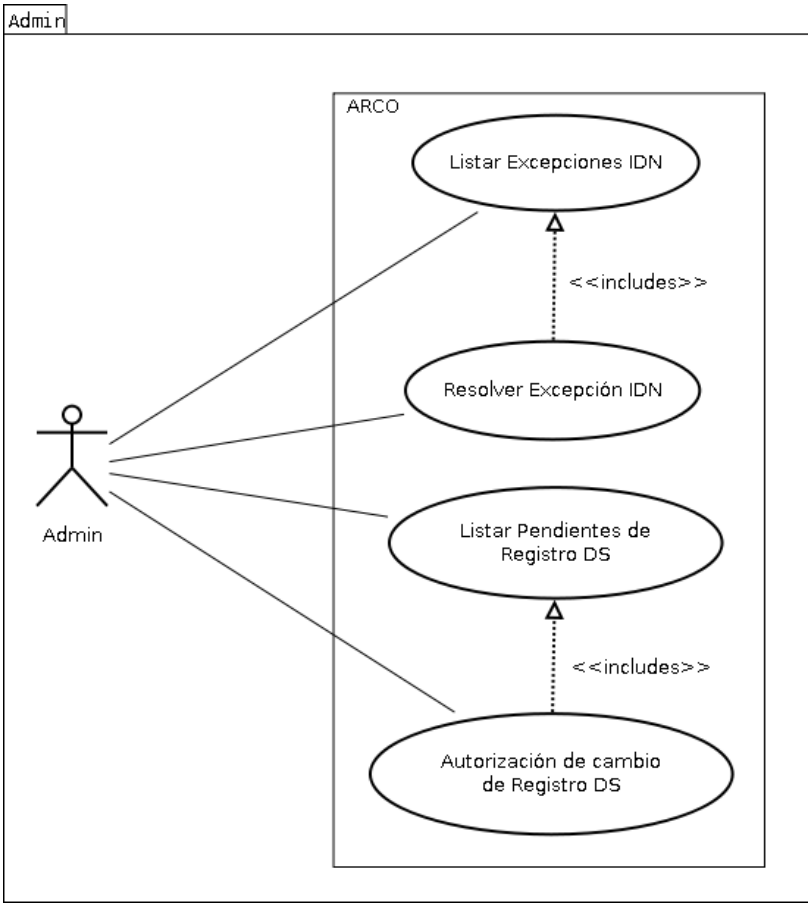


Figura 8.3: Casos de Uso para usuario Administrador

Resolver Excepción *IDN*. El registro de nombres *IDN*, puede traer aparejadas excepciones por el registro de nombres similares (e.j: pinguinos.org.uy y pingüinos.org.uy). Cuando se registra un nombre que es similar a otro que está registrado, el mismo es creado en estado *pendingCreate*, a la espera de que se resuelva si dicho nombre puede ser registrado, sin colisionar con el anterior. Los usuarios administrativos, podrán listar y analizar dichos casos, y tomar una decisión, la cual puede ser autorizar la creación del dominio, o bien, denegarla y borrarlo.

Autorización de cambios en el registro *DS*. El *registro DS*, es un dato incorporado por *DNSSEC* para posibilitar su funcionamiento. Ante un

cambio de clave de una zona hija, el *registry* de la zona padre debe ser modificado. Por motivos de seguridad, dicho cambio debe ser autorizado. Mediante este caso de uso, los usuarios administradores, podrán listar las solicitudes de cambio en el mencionado registro, para primero analizarlas, y luego, autorizarlas o denegarlas. Este caso de uso, no contempla cambios urgentes, los cuales deberán ser manejados por otros mecanismos.

Usuario Registrador

Un *usuario registrador* en *ARCO*, representa a un *registrar* del sistema de registro de dominios, y su vía principal de acceso, es la interfaz *EPP* del mismo. Por lo tanto, el único caso disponible, además de las posibilidades heredadas por ser *usuario web*, es el listado de las acciones que ha realizado a través de dicha interfaz. Los *superusuarios*, pueden ver todo el listado de acciones, mientras que los *registradores* sólo pueden ver las que ellos mismos han realizado.

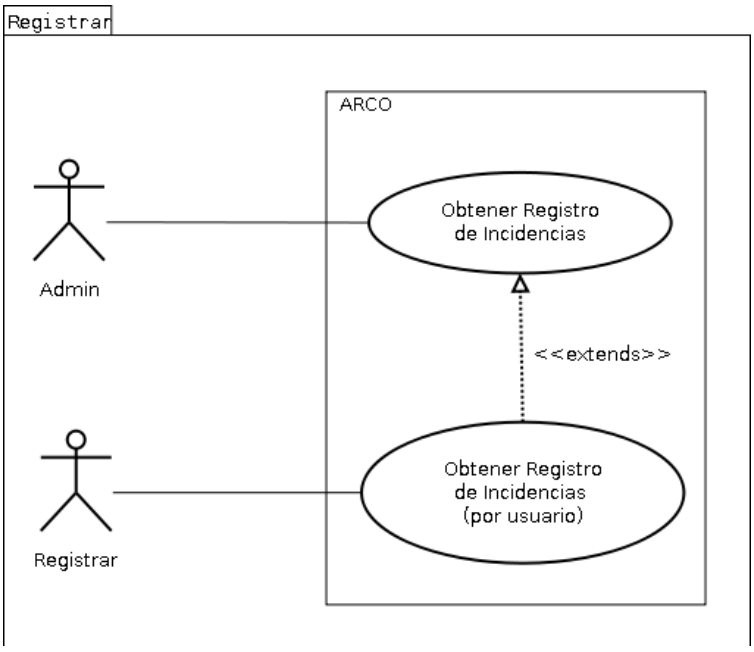


Figura 8.4: Casos de uso para Usuario Registrador

Usuario Agente Ejecutor

Los *agentes ejecutores* son entidades externas a *ARCO*, a quienes les corresponde tomar las decisiones en el proceso de validación de acciones retrasadas. Las acciones que deben ser retrasadas, son definidas por otra entidad externa, el *agente legislador*, quien debe realizar una interfaz definida por *ARCO*. A modo de ejemplo, y a continuación, se presentan los casos de uso involucrados, ejemplificando para el caso en que la acción a retrasar es la creación de un dominio, como puede ser uno gubernamental dentro de .UY.

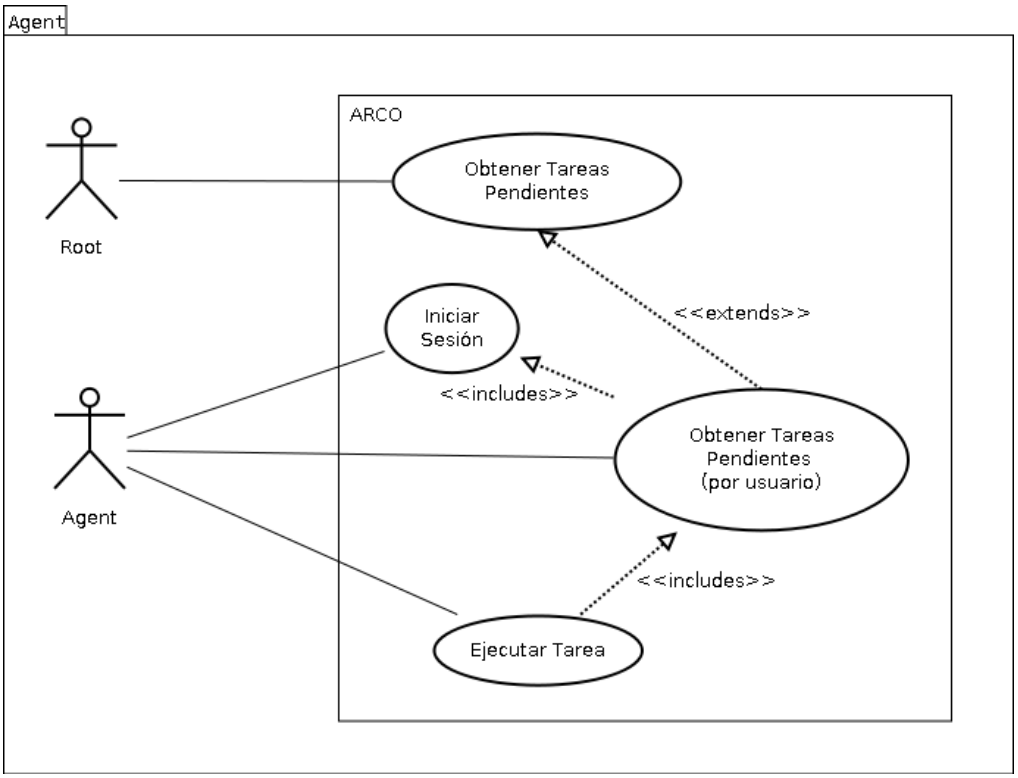


Figura 8.5: Casos de uso para Usuario Ejecutor

Iniciar sesión. El *agente* crea una sesión con sus credenciales para realizar las tareas que le hayan sido asignadas.

Obtener tareas pendientes. Para realizar un tarea, el ejecutor, primero debe conocer qué tareas tiene asignadas. Con este caso, obtiene una lista de dichas tareas.

Ejecutar tarea. Una vez tomada una decisión, el ejecutor lo informa al sistema. Si la contestación es negativa, la solicitud es cancelada, mientras

que si es positiva, se debe actualizar el estado de validación, para lo cual, puede ser necesario contactar al *agente legislador*. Además, en caso de que el proceso de validación haya finalizado autorizando la acción que lo desencadenó, dicha acción, deberá ser hecha efectiva, comunicándose, para esto, con el sistema de registros a través de la interfaz que éste disponga.

Cerrar sesión. Una vez finalizada una sesión de trabajo, elimina los datos temporales de dicha sesión.

Además, los *superusuarios* podrán consultar todas las acciones pendientes que hubieran, pero sin poder incidir en las decisiones de manera directa.

8.3. Casos de uso críticos

Existen dos casos de uso que involucran la comunicación de varios componentes del sistema. En el caso de las excepciones *IDN*, el usuario se comunica con *ARCO*, y éste con *OpenReg*, mientras que en el ciclo de resolución de acciones pendientes, se involucran todos los actores y todos los componentes del sistema, teniendo la dificultad adicional de comunicar *OpenReg* (implementado con *Perl*) con *ARCO* (implementado en *Java*). Por estas razones, aquí se presenta un análisis detallado de dichos casos de uso.

8.3.1. Registro de dominios con conflictos *IDN*

Cuando se crea un dominio que entra en conflicto de nombre con otro ya registrado, *OpenReg* lo deja en estado pendiente, y crea un registro del conflicto, relacionando los dos dominios. Con estos datos, un usuario administrador puede analizar y resolver el conflicto. En la Figura 8.6 se muestra un diagrama de secuencia correspondiente a dicha situación.

Flujo típico del caso de uso

1. Un usuario administrador indica que resolvió una excepción *IDN*

   a) Si la resolución es positiva, en caso de que no existan validaciones pendientes para el dominio, *ARCO* activa el dominio en *OpenReg*. Para activar, se debe quitar el estado *pendingCreate* y agregar el *ok*.

   b) Si es negativa, *ARCO* borra el dominio de *OpenReg*.

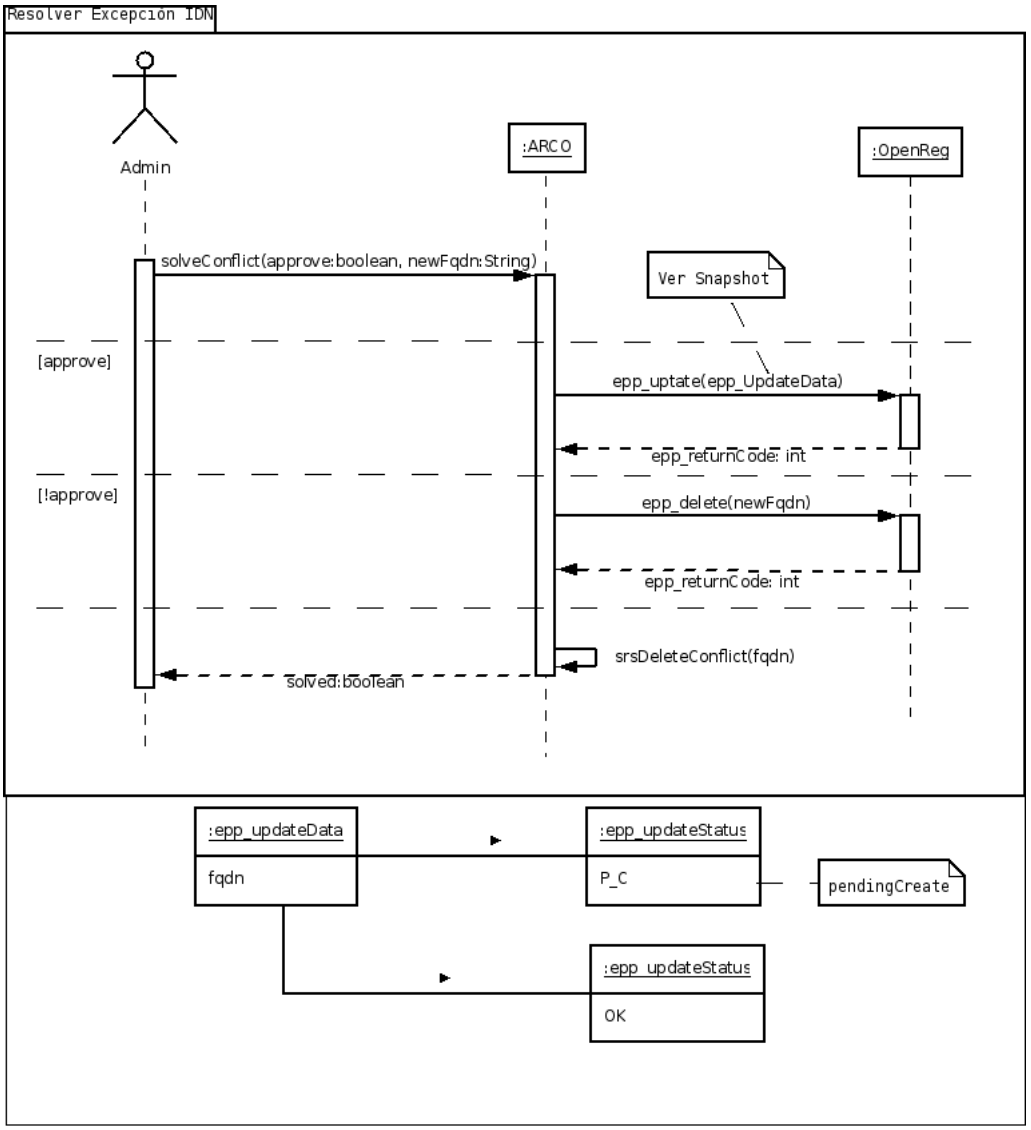


Figura 8.6: Diagrama de secuencia para resolución de conflicto IDN

2. Si se pudo realizar la acción sobre *OpenReg*, se borra el registro del conflicto de la base del registro central.

3. Se crea un mensaje de servicio para el *registrar* indicando la resolución del conflicto.

4. El sistema le indica al administrador si se pudo (o no) llevar a cabo el proceso.

8.3.2. Registrar un dominio que necesita autorización previa

Hay determinados tipos de dominios, que sólo pueden ser registrados por personas autorizadas. Para comprobar que se puede registrar un dominio con tales características, se llevan a cabo *procesos de autorización*, los cuales se detallan más abajo. En este caso de uso, se involucran, no sólo a dos entidades implementadas con tecnologías distintas, sino que también se involucran entidades externas (los agentes ejecutores y legisladores), a las cuales se les delega parte del trabajo para flexibilizar el procedimiento.

En la Figura 8.7 se muestra el inicio del proceso de validación de la creación de un dominio "especial". El mismo ocurre cuando llega un comando *EPP* de registro de un dominio. Cuando dicho comando es recibido, *OpenReg* se comunica con el *agente legislador*, el cual *debe* ofrecer determinada interfaz de *web services XML* para determinar los pasos a seguir en la creación del dominio. Si no se necesita validar, el dominio es creado inmediatamente, pero, si en cambio sí se necesita, el *legislador* le indica cuál es el primer estado de validación del dominio y la lista de tareas pendientes asociadas a dicho estado. Las tareas se representan con una tripla que indica el tipo de pendiente, el nombre del agente ejecutor que lo tiene que realizar, y un mensaje libre. El agente asignado a una tarea, debe estar registrado como usuario *agente ejecutor* en *ARCO*. Con los datos enviados por el *legislador*, *OpenReg* crea el *ticket* correspondiente al proceso de autorización.

La Figura 8.8 muestra un caso de uso muy importante: cuando un *agente ejecutor* le indica al sistema la resolución que tomó en una tarea. Dicha resolución, se puede comunicar, o bien a través de la interfaz web de *ARCO*, o bien utilizando la interfaz de *web services XML* expuesta por éste. Esto último, permite que el agente ejecute el proceso de validación que crea necesario, sin atarse a imposiciones del sistema.

Luego de iniciada una sesión, en el caso de una resolución positiva, se realizan los pasos que se detallan a continuación. En el caso de una resolución negativa, se borran el dominio y todos los datos temporales (ticket y

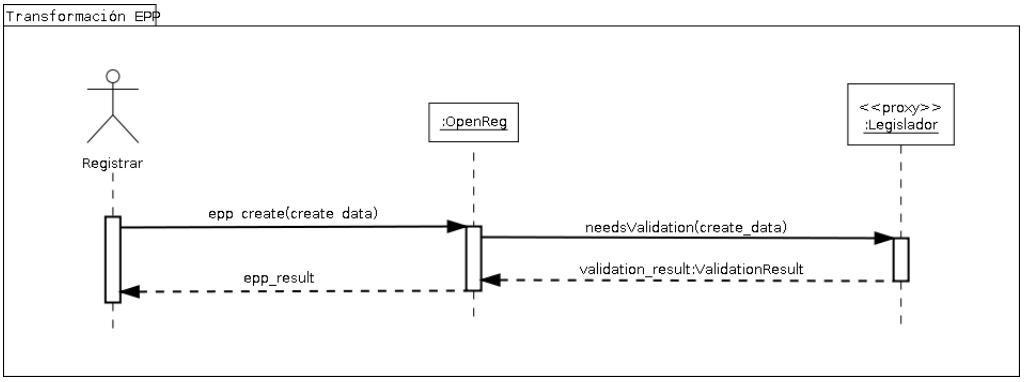


Figura 8.7: Creación del dominio

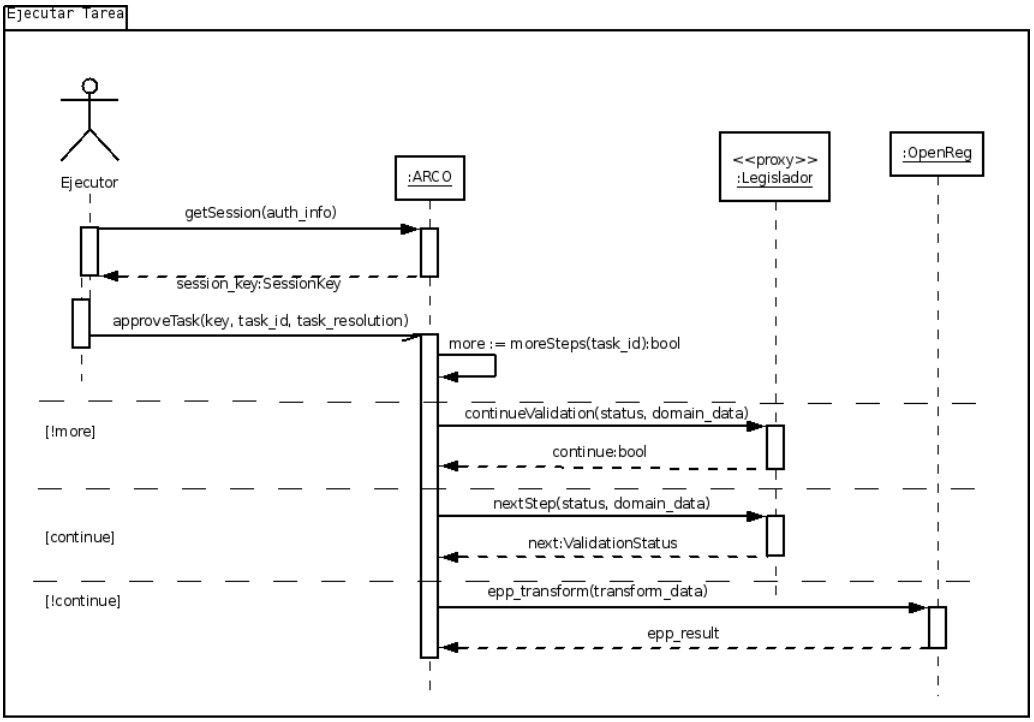


Figura 8.8: Ejecución de una tarea asociada a un ticket

pendientes), y se crea un mensaje de servicio para el *registrar* indicando lo sucedido.

Flujo típico del caso de uso

1. El *agente ejecutor* indica que aprueba la realización de una tarea.
2. *ARCO* comprueba si existen tareas sin cumplir para el estado actual.
3. Si existen tareas sin cumplir, se crea el mensaje de servicio necesario y se retorna.
4. Si no existen tareas sin cumplir
   a) Se consulta al *agente legislador* para ver si se culminó el proceso
   b) Si se culminó el proceso, y de no existir conflictos con el nombre, se activa el dominio y se borran los datos temporales (*ticket*)
   c) Si no se culminó el proceso
      1) Se consulta al *agente legislador* el siguiente paso a completar. Se deben indicar los datos del dominio y el estado de validación previo a la consulta.
      2) Se actualiza el estado de validación del ticket y se agregan los pendientes que correspondan.
   d) Se crea un mensaje de servicio para el *registrar*, indicando el cambio de estado.
   e) Se retorna.

8.3.3. Tipos de datos

Las Figuras 8.9 y 8.10 muestran los tipos de datos utilizados para interactuar entre *ARCO*, *OpenReg*, y el *agente legislador*. En la Figura 8.9, se puede ver que, ante la consulta de si se debe validar (o seguir validando), el *agente legislador* puede contestar de forma afirmativa o negativa. En caso afirmativo, se indica el estado de validación que se debe asignar, y la lista de pendientes a realizar. De cada pendiente, se conoce el tipo de pendiente, el nombre del agente asignado, y un mensaje libre.

El tipo de datos que se muestra en la Figura 8.10, es definido por la biblioteca (*EPP-RTK*) utilizada para comunicar *ARCO* con el registro (*OpenReg*). Los elementos de tipo *epp_UpdateData*, se utilizan para indicar los datos actualizados en una acción de *epp_Update*. Se indican el nombre del dominio

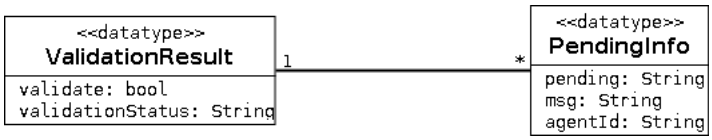


Figura 8.9: Tipos de datos de *ARCO*

a actualizar (*name*) y las listas de elementos a agregar (*add*), quitar (*rem*) o modificar (*chg*). En el caso de modificación de estados, como es el de activar un dominio, estos elementos son de tipo *epp_Status*, el cual indica el estado a agregar, quitar o modificar.

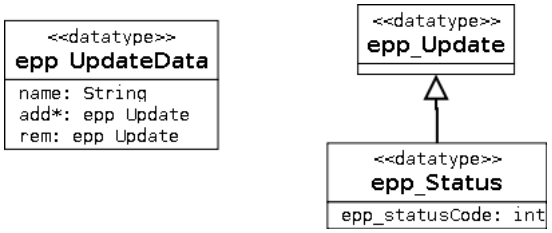


Figura 8.10: Tipos de datos de *EPP-RTK*

8.4. Modelo de Arquitectura

8.4.1. Vista lógica y de componentes

Como lo muestra la Figura 8.11 se estructura en 3 capas, cada una cubriendo un aspecto de la aplicación. Se dispone de una capa de interacción con el usuario (*presentación*), una capa de negocios (*lógica*), y una capa de persistencia. A su vez, la capa de negocio se divide en dos sub-capas: una conteniendo las reglas de negocios y otra que proporciona el acceso a los datos persistidos.

Los componentes de *ARCO* (encuadrados en el rectángulo etiquetado), son componentes de *Java Enterprise Edition* (*JEE*), en su mayoría, *Session Beans*, que permiten distribuir correctamente la aplicación y encapsular las transacciones más complicadas del sistema. Por lo tanto, se comunican utilizando el modelo de invocación remota de método (*RMI, Remote Method Invocation*) de *Java*, pero "oculto" por la plataforma *JEE*.

La sub-capa de reglas de negocio se encarga de implementar los procesos que deben seguir las operaciones del sistema. A tales efectos, debe comunicarse con diversas entidades externas. Una de ellas, es *OpenReg*, con el cual

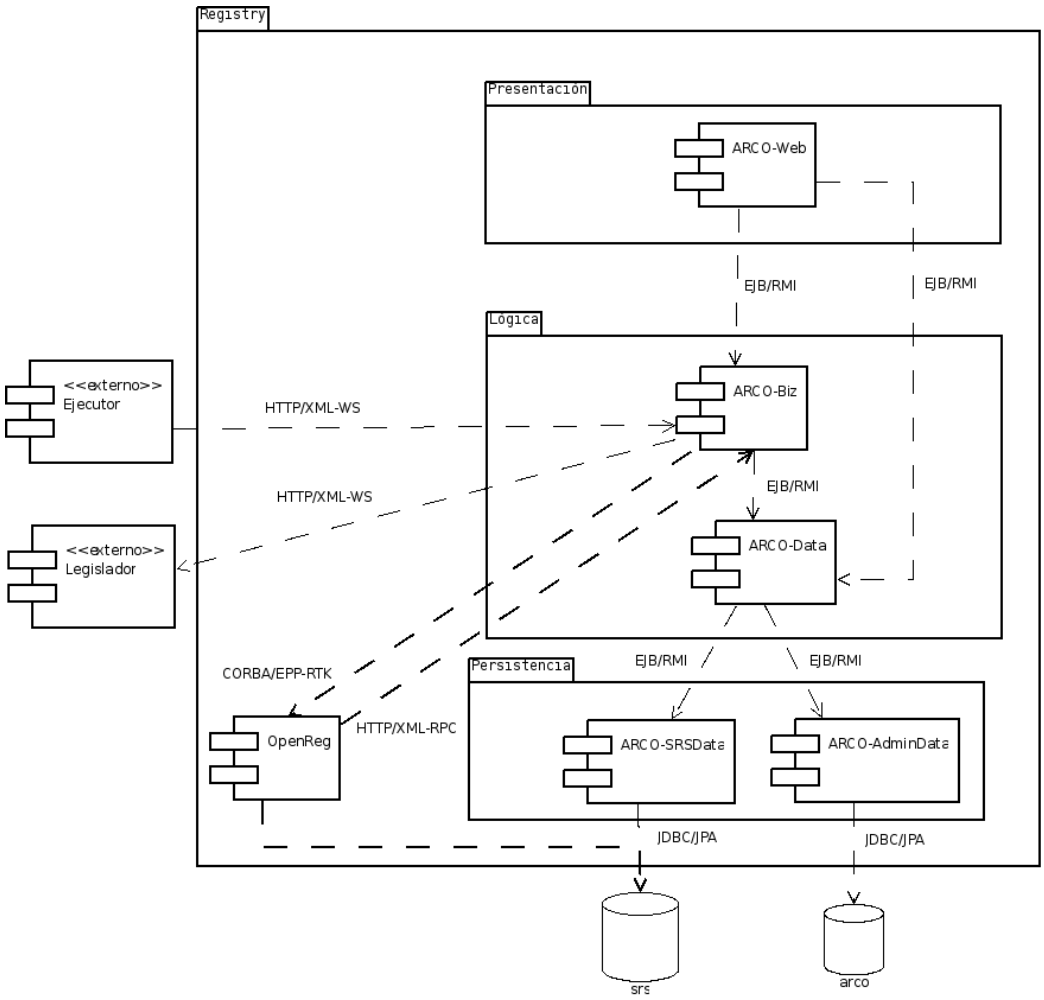


Figura 8.11: Vista lógica y de componentes

se debe comunicar para activar o borrar dominios. Dicha comunicación, se realiza con la biblioteca *EPP-RTK*, que tiene resueltas las dificultades de enviar comandos *EPP* a un puerto *TCP* abierto por un servidor, usando un modelo de objetos *Java*.

A la inversa, la comunicación se realiza con un servicio *XML-RPC* que ofrece *ARCO*. Dicho servicio, es utilizado por *OpenReg* para comunicarse con el *agente legislador*. *ARCO*, espera que dicho agente realice una determinada interfaz de *Web Services XML*, a través de la cual poder consultarlo. Asimismo, *ARCO*, realiza otra interfaz que ofrece a los *agentes ejecutores* para que realicen sus tareas de manera flexible.

Los múltiples *registrars*, se comunican con *OpenReg* a través de su interfaz *EPP*.

8.4.2. Vista de subsistemas

En la Figura 8.12, se pueden ver los subsistemas que componen la subcapa de reglas de negocio, ubicada en la capa lógica de *ARCO*, los cuales se describen a continuación.

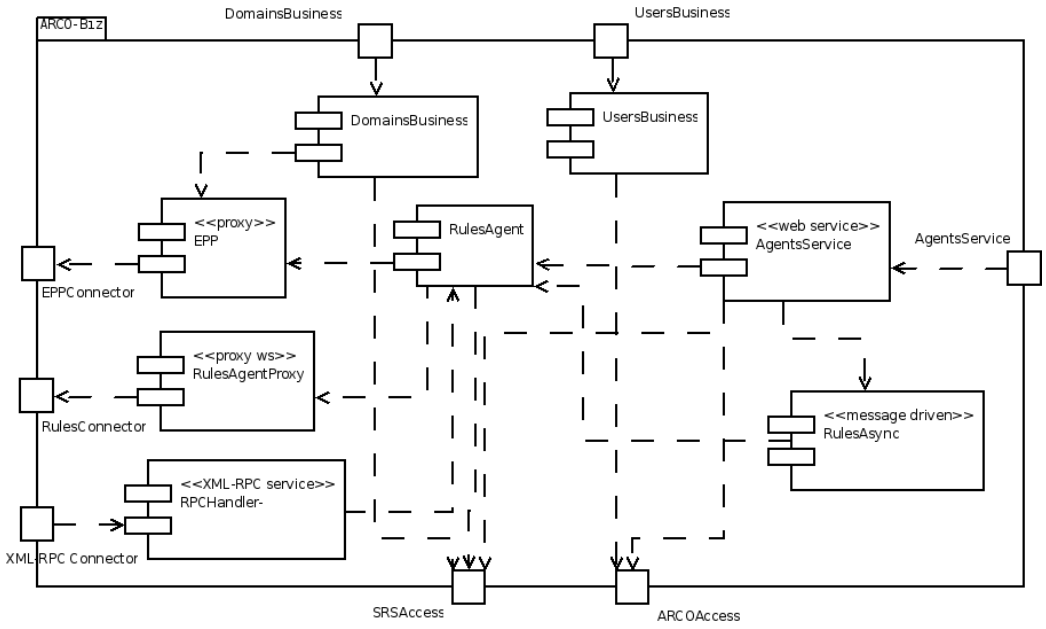


Figura 8.12: Vista de subsistemas para el componente de negocios

DomainsBusiness. Implementa las reglas de negocio relativas a dominios. Esto, incluye transacciones como la resolución de conflictos *IDN*, o la

respuesta de una tarea relativa a un proceso de validación de creación/modificación de un dominio. Debe comunicarse con el registro de dominios a través de su interfaz *EPP*, para lo que utiliza un *proxy*[1]. También debe comunicarse con la capa de persistencia, a través de la capa de datos, más concretamente, accediendo a los datos de la base del *registry*.

UsersBusiness. Realiza las transacciones relativas al alta, baja y modificación de usuarios del sistema. Accede a los datos de los usuarios, almacenados por *ARCO*.

EPP. Implementa el *proxy* de *EPP*. Se encarga de enviar los comandos del protocolo, utilizando la librería *EPP-RTK*.

RulesAgentProxy. Permite la comunicación entre *ARCO* y el servicio web expuesto por el *agente legislador*.

RulesAgent. Contiene las transacciones que implican el chequeo de las reglas de negocio impuestas por el propio sistema, y las impuestas por el *agente legislador*. Accede a los datos del repositorio, y se comunica con el registro a través del *proxy EPP*, y con el agente legislador a través del respectivo *proxy*.

RulesAsync. Las interacciones que se producen cuando se contesta una tarea relativa a algún proceso de validación, implican la comunicación con el *agente legislador*, implementado como un servicio web, lo cual puede implicar demoras que incomoden a los *agentes ejecutores*. El subsistema *RulesAsync*, se encarga de realizar dichas operaciones de manera asíncrona, con lo cual, los agentes no tienen demoras cuando responden una tarea, y el sistema puede realizar dicha tarea cuando tenga disponibilidad para hacerlo. La utilización de asincronismo, implica que los agentes se responsabilicen de consultar el estado de sus operaciones para ver si se concretaron, o si fracasaron y deben ser invocadas nuevamente.

AgentsService. Expone el servicio web para los *agentes ejecutores*. Realiza las consultas necesarias al subsistema *RulesAgent*, e invoca las operaciones sincrónicas del mismo. Cuando un agente contesta una tarea, se encarga de enviar el mensaje a *RulesAsync* para que éste invoque la transacción necesaria en *RulesAgent*.

[1] ver Design Pattern “Proxy”, GoF

8.4.3. Vista de implementación y despliegue

La Figura 8.13 muestra como se implementan y distribuyen físicamente todos los subsistemas que comprende *ARCO*.

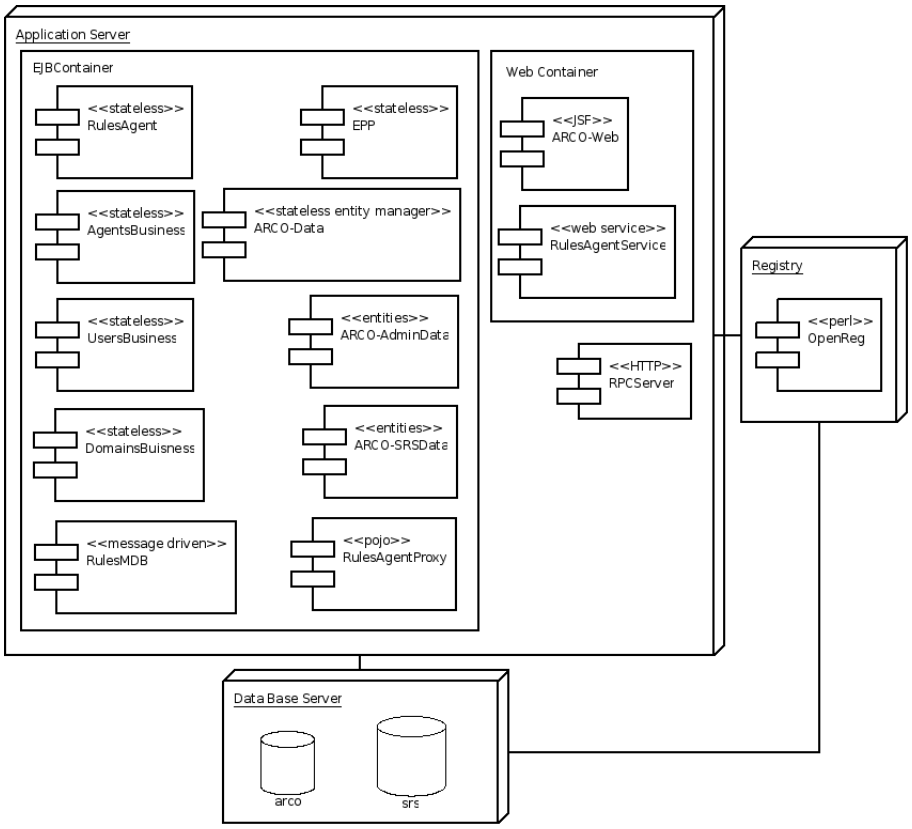


Figura 8.13: Vista de implementación y despliegue

Como puede verse, la aplicación puede distribuirse en tres nodos bien diferenciados: un servidor de aplicaciones conteniendo al sistema *ARCO*, un servidor de registro que aloja al sistema *EPP* (*OpenReg*), y un servidor de bases de datos, que tiene la base del *registry* (*srs*) y la base de *ARCO*. Eventualmente, dichas bases de datos, podrían distribuirse en nodos diferentes.

La plataforma elegida, implica la división del servidor de aplicaciones en un contenedor *EJB* y un contenedor *web*. En el contenedor web, se alojan las páginas que implementan la interfaz de usuario, y el servicio web que implementa la interfaz para los *agentes ejecutores*, mientras que el contenedor *EJB*, aloja las siguientes unidades:

RulesAgent. Ejecuta las transacciones que realizan los agentes.

EPP. En algunos casos, determinadas transacciones *EPP* requeridas por las extensiones, no están contempladas por *EPP-RTK*. En estos casos, dichas transacciones son implementadas por este componente.

AgentsBusiness. Implementa las reglas de negocio relativas a los agentes ejecutores.

UsersBusiness. Implementa las reglas de negocio relativas a los usuarios de *ARCO*.

DomainsBusiness. Implementa las reglas de negocio relativas a los dominios.

RulesMDB. Debido a las demoras que las operaciones que involucra contestar una tarea, la respuesta se hace de manera asíncrona, para que el agente no tenga que esperar innecesariamente. El servicio Web *AgentsService*, envía un mensaje *JMS* a una cola que es monitoreada por *RulesMDB*. Cuando dicho *MDB* lee el mensaje, invoca la operación en *RulesAgent*.

ARCO-AdminData. Objetos (entidades *JPA*) que representan los datos de la base *ARCO*.

ARCO-SRSData. Objetos (más entidades *JPA*) que representan los datos de la base *SRS*.

ARCO-Data. Realiza las operaciones de acceso a datos. Maneja las entidades contenidas en *ARCO-SRSData* y *ARCO-AdminData*.

RulesAgentProxy. Cliente del servicio web ofrecido por el *agente legislador*.

Como ya se mencionó, el servidor de bases de datos aloja las bases que se describen a continuación:

srs. Es la base de *OpenReg*, y en ella se guardan todos los datos (temporales y definitivos) para los objetos almacenados en el repositorio.

arco. Base administrativa para *ARCO*. Realiza el manejo de usuarios y sesiones de la aplicación web.

Para permitir la comunicación entre *OpenReg* y el servicio web del agente legislador, *ARCO* ofrece un servicio implementado con *XML-RPC*, el cual actúa a modo de proxy entre ambos servicios. Éste, se comunica con *ARCO*, que es quien conoce cuál es y dónde está el mencionado agente.

8.5. Modelo de implementación

Los diversos componentes de la aplicación, fueron desarrollados utilizando determinadas tecnologías, que se describen en esta sección.

Para la elección de las mismas, fue fundamental el hecho de que eran tecnologías que los desarrolladores querían explorar. Por otro lado, muchas de ellas ya habían sido atacadas en trabajos anteriores, lo que simplificaba el proceso de toma de contacto con las mismas, recortando, así, la etapa más lenta de su curva de aprendizaje. Además, cabe aclarar que, la implementación de esta aplicación no es el foco de este trabajo y, por lo tanto, no se entendió razonable dedicar un esfuerzo significativo en aprendizaje de tecnologías.

También se detalla cómo están implementadas las extensiones a *EPP*, realizadas sobre *OpenReg*.

Para finalizar, se describen las interfaces de *web services XML* implicadas. La primera, es la que *ARCO* espera que los agentes legisladores realicen, y, la segunda, la que los agentes ejecutores disponen para implementar sus propios procesos de validación de tareas.

8.5.1. Tecnologías empleadas

Comenzando por las capas de interacción con los usuarios, y descendiendo hasta las capas de persistencia, éstas fueron implementadas empleando las tecnologías que se describen a continuación. Para más información, se adjunta a cada una la dirección de un sitio web de referencia.

Java Server Faces La tecnología *JSF*[2] comprende una librería de *tags* para *JSP* que permite desarrollar la parte visual de la aplicación con relativa facilidad, y un conjunto de *APIs* que las representan y permiten manejar su estado, validación de entrada, etc. Define una clara separación entre la lógica de la aplicación y su presentación, y permite "enganchar" ambos aspectos de manera sencilla y directa, sin grandes esfuerzos de desarrollo. *ARCO* utiliza la versión 1.1 en su implementación de referencia, y aprovecha los componentes visuales de *Icefaces* (versiones 1.8.2)[3] para realizar su presentación.

Axis2 Webservices El servicio web para los *agentes ejecutores* está implementado utilizando esta tecnología, que está bien integrada al *IDE*

[2]http://www.oracle.com/technetwork/java/javaee/overview-140548.html
[3]http://www.icefaces.org

empleado (ver más abajo). Mediante *annotations*[4] y configuración de la aplicación web, *Axis*[5] permite desplegar muy fácilmente estos servicios en el contenedor web de un servidor de aplicaciones compatible.

Expat XML Parser *Expat*[6] es una biblioteca de procesamiento *XML* orientada a *streams* y escrita en el lenguaje de programación *ANSI-C*. *Expat* fue uno de los primeros proyectos libres para crear un procesador de *XML*, y ha sido incorporado en varios proyectos libres de relevancia mayor, como el Servidor *HTTP Apache, Python, Perl, PHP*, y la familia de navegadores web *Mozilla*. Tiene un procesamiento orientado a eventos similar al usado en la *API* Sencilla para *XML (SAX)*[7]. Los proyectos que incorporan la biblioteca *Expat*, generalmente implementan procesadores *SAX* y *DOM* por sobre la biblioteca. Ésta es la biblioteca utilizada para la decodificación de los *XML* que recibe el *front-end* de *OpenReg*.

Enterprise Java Beans 3.0 La especificación *EJB*[8] pertenece a la plataforma *Java Enterprise Edition* (*JEE*), y es el componente *server-side* de la plataforma *Java*. Permite un desarrollo rápido y simplificado de aplicaciones empresariales distribuidas, transaccionales, seguras y portables. Las interfaces del sistema y las transacciones de acceso a datos, están implementadas en base a *Stateless Session Beans*, y el componente asincrónico *RulesMDB* es un *Message Driven Bean*, ambos tipos de componentes definidos en la plataforma *JEE*. La implementación utilizada, es la provista por el servidor de aplicaciones.

Java Persistence API Perteneciente, también, a la plataforma *JEE*, *JPA*[9] define una manera para que los objetos se mapeen a una representación relacional, mediante la utilización de *annotations* o configuración, y provee una plataforma para acceder y manejar las entidades almacenadas. Ésta, es la tecnología utilizada para la persistencia y el acceso a datos. La plataforma utilizada es *EclipseLink 1.1* [10].

Java Authentication and Authorization Service Este servicio, fue utilizado para realizar el manejo de usuarios del sistema. Es un *API* que provee a las aplicaciones *Java*, de servicios de autenticación y controles

[4] mecanismo estándar de *Java* para definir propiedades del código fuente
[5] http://ws.apache.org/axis2/
[6] http://expat.sourceforge.net/
[7] http://www.saxproject.org/
[8] http://www.oracle.com/technetwork/java/index-jsp-140203.html
[9] http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html
[10] http://www.eclipse.org/eclipselink/

sobre el nivel de acceso de los usuarios. A partir de su versión 1.4, forma parte de la especificación estándar de *Java* [11].

JBoss Application Server *JBoss*[12] es un servidor de aplicaciones *JEE* de código abierto implementado en *Java* puro. Al estar basado en *Java*, *JBoss* puede ser utilizado en cualquier sistema operativo que posea una *JVM*[13]. Implementa el paquete completo de *JEE*, y, como ya se mencionó, posee una licencia de código abierto sin costo. Además, su amplia difusión hace que exista una comunidad de usuarios y desarrolladores bien establecida, que seguramente ya enfrentó los problemas originados por su uso, y que puede orientar en su solución.

PostgreSQL 8.1 *PostgreSQL*[14] es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia *BSD*. Como muchos otros proyectos de código abierto, el desarrollo de *PostgreSQL* no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales, las cuales trabajan en su desarrollo. Además, su comunidad de usuarios es una gran fuente de ayuda y soporte. Por otro lado, la elección del sistema de bases de datos, se ve restringida por los requerimientos de *OpenReg*, cuya base de datos emplea este servidor.

Eclipse Galileo *Eclipse*[15] es un *Entorno de Desarrollo Integrado* (*IDE*, por sus siglas en inglés) ampliamente difundido en la comunidad de desarrolladores *Java*, de gran potencia y flexibilidad. Posee diferentes distribuciones que están orientadas al desarrollo de diferentes tipos de aplicaciones, pero la amplia base de *plugins* existentes, muchos de ellos en forma gratuita, permite extenderlo en gran medida. La elección de este *IDE*, se basa en el conocimiento previo de los desarrolladores del proyecto y en la muy buena integración que tiene con todas las tecnologías aquí listadas. Además, posee un *plugin* destinado a trabajar con proyectos *Perl*[16]. Se utilizó la versión *Galileo* (2009) porque es la última versión que posee integración con *Icefaces*. Actualmente, se encuentra disponible la versión *Helios* (2010), pero la integración con *Icefaces* no está completamente desarrollada.

[11] http://java.sun.com/products/jaas/
[12] http://www.jboss.org
[13] máquina virtual de *Java*
[14] http://www.postresql.org
[15] http://www.eclipse.org
[16] http://www.epic-ide.org/

8.5.2. Implementación de extensiones

El desarrollo de estas extensiones, se realiza sobre *OpenReg*. A pesar de que existen otras opciones, potencialmente mejores (ver Sección 6.10), se utiliza este servidor debido a que es el empleado actualmente en SeCIU.

Manejo de Dominios Críticos

Esta extensión fue presentada en la Sección 7.2, y consiste en evitar que algunos dominios puedan ser borrados. Para identificar estos dominios, se agregó un nuevo campo llamado *critical* en la tabla dominios de la base de datos *srs*.

Se modifican dos fuentes de *OpenReg*: `EPPResultCode.pm` y `DB.pm`. En el primero están definidas constantes para cada uno de los códigos de error *EPP*, agregándose una nueva constante llamada *EPP_RF_NODELCRITICAL* para el código de error definido. Por otro lado, el segundo fuente es el *handler* donde se implementan las modificaciones a Base de Datos para cada uno de los comandos. Se modifica el método `domain_del`, correspondiente al comando de borrado de dominios, de modo que se retorne el código de error antes mencionado en caso que sea crítico, y continuando con el borrado normal en caso que no lo sea.

Registro de Nombres *IDN*

Para la realización de esta extensión, es necesario crear dos nuevas tablas en la base de datos *srs*, la primera para guardar la información relativa a *IDN* de los dominios (nombre internacionalizado, lenguaje utilizado) y la segunda para registrar aquellos casos de conflictos por similaridad de nombres.

En *OpenReg* es necesario modificar el parser de *XML* del *front-end*, llamado `EPPHandler.pm`, de manera de agregar la decodificación de los nuevos nodos necesarios al momento de crear el dominio *IDN*.

Además, en el *handler* de comandos `DB.pm`, se realizan varios cambios dentro del método de creación de dominios. En caso que el dominio creado sea *IDN*, se inserta un registro en la nueva tabla creada con la referencia al dominio, y la información relativa a *IDN*. Además se realiza la validación de conflictos por similaridad de nombres, para lo cual se realizan las sustituciones definidas en la Sección 7.3.2 y en caso de encontrarse por lo menos un dominio que cumpla las condiciones, se registra este problema insertando un registro en la nueva tabla de conflicto de nombres de dominio.

Finalmente, de haberse detectado este caso, se retorna al cliente *EPP* un código de retorno especial, llamado `OK_PENDING`, para que el *registrar* esté al tanto que la activación del dominio creado tomará más tiempo, ya que debe

pasar por un proceso de validación. Para estos casos, el dominio insertado quedará en estado *pendingCreate* en vez de *ok* de modo que no se publique hasta no completarse su validación. Dicha validación posterior es realizada desde la aplicación *ARCO*.

Como fue explicado al presentar la extensión, se modifica también la respuesta del comando *info*, de modo que en caso de ser un dominio *IDN*, se muestre el nombre internacionalizado y el lenguaje utilizado. Se cambia el *handler* `DB.pm` para que recupere estos datos al invocar el comando *info*, y el fuente `EPPWriter.pm` para agregar los nuevos nodos al *XML* de la respuesta.

Extensiones *DNSSEC*

En esta extensión es necesario guardar datos de los registros *DS* requeridos para el correcto funcionamiento de *DNSSEC*. De manera análoga a las extensiones previas, se crea una nueva tabla en la base de datos *srs* para guardar una referencia al dominio, y los datos de las claves.

Como fue presentado en la Sección 7.4, se modifican los comandos *create* y *update* de dominios para permitir que creen, agreguen, modifiquen y eliminen registros *DS*. Para que esto pueda realizarse, en *OpenReg* se modifica la decodificación *XML* realizada por el módulo *EPP front-end*, para agregar los nuevos campos (*digest*, etc.), y además se codifica dentro del módulo *xaction* para almacenar dichos datos en la base.

Se inserta, además, un *ticket* para realizar la gestión de la validación de los datos ingresados.

Validación de Acciones

La validación de transformaciones se encuentra implementada para el caso de la creación de dominios.

Al recibir un *create* de un dominio, con sus datos se invoca un servicio *XML-RPC* ofrecido por *ARCO*, que se comunica con el agente legislador para decidir si es necesaria la validación de la transformación. En caso de no ser necesaria, se continúa normalmente, mientras que, si se requiere validar, se inserta un *ticket* a partir de la información devuelta por *ARCO* y se deja al dominio en estado *pendingCreate*. *ARCO* es el responsable de la posterior activación de este dominio.

Todos estos cambios se realizan dentro del fuente `DB.pm` del módulo *xaction*.

Listado de Objetos

Para esta extensión, se requiere la creación de un nuevo comando llamado *report*. Dentro del módulo *EPP Front-end* se implementan la decodificación del nuevo comando y la elaboración del *XML* de la respuesta. Por otro lado, en el módulo *xaction* se implementa la búsqueda propiamente dicha.

Auditoría de Acciones

La generación de los *logs* de auditoría, se implementa en el módulo *xaction* que, por cada comando de transformación, invoca una nueva función que inserta los argumentos recibidos y la respuesta del comando. Estos *logs* son guardados en una nueva tabla creada para tal fin, cuyo modelo ya fue planteado en la Sección 7.8, en el Cuadro 7.21.

8.5.3. Implementación de un Agente Legislador

El sistema *D-REX* (*ARCO* + *OpenReg* extendido), delega el trabajo de decidir qué dominios deben ser validados antes de ser registrados, a un agente externo denominado *Agente Legislador*. Dicho agente, puede realizar el proceso que desee para determinar las validaciones a realizar, siempre que implemente, a través de un *web service XML*, la interfaz que se muestra en la Figura 8.14.

Figura 8.14: Interfaz exigida a los Agentes Legisladores

validate Esta operación, recibe los datos del dominio a registrar, y devuelve la información relativa a su validación.

validateNext Recibe la información de un dominio que está en un proceso de validación determinado, y la información relativa a ese estado. Contesta con el siguiente paso a realizar.

El tipo de datos *InfoDomain*, replica la información almacenada por los dominios, incluyendo a sus contactos. El tipo de datos *InfoValidation*, se muestra en la Figura 8.15.

Los significados de cada campo son los siguientes:

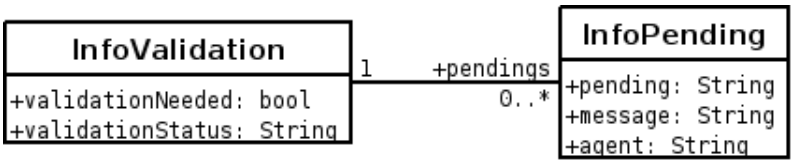


Figura 8.15: Tipo de datos relativo al estado de validación

validationNeeded indica si se debe validar (o seguir validando).

validationStatus si se necesita validar, indica el estado que se debe asignar al dominio. Cuando es utilizado como parámetro, indica el estado actual del dominio.

pending es el tipo de pendiente asociado. Puede ser pendiente por documentación, pendiente por habilitación, pendiente por *DNS* o pendiente por verificación de registro DS (ver Sección 7.5.2).

message mensaje legible que describe el pendiente.

agent identifica al agente ejecutor que debe realizar la tarea. El agente debe estar registrado en *ARCO*.

8.5.4. Implementación de un Agente Ejecutor

Si bien *ARCO* posee la funcionalidad, para los usuarios registrados como agentes ejecutores, de contestar las tareas asociadas a los procesos de validación, la distinción de agente ejecutor surge para flexibilizar este proceso, y permitir que los agentes implementen el proceso que quieran, sin depender de los de *ARCO*.

Para poder implementar dichos procesos, tienen disponible la interfaz de la Figura 8.16, implementada como un *web service XML*, para permitir la mayor flexibilidad, interoperabilidad y disponibilidad posible.

Figura 8.16: Interfaz para agentes ejecutores

Los contratos de dichas operaciones, se resumen a continuación:

createSession dada la información de autenticación del agente, crea una sesión de trabajo. Si la información es válida, retorna los datos de dicha sesión.

listTasks dada la información de una sesión iniciada por un agente, devuelve la lista de tareas asignadas al mismo, en caso de que la sesión indicada sea válida para ese agente.

answerTask recibe la información válida de una sesión, la información de una tarea asignada al agente dueño de la sesión, y una resolución, positiva o negativa, para dicha tarea. Retorna el resultado de efectuar dicha tarea.

closeSession cierra la sesión asociada a la clave indicada.

El canal de comuniación no está encriptado, ni se ofrecen características de seguridad avanzadas, como esquemas de autorización o manejo de sesiones avanzado. En la Sección 10.4, se analizan estos aspectos.

Capítulo 9

Modelo de pruebas

9.1. Introducción

El objetivo del modelo de pruebas que se presenta en este capítulo, es probar la correctitud de los conceptos definidos en el Capítulo 7 y desarrollados en el 8.

El modelo se focaliza en pruebas funcionales, ya que el sistema está pensado como un prototipo funcional que sirva de banco de pruebas para las definiciones realizadas. En tal sentido, las pruebas no funcionales se dejan en segundo plano. A pesar de ello, en caso de que durante la ejecución de las pruebas funcionales se detecten indicios de problemas de, por ejemplo, performance, se registrarán para que sean analizados y corregidos en etapas posteriores que no están en el alcance del proyecto.

Las pruebas se centran en los casos de uso medulares, entendiendo por tales a aquellos que involucran la utilización de la aplicación de repositorio, la cual se encuentra extendida respecto a su implementación original. Además, eligiendo los casos de prueba con este criterio, se cubren todos los casos de uso que son críticos para la arquitectura de *D-REX*, definidos en la Sección 8.3.

Los casos de prueba detectados, son los siguientes:

Creación de dominio especial Los dominios especiales son aquellos que requieren un proceso de validación. Este caso de prueba, sirve para verificar la factibilidad del marco de trabajo propuesto por la extensión relativa a la validación de acciones de transformación en objetos del repositorio, así como la correctitud de lo implementado.

Creación de dominio *IDN* La implementación de la extensión "Registro de nombres *IDN*" introduce los conceptos de conflicto de nombres y similitud. El fin de este caso de prueba, es verificar la correctitud e implementación de dichos conceptos, así como ver el funcionamiento de los comandos *EPP* extendidos.

DNSSEC En este caso, se busca probar el soporte de *D-REX* a las extensiones de seguridad para *DNS*, *DNSSEC*. Una prueba completa requeriría integrar éste sistema a un entorno de nombres firmado. Esto implica la implantación de un sistema de *DNSSEC*, y comprobar que toda la información relativa a éste protocolo (proporcionada por *D-REX)*, es interpretada correctamente por todos los actores involucrados: *resolvers*, *nameservers*, etc.. Dichos aspectos, se alejan mucho del alcance planteado para el proyecto, por lo que las pruebas se focalizarán en los aspectos de provisión de registros *DS* para dar soporte a un entorno *DNSSEC*.

Reportes Con esta prueba, se muestra el funcionamiento del comando *report* agregado a *EPP*. La prueba se refiere a los reportes implementados en *D-REX*, es decir, a la *Consulta Incorporada para Dominios*.

Manejo de Dominios Críticos Este caso de prueba, intenta mostrar que se puede definir la criticidad de un dominio a través de un mecanismo externo, y que los intentos de borrado de dominios críticos son bloqueados y notificados, tal como se definiera en la Sección 7.2.

Logs Se deberá verificar que las acciones ejecutadas por los *registrars*, son almacenadas, como está definido en la Sección 7.8.

Para poder ejecutar estos casos, es necesario contar con el entorno descrito a continuación.

9.2. Entorno de pruebas

El entorno de pruebas debe basarse en las características del entorno de producción utilizado por SeCIU. Se trata de un entorno *GNU/Linux*, usando la distribución *Debian 4.0r8 (Etch)*. Debe tener instalado *PostgreSQL 8.1* como motor de base de datos, la versión 5.8 de *Perl* y el servidor de aplicaciones *JBoss 4.2.3*. A partir de este software base, se instala *D-REX*, o sea *OpenReg* y *ARCO*, con todas las librerías necesarias, las cuales se encuentran detalladas en sus respectivos manuales de instalación (ver anexos). Se debe instalar, además *Preppi*, como cliente *EPP*, para invocación de comandos al servidor, y *EPP Testing Tool* para el envío de lotes de comandos para las pruebas de performance.

9.3. Casos de prueba

9.3.1. Creación de dominio especial

Se debe comprobar que, mientras un dominio común queda en estado activo inmediatamente después de su creación, uno especial queda en estado pendiente, con un *ticket* asignado.

Debe comprobarse, además, que los pendientes asignados puedan resolverse tanto desde la interfaz web, como por la interfaz de *web services*. Si el proceso finalizó todos los pendientes, debe verificarse que el dominio pasa a estado activo. Si un paso fue reprobado, debe probarse que el dominio es borrado

9.3.2. Creación de dominio *IDN*

Se debe verificar que puedan registrarse dominios *IDN*, esto es, conteniendo caracteres *no-ASCII*. Podrán contener únicamente aquellos caracteres soportados por el *registry*, informando con un error al usuario en caso de no cumplirse esta condición.

También es necesario comprobar los casos en que el dominio a ser registrado es similar a otro dominio existente. Se debe verificar que si no existe un conflicto, el nuevo dominio queda activo, mientras que si ocurre un conflicto, éste queda pendiente. En caso de conflicto, debe comprobarse, además, que si el registro del nombre en conflicto es aprobado, el nuevo dominio pasa a activo, mientras que si es denegado, el dominio es eliminado.

Se debe probar la intersección con el caso de prueba anterior, que ocurriría al registrar un dominio especial *IDN*. En particular, deben probarse los diferentes flujos de aprobación/denegación cuando ocurre un conflicto en un dominio especial *IDN*.

Finalmente, debe verificarse que la generación de zona incluye a los dominios *IDN* y que, en los archivos generados, se está utilizando la codificación *ACE* para éstos.

9.3.3. *DNSSEC*

Se debe probar que durante la creación de un dominio, se puede enviar la información de delegación. Por otro lado, se debe verificar que sobre un dominio existente, es posible ejecutar un comando *EPP* agregando, modificando o borrando su información de delegación.

Esta información de validación no queda activa automáticamente, siendo necesario que se apruebe o rechace desde la aplicación *ARCO*. Una vez completada esta validación, debe verificarse que se publican los registros *DS* provistos, en la generación de zona.

Además, debe comprobarse que, al ejecutar un comando *info* sobre un dominio *DNSSEC*, se muestra el detalle de la información de delegación.

9.3.4. Reportes

Se verifica el uso del nuevo comando *EPP report* aplicado a dominios. Para esto se prueban diferentes criterios de búsqueda, basados en combinaciones de los atributos de los dominios. Esta prueba, se combina con el filtrado de campos de la respuesta. En ambos casos, se debe probar utilizando atributos inexistentes y valores inválidos para los existentes.

9.3.5. Dominios Críticos

En este caso, se debe verificar la funcionalidad de la aplicación *ARCO* de marcar y desmarcar dominios como críticos. Para esto, es posible ejecutar varios filtros que permitan encontrar los dominios a marcar (o desmarcar), los cuales deberán ser probados.

Es necesario probar que aquellos dominios no marcados como críticos pueden ser borrados por *EPP*, mientras que los marcados, no es posible borrarlos, y se retorna el código de error esperado. Para esto, se deben ejecutar comandos *delete* de *EPP* contra el servidor *OpenReg*.

9.3.6. Logs

Se verifica que todas las transformaciones, tanto exitosas como fracasadas, a dominios, *hosts* y contactos, queden registradas, y que puedan ser consultadas desde la aplicación *ARCO*, con diferentes criterios de búsqueda.

Los criterios de búsqueda posibles, son filtros sobre los atributos fecha, *registrar*, tipo de objeto, identificador del objeto (sea nombre de dominio, *handle* de contacto, o nombre de *host*) y código de respuesta a la solicitud.

9.3.7. Pruebas de Performance

Debido a que *D-REX* es un prototipo funcional, no es relevante la realización de pruebas de estrés o performance, y sólo se testearán, desde este punto de vista, aquellas funcionalidades donde el desempeño de *OpenReg* pudiera llegar a verse afectado de manera importante.

Se identifica la creación de dominios como la única funcionalidad de interés para ser analizada, ya que se agregó una consulta a un servicio externo (Agente Legislador) para verificar si el dominio debe ser validado o no. En este sentido, es interesante analizar el impacto del cambio comparando los tiempos de ejecución de este comando en un servidor *OpenReg* estándar, y el incluido con *D-REX*.

Estas pruebas, se realizan con la herramienta *EPP Testing Tool*, comentada en la Sección 6.7. La misma, permite ejecutar lotes de comandos *EPP* y extraer estadísticas.

9.4. Conclusiones

El detalles de las pruebas realizadas puede consultarse en el Anexo A. La ejecución de las mismas permitió corregir el software desarrollado hasta que todas fueran correctas.

Además, a partir de los resultados de las pruebas de performance (ver Sección A.25), se obtienen varias conclusiones. Por un lado, el registro de dominios en *D-REX* se encuentra penalizado respecto a *OpenReg* estándar, pasando de un promedio de 0.038 segundos a uno de 0.097 segundos. Esta diferencia implica un aumento del 155 %. A pesar de que la diferencia porcentual sea tan alta, no representa un problema, ya que el nuevo tiempo de registro de un dominio aún permanece dentro de un rango imperceptible para un usuario humano [58].

Con estos tiempos, según los cálculos obtenidos, el máximo *throughput* alcanzable para el entorno de pruebas es de 619 dominios registrados por minuto.

Por otro lado, el aumento en tiempo de ejecución del primer registro de dominio es muy notorio. Este valor aumentó de 0.099s a 1.73s, o sea, 17 veces. La razón para este salto tan grande, es que *D-REX* necesita establecer una conexión a través del servicio *XML-RPC*, lo cual tiene un costo muy importante. Esta situación no es grave ya que ocurre únicamente para la primera transacción.

Otra conclusión que se obtiene del análisis de los datos, es que la diferencia en tiempo de ejecución entre el registro de un dominio militar (más compleja, ya que se inserta un ticket y los pendientes relacionados) y un dominio comercial (más simple, no tiene ticket) es despreciable.

Parte III

Final

Capítulo 10

Conclusiones

10.1. Conclusiones

10.1.1. Sistemas de repositorio

Luego del análisis realizado en el Capítulo 6, y de las pruebas realizadas sobre los diferentes sistemas, se puede concluir que *OpenReg* ha quedado relegado con respecto a otros Sistemas de Registro. La falta de mantenimiento de este *software* es una de las principales razones por las que, día a día, está aumentando su distancia respecto a las otras soluciones analizadas. Una de las premisas en este estudio fue que las soluciones debían de soportar el protocolo *EPP*, con lo que las alternativas fuertes a *OpenReg* se reducen a sólo dos: *CoCCA* y *FRED*. Si se flexibilizara esa condición, *DNRS* sería una tercer alternativa interesante de estudiar.

Al analizar en mayor profundidad *CoCCA* y *FRED*, se encuentra un aspecto fundamental: ambos son técnica y funcionalmente superiores a *OpenReg*. Sin embargo, si se plantea una migración de SeCIU a uno de estos dos sistemas¸ la decisión de cual de los dos utilizar, no debe tomarse a la ligera. Si se desea mantener en la mayor medida posible el modelo de dominio *EPP* que está utilizando actualmente, entonces la decisión debe ser *CoCCA* debido a que en este aspecto no presenta modificaciones respecto a *OpenReg*. Por el contrario, en *FRED* cambia mucho el modelo de *EPP* para los *NSSET's* y sería necesario analizar si a SeCIU le aporta beneficios o dificultades manejar los objetos nameservers y contactos técnicos de la manera *FRED*.

Otras diferencias relevantes son las funcionalidades de *registrar* (recordar que SeCIU también actúa como *registrar*). *CoCCA* tiene más funcionalidades implementadas para los *registrar* en su sitio de administración web que *FRED*, quien se limita a dar un cliente *EPP*. Sin embargo, si comparamos las funcionalidades de *registry*, *FRED* tiene más prestaciones, por ejemplo con su implementación de *DNSSEC*. Finalmente, desde el punto de vista de la licencia del software, *FRED* es realmente *opensource* mientras que en el caso de *CoCCA* su definición es un tanto compleja.

Cabe destacar que *CoCCA* es sustancialmente más simple de instalar y configurar que otras alternativas analizadas, y, al estar desarrollado en *Java*, es más simple mantenerlo, ya que existen muchos más programadores capacitados que en el caso de las otras soluciones, desarrolladas en *Perl*, *Python*, etc [1].

La visión que se quiere transmitir en estos párrafos, es que se debería, al menos, considerar (a mediano o largo plazo) una migración a alguno de estos

[1] una búsqueda en Google con la clave "java developers" arroja (aprox.) 1:760.000 resultados, mientras que con "perl developers", sólo 23.700. En el caso de *Python*, la cifra se aproxima a 50.100

dos sistemas. De lo contrario, SeCIU debería, de alguna manera, generar la masa crítica necesaria para mantener actualizado *OpenReg*, y esto se constituye en un nuevo objetivo para este proyecto: generar conocimientos sobre *OpenReg*, que permitan a SeCIU extender y/o mantener el sistema.

En lo que respecta a librerías para el manejo de *EPP*, se encontró que existe una amplia variedad de librerías y clientes que, por la propia naturaleza de las librerías, la decisión de cuál elegir, depende en primera instancia del lenguaje de programación a utilizar. Si bien la librería más completa existente en la actualidad es *Net::DRI*, está disponible sólo para *Perl*. Si el desarrollo fuera realizado en *Java*, habría que descartar la opción anterior, siendo la mejor opción en este caso *EPP-RTK*.

En lo que respecta a clientes, la decisión también se ve afectada por las variaciones de *EPP*. No es lo mismo un cliente para un servidor *EPP* estándar, que un cliente que necesite implementar determinadas extensiones particulares. A pesar de esto, se identificaron, como de interés, a los clientes *Preppi* y *EPP Testing Tool*. Uno de los aspectos más destacables de *Preppi* es su flexibilidad, aunque, cabe aclarar, está pensado para un entorno de desarrollo o *testing*, y no es un cliente manejable para un entorno de producción. Por otro lado, *EPP Testing Tool* resulta especialmente útil en las pruebas de performance, gracias a su capacidad de enviar lotes de comandos *EPP*. Los resultados de la ejecución de estos lotes se guarda en logs, que posibilitan su análisis posterior.

Para el estudio de estos sistemas, se generaron entornos virtuales que se entregan junto con la documentación del proyecto, que permiten probar los diferentes sistemas sin necesidad de instalarlos.

10.1.2. *DNSSEC*

La puesta en producción de *DNSSEC* en un ambiente de repositorio centralizado *EPP*, es un problema complejo para el cual deben tenerse en consideración ciertos aspectos que ya fueron delineados en la Sección 7.4.2.

En primera instancia, los *registrars* necesitan contar con un *front-end* para el ingreso de claves *DS*, de modo que los responsables técnicos de los dominios tengan un mecanismo para solicitar la actualización de su información de delegación. En el caso de SeCIU, el candidato natural para incluir dicho *front-end* es el *SIGD* (ver Sección 3.2.2), debido a que es el punto de acceso de los *registrants* a la administración de de sus dominios.

Además, para el caso de actualizaciones planificadas, es recomendable contar con un agente que realice una confirmación manual. Sin embargo, cuando se trata de manejar actualizaciones urgentes, por ejemplo las que son debido a que la clave del dominio fue comprometida, contar con un agente

que confirme el cambio es indispensable.

Por otro lado, los *registrars* necesitan enviar esta información al *registry*. Aquí entra en juego *EPP*. Si bien este envío podría realizarse por cualquier mecanismo, es razonable realizarlo por *EPP* para centralizar la comunicación, logrando que las actualizaciones de registros *DS* sean enviadas de manera análoga al resto de las modificaciones de dominios.

Para poder utilizar *EPP* para el envío de esta información, es necesario que los *registrars* tengan la posibilidad de enviar comandos *EPP* extendidos. El *front-end* nombrado anteriormente, *SIGD*, utiliza la librería *EPP-RTK* (ver Sección 6.6.1) para enviar comandos *EPP*. Si bien esta librería contempla la posibilidad de extensiones, en su versión actual no tiene implementadas las extensiones *DNSSEC*, por lo que es necesario que éstas sean desarrolladas.

En lo que respecta al *registry*, necesita contar con un servidor *EPP* extendido para poder recibir y poder procesar las claves *DNSSEC*. Este punto es cubierto por la extensión presentada en el Capítulo 7.4 e implementado por el *Repositorio Extendido de Dominios* (ver Capitulo 8). Además, pueden ser necesarias modificaciones en los *scripts* de generación de zona, ya que los registros *DS* que se encuentran almacenados, deben ser publicados al igual que el resto de tipos de registro.

Como se debe firmar el archivo de zona generado antes de publicarlo, es necesario crear los procedimientos para la generación de las claves a utilizar y sus cambios periódicos, así como planes de contingencia en caso de extravío o si la clave fue comprometida. Este proceso de firmado es un proceso computacionalmente pesado, y además por su criticidad deben tomarse medidas de seguridad importantes, tanto de acceso al servidor de firmado como de almacenamiento de las claves utilizadas.

Finalmente, es necesario configurar los servidores de nombres para que respondan y validen las claves *DNSSEC*. Para este punto es importante estar utilizando versiones de servidores de nombres razonablemente actualizadas, ya que esta característica, por ejemplo, no estaba presente en versiones anteriores de *BIND*.

10.1.3. Repositorio Extendido de Dominios

La ejecución de las pruebas funcionales que se muestran en el Apéndice A, permite afirmar que el funcionamiento de *D-REX* es correcto, ya que luego de corregidos todos los errores detectados en las mismas, se pueden ejecutar, nuevamente, cada una de ellas sin errores. Si bien el sistema es un prototipo, estos resultados avalan el esquema de trabajo planteado mediante la definición de extensiones.

Las penalizaciones de performance obtenidas, respecto a *OpenReg* estándar, son, en todo caso, despreciables o aceptables. El mayor impacto ocurre durante la primer transacción (desde que se levanta el sistema) de creación de dominios, debido a la consulta al Agente Legislador. Luego de ocurrida la misma, la diferencia en el tiempo de respuesta es imperceptible para un usuario humano.

10.2. Conocimientos Adquiridos

El estudio del marco teórico que rodea tanto al *DNS*, como a *EPP*, no sólo permite un conocimiento profundo de ambos, sino que también permite un acercamiento a diversos sistemas de registros de dominios. Las particularidades de éstos, pueden resultar útiles para introducir modificaciones al sistema uruguayo, ya sea para extenderlo o mejorarlo.

Algunos de estos sistemas, así como otras entidades involucradas en el negocio, utilizan *EPP* para interactuar con los *registrars*, y, en algunos casos, realizan extensiones. Estudiar esta utilización, brinda la posibilidad de tomar ideas que permitan adaptar el protocolo a las necesidades del sistema uruguayo.

Asimismo, la implementación del prototipo *D-REX*, incluyendo la implementación de varias extensiones y la gestión de los procesos ligados a las mismas, posibilita un acercamiento bastante profundo a las tecnologías empleadas. Para implementar las extensiones en el servidor *OpenReg*, es necesario dominar *Perl*, fundamentalmente, en lo que a procesamiento de *XML* y acceso a bases de datos se refiere.

Además, la implementación de la herramienta de gestión *ARCO*, permite obtener conocimientos básicos muy importantes sobre diversas tecnologías de la plataforma ofrecida por *Java*. La implementación de las reglas de negocio se realiza con *Enterprise Java Beans*, la persistencia de datos mediante *Java Persistence API*, y la interacción con el usuario se realiza a través de páginas web implementadas con *Java Server Faces* y *Facelets*. Resulta muy interesante, además, la utilización de *Java Authentication and Authorization Service* para implementar los servicios de autenticación y autorización.

Finalmente, cabe destacar que para poder realizar todo lo antedicho, se requiere tener conocimientos de cómo configurar el ambiente de trabajo, que en el caso de este proyecto, consiste en un servidor *Debian* que ofrece servicios de *EPP*, bases de datos a través de *PostgreSQL*, y aplicaciones distribuidas, alojadas en un servidor *JBoss*.

10.3. Evaluación del trabajo

La evaluación del proyecto, es positiva. Los objetivos planteados al inicio, son cubiertos por los puntos tratados en las Partes I y II. La Parte I, proporciona el marco teórico indispensable para entender el contexto en el que se enmarca el proyecto. Además, proporciona lineamientos que sugieren cómo, extendiendo *EPP*, se pueden introducir mejoras en los sistemas de registro de dominios, dando soporte a características particulares de cada uno.

En la Parte II, fundamentalmente, se plantea un esquema de trabajo para adaptar *EPP* a las necesidades de un *registry* en particular. Dicho esquema, es avalado por la definición e implementación de un conjunto de extensiones, que contemplan los aspectos más relevantes del Sistema de Registro de Nombres de Dominio .UY.

Se establece una clara diferencia entre las cosas que deben ser guardadas en el *registry* y las que deben ser obtenidas externamente. El repositorio debe contener sólo instantáneas del estado actual de cada objeto. Esto se constituye en un punto de partida importantísimo cuando de definir extensiones se trata. Una vez identificada la posibilidad de extensión, y validada la utilidad de *EPP* para proveerla, se identifican los datos que se deben almacenar para, por ejemplo, satisfacer los requerimientos de información de los clientes. El resto de los aspectos, debe ser manejado por fuera del sistema de repositorio.

Con la extensión relativa al registro de nombres *IDN*, se amplía el espacio de nombres disponible, y, además, éste se adapta a particularidades del idioma del *registry*, el español.

El trabajo realizado sobre *DNSSEC*, otorga a SeCIU las pautas necesarias para la implantación de tan importante servicio. Estas pautas, incluyen las modificaciones necesarias al protocolo *EPP*, y cómo deben ser implementadas modificando el servidor correspondiente. Más importante aún, resulta el análisis del impacto que esta implantación tendría sobre los sistemas y procedimientos para registro de dominios.

Varias de las extensiones llegaron a ser implementadas (ver Cuadro 10.1. A pesar de que algunas no están completas (y de que otras no se implementaron), sí quedan sus definiciones y un material bastante amplio que muestra cómo se trabajó en este proyecto, lo cual allana el camino para terminarlas (o implementarlas desde el principio, si corresponde).

Cabe destacar la modificación realizada al comando *create* en la extensión relativa a la validación de acciones, a pesar de que la parte implementada sólo cubre una pequeña porción de las posibilidades. Con la implementación de *D-REX*, que realiza el esquema de trabajo planteado al respecto, el cual tiene una complejidad relativamente alta, queda demostrada la factibilidad de dicho marco de trabajo. Esta afirmación, se basa en el hecho de que con este

Extensión	**Estado**
1- Dominios críticos	Implementación completa
2- Registro *IDN*	Implementación completa
3- *DNSSEC*	Implementación completa
4- Validación de acciones	Implementación parcial
5- Listado de objetos	Implementación parcial
6- Período de gracia	No implementado
7- Auditoría de acciones	Implementación completa

Cuadro 10.1: Extensiones realizadas a *EPP*

sistema es posible reproducir los procedimientos que, actualmente, SeCIU realiza de manera manual, en particular el tratado en este párrafo. Respalda esto último, el hecho de los resultados que se obtienen al ejecutar las pruebas planteadas en el Capítulo 9, tal como se muestra en el Apéndice A.

Asimismo, el análisis, la definición, y la implementación de los *logs* de auditoría, plantea cómo hacer para modificar y/o introducir mejoras al servidor utilizado por SeCIU. Además, muestra un ejemplo en el cual extender *EPP* no es adecuado para cumplir el requerimiento.

Otro resultado importante, es el descubrimiento de debilidades en el Sistema de Registros .UY de SeCIU. El *software* de repositorio utilizado no implementa todo el protocolo, define funcionalidades que después no implementa, y está muy desactualizado. Que no existan comandos como el *report* (y que otros estándares, como el *poll* o el *transfer*, no estén implementados), implica repetir algunas informaciones en el *registrar*, con los consiguientes problemas de consistencia.

A pesar de esta evaluación positiva, el proceso a través del cual se llega a la misma, no es el mejor. En primer instancia, se destaca el tiempo empleado en su concreción. Contribuyen a esto, fuertemente, las características del planteo del proyecto. Este planteo, no indica requerimientos específicos, sino que plantea "investigar" las tecnologías involucradas, especialmente *EPP*, para ver qué soporte puede dar éste al Sistema de Registro de Dominios .UY. Un proyecto así, nunca había sido realizado por ninguno de los autores, lo cual dificultó muchísimo la materialización de esta etapa. Esta falta de materialización, hizo que no se tuviera una medida del avance alcanzado, lo que introdujo atrasos y, todavía más importante, redujo la motivación que el proyecto en sí mismo provocaba. También, al no tener materializado este avance, fue casi imposible seguir el cronograma diseñado al inicio. Resulta notable como, al comenzar a producir artefactos (documentos y *software*), la motivación aumentó, y se tuvo un cabal entendimiento de cómo encarar el trabajo.

A partir de ese momento, resultó mucho más fácil realizar y ejecutar una planificación. Se pudieron trazar objetivos claros, marcados por hitos bien definidos, como ser tener decididos los puntos de extensión de *EPP*, tener definidas, validadas y documentadas las extensiones, para luego poder implementarlas.

También se nota la ausencia de un proceso de desarrollo, pero existen varios motivos que la justifican o explican. En primer lugar ya, se mencionó la falta de objetivos bien definidos sobre los cuales desarrollar una planificación, lo que imposibilita la elección o definición de uno, ya que no se sabe exactamente qué es lo que hay que hacer. En los casos en que se definieron extensiones, no se encontró un proceso o una manera de materializar las definiciones, y cuando se consultó a la contraparte académica, la respuesta demoró. Además, dicha respuesta no arrojó demasiada luz respecto al tema. En este tópico, el proceso finalmente seguido, está implícito en el trabajo presentado en el Capítulo 7.

En el cuanto al desarrollo de *D-REX*, si bien no se escogió un proceso de desarrollo específico, si se siguió uno. El mismo, es el mostrado en el Capítulo 8 y es fruto de la experiencia académica y laboral de los desarrolladores. Además, como ya se explicó, este sistema no era el foco principal del proyecto, y por esta razón no se evaluaron diferentes procesos existentes, y se confió en las capacidades de los desarrolladores para implementar el sistema.

La manera en que estas dificultades fueron salvadas, permite afirmar que, más allá de las extensiones definidas, y mucho más allá del sistema implementado, el resultado más importante de este proyecto, es el camino seguido para evaluar y definir extensiones a *EPP* que está implícito en este documento. Seguramente, los siguientes trabajos en este sentido, sobretodo los que realice el propio SeCIU, van a tener un camino mucho menos empinado.

10.4. Trabajo a futuro

10.4.1. Tareas fuera del alcance

Como se explica en capítulos anteriores, la meta principal del proyecto, es definir extensiones al protocolo *EPP* y validar el enfoque tomado en las mismas, así como probar la viabilidad de la metodología empleada. Por tal motivo, existen puntos que fueron excluidos del alcance por entenderse secundarios para los objetivos planteados. Además, existen otros aspectos que no llegaron a ser implementados, ya que los plazos para su realización excederían enormemente el ámbito de este trabajo.

Se destaca la definición formal de las extensiones, la cual está ausente

debido a que su elaboración implica algunas dificultades, fundamentalmente relativas a estándares que habría que estudiar. A pesar de esto, los elementos volcados en el Capítulo 7, son más que suficientes para entender dichas extensiones y poder interactuar con un *registry* que las implemente.

Otro aspecto importante en sistemas como al que apunta *D-REX*, es el de la seguridad de la aplicación. Si bien se implementó un manejo de usuarios, éste es muy básico, y su único objetivo es modelar los actores que aparecen en la definición de las extensiones. Por motivos similares, en la implementación de los servicios ofrecidos a los agentes ejecutores, y el servicio de comunicación entre *OpenReg* y *ARCO*, no se consideraron aspectos de seguridad. En este caso, los servicios buscan implementar el modelo de comunicación que las mismas extensiones plantean.

En cuanto al alcance de la implementación, se priorizaron las extensiones que resultaban más interesantes, teniendo en cuenta las actuales reglas y características del Sistema de Registro de Dominios .UY. En este sentido, no se implementó la extensión relativa al Período de Gracia, y las extensiones sobre Validación de Acciones de Transformación y Reportes, fueron implementadas de manera parcial. Tampoco se implementó el envío de mensajes de servicio para informar los cambios de estado. Uno de los motivos principales, es que *OpenReg* no tiene implementado el comando *EPP* destinado a tales efectos, *poll*. Estos mensajes, podrían ser enviados desarrollando el comando, e implementando los mensajes definidos, o utilizando algún medio alternativo, como el correo electrónico.

Por último, resultaría interesante realizar más pruebas no funcionales, cómo podrían ser pruebas de *stress*, carga o performance. Éstas no fueron realizadas por tratarse *D-REX*, como ya fuera mencionado anteriormente, de un prototipo destinado a probar funcionalidades y flujos de trabajo.

10.4.2. Problemas conocidos

La siguiente lista, son los problemas (*bugs*) identificados, que no llegaron a ser corregidos.

Alta de los agentes. Cuando se da de alta un usuario del tipo Agente Ejecutor en *ARCO*, el mismo debería ser registrado en la base de datos de *OpenReg*. Esto implica que desde una misma transacción, sean accedidas diferentes fuentes de datos, lo cual no es técnicamente trivial, y no pudo ser solucionado.

Implantación de *D-REX*. En el caso de que *D-REX* evolucionara a un sistema para producción, surgirían diversas dificultades para su implantación, debido al ambiente en que actualmente SeCIU realiza el registro

de dominios. El servidor *Tomcat* que aloja la aplicación de registro, no posee un contenedor *EJB* incorporado, necesario para alojar las capas de negocio y persistencia. El proyecto *OpenEJB* de la *Apache Software Foundation*[2], desarrolladores de *Tomcat*, permite incorporarlo, pero a pesar de que *JEE* es un estándar, los diversos *vendors* de *JEE* incorporan algunas modificaciones a sus productos. Esto, hace suponer que la portabilidad de *D-REX* no sea un asunto trivial. Otra alternativa, sería incorporar *JBoss* a este ambiente.

Performance y escalabilidad. Pruebas realizadas utilizando bases de datos con más de 5.000 dominios registrados, permitieron detectar demoras importantes en la obtención de dominios para su listado, ya sea en una búsqueda con condiciones o no. El problema fue atacado mediante la utilización de paginado en la interfaz, pero no se obtuvieron mejoras.

10.4.3. Oportunidades de mejora

Durante la realización del trabajo, fueron identificados algunos puntos que podrían contribuir a su extensión y enriquecimiento. Estos son:

Implementación del comando *poll*. Más allá del envío de los mensajes de servicio aquí planteados, es interesante destacar su utilidad para la comunicación entre el *registry* y los *registrars*.

Portabilidad de *OpenReg*. El hecho de que *OpenReg* no funcione en sistemas operativos actualizados, hace dudar de su uso a futuro debido a la falta de soporte. Esto es crucial en lo que a seguridad se refiere, y por lo tanto debería atacarse el problema. Algunos lineamientos pueden encontrarse en la Sección 6.9.5.

Extensión de *EPP-RTK*. Las extensiones se implementaron a nivel de servidor, y resultaría de gran utilidad disponer de herramientas que resuelvan la comunicación con el *registry*, enviando comandos *EPP* extendidos. En particular, se identifica a la librería *EPP-RTK*, utilizada en el sistema *SIGD* de SeCIU y en *D-REX*, como candidato a ser extendido.

Algoritmo de Similaridad de Nombres. El chequeo de similaridad de nombres, sean *IDN* o no, es un problema computacional complejo, debido a la gran cantidad de combinaciones posibles. En la Sección 7.3.2, se planteó una versión simplificada que chequea únicamente dominios

[2]http://openejb.apache.org/

IDN nuevos, contra los dominios *no-IDN* existentes. La implementación de un algoritmo más completo, requeriría un análisis que escapa a los límites de este proyecto. Este análisis, implicaría una mejora sustancial de la extensión definida.

Dominios para educación pública. En la zona .UY, el registro de dominios para entidades de educación públicas tiene características especiales. Para determinar si un dominio pertenece o no a dicha categoría, actualmente, se siguen procesos manuales idénticos a los requeridos para dominios gubernamentales y militares. Las definiciones respecto a estos dominios, detalladas en la Sección 7.5, son perfectamente aplicables a los dominios .edu de entidades públicas. Para esto es necesario extender el objeto *Dominio* de *EPP* con un atributo que indique si el dominio es público o no. La solicitud de un dominio con estas características, debería generar un proceso de validación.

Bibliografía

[1] S. Hollenbeck. Extensible Provisioning Protocol (EPP). RFC 5730 (Standard), ago. 2009.

[2] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Domain Name Mapping. RFC 5731 (Standard), ago. 2009.

[3] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Host Mapping. RFC 5732 (Standard), ago. 2009.

[4] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Contact Mapping. RFC 5733 (Standard), ago. 2009.

[5] P.V. Mockapetris. Domain names: Concepts and facilities. RFC 882, nov. 1983. Obsoleted by RFCs 1034, 1035, updated by RFC 973.

[6] P.V. Mockapetris. Domain names: Implementation specification. RFC 883, nov. 1983. Obsoleted by RFCs 1034, 1035, updated by RFC 973.

[7] S. Hollenbeck. Guidelines for Extending the Extensible Provisioning Protocol (EPP). RFC 3735 (Informational), mar. 2004.

[8] D. Eastlake 3rd. Domain Name System Security Extensions. RFC 2535 (Proposed Standard), mar. 1999. Obsoleted by RFCs 4033, 4034, 4035, updated by RFCs 2931, 3007, 3008, 3090, 3226, 3445, 3597, 3655, 3658, 3755, 3757, 3845.

[9] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), mar. 2005.

[10] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034 (Proposed Standard), mar. 2005. Updated by RFC 4470.

[11] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035 (Proposed Standard), mar. 2005. Updated by RFC 4470.

[12] Dan Kaminsky. Multiple dns implementations vulnerable to cache poisoning. US-CERT Vulnerability Note VU #800113, jun. 2009.

[13] B. Laurie, G. Sisson, R. Arends, and D. Blacka. DNS Security (DNSSEC) Hashed Authenticated Denial of Existence. RFC 5155 (Proposed Standard), mar. 2008.

[14] P. Vixie. Extension Mechanisms for DNS (EDNS0). RFC 2671 (Proposed Standard), ago. 1999.

[15] M. StJohns. Automated Updates of DNS Security (DNSSEC) Trust Anchors. RFC 5011 (Proposed Standard), sep. 2007.

[16] Cricket Liu and Paul Albitz. *DNS and BIND (5th Edition).* O'Reilly Media, Inc., 2006.

[17] Servicio Central de Informática Universitaria. REGISTRO DE NOMBRES BAJO EL DOMINIO .UY. http://www.rau.edu.uy/servicios/dom/reg-dns.htm. Último acceso Octubre de 2009.

[18] ANTELDATA. CONDICIONES GENERALES DE CONTRATACIÓN DE SERVICIO. https://nic.anteldata.com.uy/dns/common/verCondiciones.jsp. Último acceso Octubre de 2009.

[19] Universidad de la República and ANTEL. Cooperación para la administración y registro de dominios bajo UY. Convenio específico entre ANTEL y la Universidad de la República.

[20] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (Standard), nov. 1987. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

[21] NIC Chile. Reglamentación para el funcionamiento del Registro de Nombres del Dominio CL. http://www.nic.cl/reglamentacion.html. Último acceso Noviembre de 2009.

[22] DOMAIN NAMES REGULATIONS as of 18 December 2006. http://dns.pl/english/regulations.html. Último acceso Noviembre de 2009.

[23] Internet Corporation for Asigned Names and Numbres. ICANNWiki, thin registry. http://icannwiki.org/Thick_Registry. Último acceso Enero de 2010.

[24] Internet Corporation for Asigned Names and Numbres. ICANNWiki, thick registry. http://icannwiki.org/Thick_Registry. Último acceso Febrero de 2011.

[25] Gavin Brown. The Extensible Provisioning Protocol. http://www.uknof.org.uk/uknof05/Brown-EPP/, 2006. Último acceso: Julio de 2009.

[26] S. Hollenbeck. Extensible Provisioning Protocol (EPP). RFC 4930 (Draft Standard), mayo 2007. Obsoleted by RFC 5730.

[27] K. Zeilenga. The PLAIN Simple Authentication and Security Layer (SASL) Mechanism. RFC 4616 (Proposed Standard), ago. 2006.

[28] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Contact Mapping. RFC 4933 (Draft Standard), mayo 2007. Obsoleted by RFC 5733.

[29] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Host Mapping. RFC 4932 (Draft Standard), mayo 2007. Obsoleted by RFC 5732.

[30] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Domain Name Mapping. RFC 4931 (Draft Standard), mayo 2007. Obsoleted by RFC 5731.

[31] Oxford Swindon & Gloucester Co-op Domains Ltd, http://www.nic.coop. *EPP Extensions for the .coop TLD Registrant Verification.*

[32] P. Faltstrom, P. Hoffman, and A. Costello. Internationalizing Domain Names in Applications (IDNA). RFC 3490 (Proposed Standard), mar. 2003.

[33] P. Hoffman and M. Blanchet. Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN). RFC 3491 (Proposed Standard), mar. 2003.

[34] P. Hoffman and M. Blanchet. Preparation of Internationalized Strings ("stringprep"). RFC 3454 (Proposed Standard), dic. 2002.

[35] A. Costello. Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA). RFC 3492 (Proposed Standard), mar. 2003.

[36] Verisign, http://www.verisign.com/domain-name-services/domain-information-center/domain-name-resources/. *Extensible Provisioning Protocol Extension Mapping: IDN Language Tag.*

[37] Andrzej Bartosiewicz. *Registering Internationalized Domain Names under .PL.* NASK, .PL TLD, http://www.bartosiewicz.pl/, nov. 2003. Último acceso Febrero de 2010.

[38] S. Hollenbeck. Domain Name System (DNS) Security Extensions Mapping for the Extensible Provisioning Protocol (EPP). RFC 4310 (Proposed Standard), dic. 2005. Obsoleted by RFC 5910.

[39] Tomek Zygmutowicz et al. *EPP parameters for .pl ccTLD.* NASK .PL ccTLD, http://www.nask.pl, oct. 2004.

[40] NASK, http://www.nask.pl. *NASK EPP Extension*, feb. 2008.

[41] F. Neves. BR Organization Mapping for the Extensible Provisioning Protocol (EPP). ftp://ftp.registro.br/pub/libepp-nicbr/draft-neves-epp-brdomain-03.txt, Último acceso enero de 2010, ago. 2006.

[42] F. Neves. BR Domain Mapping for the Extensible Provisioning Protocol (EPP). ftp://ftp.registro.br/pub/libepp-nicbr/draft-neves-epp-brdomain-03.txt, Último acceso enero de 2010, ago. 2006.

[43] F. Neves. BR Portuguese Language Table. http://www.iana.org/assignments/idn/br-portuguese.html, ago. 2005.

[44] S. Hollenbeck. Domain Registry Grace Period Mapping for the Extensible Provisioning Protocol (EPP). RFC 3915 (Proposed Standard), sep. 2004.

[45] Mark V.B. Partridge and Scott T. Lonardo. ICANN Can or Can It?: Recent Developments in Internet Governance Involving Cybersquatting, Online Infringement, and Registration Practices. *Landslide*, 1(5), mayo 2009. © 2009 by the American Bar Association.

[46] S. Hollenbeck. Extensible Provisioning Protocol (EPP). RFC 3730 (Proposed Standard), mar. 2004. Obsoleted by RFC 4930.

[47] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Domain Name Mapping. RFC 3731 (Proposed Standard), mar. 2004. Obsoleted by RFC 4931.

[48] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Host Mapping. RFC 3732 (Proposed Standard), mar. 2004. Obsoleted by RFC 4932.

[49] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Contact Mapping. RFC 3733 (Proposed Standard), mar. 2004. Obsoleted by RFC 4933.

[50] S. Hollenbeck. Extensible Provisioning Protocol (EPP) Transport Over TCP. RFC 3734 (Proposed Standard), mar. 2004. Obsoleted by RFC 4934.

[51] L. Daigle. WHOIS Protocol Specification. RFC 3912 (Draft Standard), sep. 2004.

[52] M. Hunt. System for managing a shared domain registry. http://tools.ietf.org/html/draft-nzrs-srs-02, Julio 2010. Draft.

[53] S. Hollenbeck and M. Srivastava. NSI Registry Registrar Protocol (RRP) Version 1.1.0. RFC 2832 (Informational), mayo 2000. Updated by RFC 3632.

[54] S. Hollenbeck, S. Veeramachaneni, and S. Yalamanchilli. VeriSign Registry Registrar Protocol (RRP) Version 2.0.0. RFC 3632 (Informational), nov. 2003.

[55] S. Hollenbeck. E.164 Number Mapping for the Extensible Provisioning Protocol (EPP). RFC 4114 (Proposed Standard), jun. 2005.

[56] B. Hoeneisen. ENUM Validation Information Mapping for the Extensible Provisioning Protocol. RFC 5076 (Proposed Standard), dic. 2007.

[57] A. Newton and M. Sanz. IRIS: The Internet Registry Information Service (IRIS) Core Protocol. RFC 3981 (Proposed Standard), ene. 2005. Updated by RFC 4992.

[58] Jakob Nielsen. *Usability Engineering*. Academic Press, 1993.

FIN

Parte IV

Apéndices

Apéndice A

Ejecución del Modelo de Pruebas

A.1. Introducción

Los juegos de datos utilizados se encuentran en el directorio *pruebas* de la raíz del disco de la entrega. El *script* para la carga inicial de los datos está en el sub-directorio *00_carga_inicial*. Las tablas a continuación detallan los datos usados en las pruebas.

A.2. Mapeo de Casos de Pruebas

Los casos de prueba definidos en el Capítulo 9 se relacionan con la ejecución de pruebas de la siguiente manera:

9.3.1 Creación de dominio especial	
A.3	Creación de Dominio Común
A.4	Denegación de Creación de Dominio Especial
A.5	Aprobación de Creación de Dominio Especial
A.10	Aprobación de dominio *IDN* especial en conflicto, aprobando el conflicto primero
A.11	Aprobación de dominio *IDN* especial en conflicto aprobando especial primero
A.12	Denegación de creación de dominio *IDN* gubernamental en conflicto aprobando *IDN*
A.13	Denegación de dominio *IDN* gubernamental en conflicto aprobando creación de gubernamental
A.14	Denegación de dominio *IDN* gubernamental en conflicto denegando *IDN*
A.15	Denegación de Dominio *IDN* especial en conflicto aprobando especial

9.3.2 Creación de dominio *IDN*	
A.6	Creación de Dominio *IDN* inválido
A.7	Creación de Dominio *IDN*
A.8	Denegación de creación de dominio *IDN* en conflicto
A.9	Aprobación de la creación de un dominio *IDN* en conflicto
A.10	Aprobación de dominio *IDN* especial en conflicto, aprobando el conflicto primero
A.11	Aprobación de dominio *IDN* especial en conflicto aprobando especial primero

A.12	Denegación de creación de dominio *IDN* gubernamental en conflicto aprobando *IDN*
A.13	Denegación de dominio *IDN* gubernamental en conflicto aprobando creación de gubernamental
A.14	Denegación de dominio *IDN* gubernamental en conflicto denegando *IDN*
A.15	Denegación de Dominio *IDN* especial en conflicto aprobando especial

9.3.3 *DNSSEC*	
A.17	Creación de un dominio con información de *DNSSEC*
A.18	Agregar información de *DNSSEC* a un dominio
A.19	Modificar la información *DNSSEC* de un dominio
A.20	Agregar una segunda clave a un dominio con registro *DS* asociado
A.21	Borrar uno de los registros *DS* asociados a un dominio
A.22	Borrar el único registro *DS* asociado a un dominio

9.3.4 Reportes	
A.23	Obtener un reporte válido
A.24	Intentar un reporte inválido

9.3.5 Dominios Críticos	
A.16	Dominio Crítico

9.3.6 Logs	
A.3	Creación de Dominio Común
A.4	Denegación de Creación de Dominio Especial
A.5	Aprobación de Creación de Dominio Especial
A.6	Creación de Dominio *IDN* inválido
A.7	Creación de Dominio *IDN*
A.8	Denegación de creación de dominio *IDN* en conflicto
A.9	Aprobación de la creación de un dominio *IDN* en conflicto
A.10	Aprobación de dominio *IDN* especial en conflicto, aprobando el conflicto primero

A.11	Aprobación de dominio *IDN* especial en conflicto aprobando especial primero
A.12	Denegación de creación de dominio *IDN* gubernamental en conflicto aprobando *IDN*
A.13	Denegación de dominio *IDN* gubernamental en conflicto aprobando creación de gubernamental
A.14	Denegación de dominio *IDN* gubernamental en conflicto denegando *IDN*
A.15	Denegación de Dominio *IDN* especial en conflicto aprobando especial
A.16	Dominio Crítico
A.17	Creación de un dominio con información de *DNSSEC*
A.18	Agregar información de *DNSSEC* a un dominio
A.19	Modificar la información *DNSSEC* de un dominio
A.20	Agregar una segunda clave a un dominio con registro *DS* asociado
A.21	Borrar uno de los registros *DS* asociados a un dominio
A.22	Borrar el único registro *DS* asociado a un dominio

9.3.7 Pruebas de Performance	
A.25	Pruebas de performance

A.3. Creación de Dominio Común

Precondiciones	▪ no existe primero.com.uy ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio común	
Entrada	▪ *XML* para crear el dominio primero.com.uy con los datos anteriores
Salida	▪ *XML* de salida
Acciones	▪ utilizar *Preppi* para conectarse con el registrar dueño del registro y enviar el *XML*

Resultados esperados	▪ el *XML* de salida retorna un EPP_1000 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP ok* y los datos ingresados ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes ▪ no es posible ejecutar el caso con el mismo nombre de dominio (el xml de salida al create devuelve EPP 2302) ▪ chequear log de auditoría: create

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
		1.07_create_in.xml	crear otra vez
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
		1.07_create_in.xml	salida del segundo *create*

A.4. Denegación de Creación de Dominio Especial

Precondiciones	▪ no existe gobierno.gub.uy ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com ▪ existe el usuario *ARCO* 'estado'y existe el agente *SRS* de nombre 'estado'
1. Crear un dominio especial (que se necesita validar la creación)	
Entrada	▪ *XML* para crear dominio gobierno.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con el registrar y enviar el *XML*
Salida	▪ *XML* de salida

Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso; acción pendiente) ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el *ticket* ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes ▪ no es posible ejecutar el caso con el mismo nombre de dominio (el *XML* de salida al create devuelve EPP_2302) ▪ chequear log de auditoría: create (EPP_1001)
2. Denegar la creación del dominio	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, denegar
Resultados esperados	▪ el dominio se borra de *SRS* ▪ no se puede visualizar el dominio desde *ARCO* ▪ *check* indica que el dominio está disponible ▪ *info* indica *EPP_2303* (el objeto no existe)

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*

A.5. Aprobación de Creación de Dominio Especial

Precondiciones	▪ no existe el dominio gobierno.gub.uy ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com ▪ existe el usuario arco 'estado'y existe el agente *SRS* de nombre 'estado'
1. Crear un dominio especial	
Entrada	▪ *XML* para crear dominio gobierno.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con el registrar y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el ticket ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes ▪ no es posible ejecutar el caso con el mismo nombre de dominio (el *XML* de salida al create devuelve EPP 2302) ▪ chequear log de auditoría: create (1001)
2. Aceptar la creación del dominio	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, aceptar
Resultados esperados	▪ el dominio se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *EPP ok* ▪ *check* indica que el dominio no está disponible ▪ *info* entrega datos acordes con el *XML* de entrada

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.05_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*

A.6. Creación de Dominio *IDN* inválido

1. Creamos dominio *IDN* inválido	
Precondiciones	▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
Entrada	▪ *XML* para crear dominio parägua̲s.com.uy con los datos anteriores
Acciones	▪ utilizar *preppi* para conectarse con el registrar y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un error EPP_2306 ▪ chequear log de auditoría: create

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
	Salida	1.02_create_out.xml	salida del primer *create*

A.7. Creación de Dominio *IDN*

Precondiciones	▪ no existe el dominio pingüinos.org.uy ▪ no existe un domino que entre en conflicto con el anterior ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio *IDN*	
Entrada	▪ *XML* para crear dominio pingüinos.org.uy con los datos anteriores
Acciones	▪ utilizar *preppi* para conectarse con el registrar y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea activo en *SRS* ▪ el xml de salida retorna un EPP_1000 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP ok* y los datos ingresados y el registro *IDN* ▪ el volcado de zona (*write-zone*) contiene los NS con los nombres *ACE* ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* (con el nombre ACE) y los resultados son acordes ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el xml de salida al create devuelve EPP_2302) ▪ chequear log de auditoría

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
		1.07_write-zone.txt	volcado de *write-zone*

A.8. Denegación de creación de dominio *IDN* en conflicto

Precondiciones	▪ existe canierias.com.uy ▪ no existe un dominio *IDN* cañerías.com.uy ▪ existe un usuario administrador o superusuario en *ARCO* ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Intentar crear un dominio *IDN* en conflicto	
Entrada	▪ *XML* para crear dominio cañerías.com.uy con los datos anteriores
Acciones	▪ utilizar *preppi* para conectarse con el registrar y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea en *pendingCreate* en *SRS* ▪ el *XML* de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el registro *IDN* ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el xml de salida al create devuelve EPP 2302) ▪ chequear log de auditoría: create
2. Denegar la creación del dominio	
Acciones	▪ acceder a *ARCO* con un administrador o un superusuario ▪ ver las tareas *IDN* y seleccionar la asociada, denegar
Resultados esperados	▪ el dominio se borra de *SRS* ▪ no se puede visualizar el dominio desde *ARCO* ▪ *check* indica que el dominio está disponible ▪ info retorna un código EPP_2303

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*

A.9. Aprobación de la creación de un dominio *IDN* en conflicto

Precondiciones	▪ existe el dominio canierias.com.uy ▪ no existe un dominio *IDN* cañerías.com.uy ▪ existe un usuario administrador o superusuario ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Intentar crear un dominio *IDN* en conflicto	
Entrada	▪ *XML* para crear dominio cañerías.com.uy con los datos anteriores
Acciones	▪ utilizar *preppi* para conectarse con un *registrar* y enviar el *XML*
Salida	▪ xml de salida
Resultados esperados	▪ el dominio se crea en *pendingCreate* en *SRS* ▪ el *XML* de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el registro *IDN* ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el *XML* de salida al *create* devuelve EPP_2302) ▪ chequear log de auditoría: create

2. Aceptar la creación del dominio	
Acciones	▪ acceder a *arco* con un usuario administrador o superusuario ▪ ver las tareas *IDN* y seleccionar la asociada, aceptar
Resultados esperados	▪ el dominio se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* ▪ *check* indica que el dominio no está disponible ▪ *info* acorde

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
	Salida	2.02_info_out.xml	salida del *info*

A.10. Aprobación de dominio *IDN* especial en conflicto, aprobando el conflicto primero

Precondiciones	▪ existe republica.gub.uy ▪ no existe un dominio *IDN* república.gub.uy ▪ existe un usuario administrador o superusuario ▪ existe el agente 'estado' ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio *IDN* especial (.MIL o .GUB, que requieren autorización) en conflicto	
Entrada	▪ *XML* para crear dominio república.gub.uy con los datos anteriores

Acciones	▪ utilizar *preppi* para conectarse con un *registrar* y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea en *pendingCreate* en *SRS* ▪ se crea el conflicto *IDN* ▪ el *XML* de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el registro *IDN* ▪ es posible ver el conflicto desde *ARCO* (con un usuario autorizado) ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* (con el nombre *ACE*) y los resultados son acordes (se muestran el estado *pendingCreate*, el *ticket* y el registro *IDN*) ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el *XML* de salida al create devuelve EPP_2302) ▪ chequear log de auditoría: create
2. Aprobar el conflicto	
Acciones	▪ acceder a arco con un usuario administrador o superusuario ▪ ver las tareas *IDN* y seleccionar la asociada, aprobar
Resultados esperados	▪ se borra el conflicto ▪ el dominio no se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *pendingCreate* ▪ *check* indica que el dominio no está disponible ▪ *info* acorde (sigue en *pendingCreate* y se muestra el *ticket*)
3. Aprobar la creación del dominio especial	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, aceptar

Resultados esperados	▪ el dominio se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *EPP ok* ▪ *check* indica que el dominio no está disponible ▪ *info* muestra información acorde con el *XML* de entrada

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*
3	Entrada	3.01_info_in.xml	*info* del dominio
		3.03_check_in.xml	*check* del dominio
	Salida	3.02_info_out.xml	salida del *info*
		3.04_check_out.xml	salida del *check*

A.11. Aprobación de dominio *IDN* especial en conflicto aprobando especial primero

Precondiciones	▪ existe el dominio republica.gub.uy (sin tilde en la u) ▪ no existe un dominio *IDN* república.gub.uy ▪ existe un usuario administrador ▪ existe el agente 'estado' ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com

1. Crear un dominio *IDN* gubernamental en conflicto	
Entrada	▪ *XML* para crear el dominio república.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea en *pendingCreate* en *SRS* ▪ se crea el conflicto *IDN* con republica.gub.uy ▪ el xml de salida retorna un código EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el registro *IDN* ▪ es posible ver el conflicto desde *ARCO* (con un usuario autorizado) ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* (con el nombre *ACE*) y los resultados son acordes (estado *pendingCreate*, se muestra el *ticket* correspondiente y se muestran los datos del registro *IDN*) ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el *XML* de salida al *create* devuelve EPP_2302) ▪ el log de auditoría registra un *create* con código de salida EPP_1001 (comando exitoso, acción pendiente)
2. Aprobar la creación del dominio gubernamental	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, aceptar

Resultados esperados	▪ se borra el *ticket* ▪ sigue existiendo el conflicto ▪ el dominio no se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *pendingCreate* ▪ *check* indica que el dominio no está disponible ▪ *info* muestra estado *pendingCreate* y los datos del registro *IDN*. No se muestra más el ticket.
3. Aprobar el conflicto *IDN*	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *IDN* y seleccionar la asociada, aprobar
Resultados esperados	▪ el dominio se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *EPP ok* ▪ *check* indica que el dominio no está disponible ▪ *info* muestra estado *EPP ok* y los datos del registro *IDN*

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*
3	Entrada	3.01_info_in.xml	*info* del dominio
		3.03_check_in.xml	*check* del dominio
	Salida	3.02_info_out.xml	salida del *info*
		3.04_check_out.xml	salida del *check*

A.12. Denegación de creación de dominio *IDN* gubernamental en conflicto aprobando *IDN*

Precondiciones	▪ existe el dominio republica.gub.uy (sin tilde en la u) ▪ no existe un dominio *IDN* república.gub.uy ▪ existe un usuario administrador ▪ existe el agente 'estado' ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio *IDN* gubernamental en conflicto	
Entrada	▪ *XML* para crear dominio república.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* y enviar el *XML*
Salida	▪ *XML* de salida

Resultados esperados	▪ el dominio se crea en estado *pendingCreate* en *SRS* ▪ se crea el conflicto *IDN* con republica.gub.uy ▪ el xml de salida retorna código EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate*, los datos ingresados y el registro *IDN* ▪ es posible ver el conflicto desde *ARCO* (con un usuario autorizado) ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes (estado *pendingCreate*, se muestra el *ticket* correspondiente, se muestran los datos del registro *IDN*) ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el *XML* de salida al create devuelve EPP_2302) ▪ el log de auditoría muestra un *create* con resultado EPP_1001 (comando exitoso, acción pendiente)
2. Aprobar el conflicto *IDN*	
acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *IDN* y seleccionar la asociada, aprobar
Resultados esperados	▪ se borra el conflicto *IDN* ▪ el dominio no se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *EPP pendingCreate* ▪ *check* indica que el dominio no está disponible ▪ *info* muestra estado *pendingCreate* y *ticket*)
3. Negar la creación del dominio gubernamental	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, negar

Resultados esperados	▪ el dominio se borra de *SRS* ▪ no se puede visualizar el dominio desde *ARCO* ▪ se borra el *ticket* ▪ *check* indica que el dominio está disponible ▪ *info* indica que el objeto no existe (código EPP_2303)

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*
3	Entrada	3.01_info_in.xml	*info* del dominio
		3.03_check_in.xml	*check* del dominio
	Salida	3.02_info_out.xml	salida del *info*
		3.04_check_out.xml	salida del *check*

A.13. Denegación de dominio *IDN* gubernamental en conflicto aprobando creación de gubernamental

Precondiciones	▪ existe republica.gub.uy (sin tilde en la u) ▪ no existe un dominio *IDN* república.gub.uy ▪ existe un usuario administrador ▪ existe el agente 'estado' ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio *IDN* gubernamental en conflicto	

Entrada	▪ *xml* para crear dominio república.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea en estado *pendingCreate* en *SRS* ▪ se crea el conflicto *IDN* con republica.gub.uy ▪ el *XML* de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate*, los datos ingresados y el registro *IDN* ▪ es posible ver el conflicto desde *ARCO* (con un usuario autorizado) ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes (se muestra estado *pendingCreate*, se muestra el *ticket*, *IDN*) ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* ▪ el log de auditoría muestra un *create* con resultado EPP_1001 (comando exitoso, acción pendiente)
2. Aprobar la creación del dominio gubernamental	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, aceptar
Resultados esperados	▪ se borra el *ticket* ▪ sigue estando el conflicto *IDN* ▪ el dominio no se activa en *SRS* ▪ se puede visualizar el dominio desde *ARCO* con estado *EPP pendingCreate* ▪ *check* indica que el dominio no está disponible ▪ *info* muestra que el dominio sigue en *pendingCreate*
3. Denegar el conflicto *IDN*	

Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *IDN* y seleccionar la asociada, negar
Resultados esperados	▪ el dominio se borra de *SRS* ▪ no se puede visualizar el dominio desde *ARCO* ▪ *check* indica que el dominio está disponible ▪ *info* indica que el dominio no existe (EPP_2303)

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*
3	Entrada	3.01_info_in.xml	*info* del dominio
		3.03_check_in.xml	*check* del dominio
	Salida	3.02_info_out.xml	salida del *info*
		3.04_check_out.xml	salida del *check*

A.14. Denegación de dominio *IDN* gubernamental en conflicto denegando *IDN*

Precondiciones	▪ existe el dominio republica.gub.uy (sin tilde en la u) ▪ no existe un dominio *IDN* república.gub.uy ▪ existe un usuario administrador ▪ existe el agente 'estado' ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio *IDN* gubernamental en conflicto *IDN*	
Entrada	▪ *XML* para crear el dominio república.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea en estado *pendingCreate* en *SRS* ▪ se crea el conflicto *IDN* ▪ el xml de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate*, los datos ingresados y el registro *IDN* ▪ es posible ver el conflicto desde *ARCO* (con un usuario autorizado) ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes (estado *pendingCreate*, existe el *ticket*, registro *IDN*) ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* ▪ el log de auditoría muestra un *create* con resultado EPP_1001 (comando exitoso, acción pendiente)
2. Denegar el conflicto *IDN*	

Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *IDN* y seleccionar la asociada, denegar
Resultados esperados	▪ el dominio se borra de *SRS* ▪ se borra el conflicto *IDN* ▪ no se puede visualizar el dominio desde *ARCO* ▪ se borra el *ticket* ▪ *check* indica que el dominio está disponible ▪ *info* indica que el objeto no existe (EPP_2303)

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*

A.15. Denegación de Dominio *IDN* especial en conflicto aprobando especial

Precondiciones	▪ existe el dominio republica.gub.uy (sin tilde en la u) ▪ no existe un dominio *IDN* república.gub.uy ▪ existe un usuario administrador ▪ existe el agente 'estado' ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio *IDN* gubernamental en conflicto	

Entrada	▪ *XML* para crear el dominio república.gub.uy con los datos anteriores
Acciones	▪ utilizar *Preppi* para conectarse con un *registrar* y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el dominio se crea con estado *pendingCreate* en *SRS* ▪ se crea el conflicto *IDN* con republica.gub.uy ▪ el *XML* de salida retorna un EPP_1001 ▪ es posible ver el dominio desde *ARCO* (con un usuario autorizado) con estado *EPP pendingCreate* y los datos ingresados y el registro *IDN* ▪ es posible ver el conflicto desde *ARCO* (con un usuario autorizado) ▪ es posible ejecutar los comandos de consulta *EPP info* y *check* y los resultados son acordes (estado *EPP pendingCreate*, existe el *ticket*, aparece el registro *IDN*) ▪ no es posible ejecutar el caso con el mismo nombre de dominio *ACE* (el *XML* de salida al create devuelve EPP_2302) ▪ el log de auditoría muestra un *create* con resultado EPP_1001 (comando exitoso, acción pendiente)
2. Denegar la creación del dominio gubernamental	
Acciones	▪ acceder a *ARCO* con el agente 'estado' ▪ ver las tareas y seleccionar la asociada, denegar
Resultados esperados	▪ el dominio se borra de *SRS* ▪ se borra el *ticket* ▪ se borra el conflicto *IDN* ▪ no se puede visualizar el dominio desde *ARCO* ▪ *check* indica que el dominio está disponible ▪ *info* que el objeto no existe (EPP_2303)

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
		1.05_check_in.xml	*check* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.06_check_out.xml	salida del *check*
2	Entrada	2.01_info_in.xml	*info* del dominio
		2.03_check_in.xml	*check* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.04_check_out.xml	salida del *check*

A.16. Dominio Crítico

Precondiciones	▪ existe el dominio critico.com.uy marcado como crítico
1. Borrar el dominio crítico	
Entrada	▪ *XML* para borrar el dominio critico.com.uy
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_2309 (el dominio no se puede borrar) ▪ el dominio no se borró continuando en el estado original
2. Marcar el dominio como no crítico y borrarlo	
Entrada	▪ *XML* para borrar el dominio critico.com.uy
Acciones	▪ utilizar *ARCO* para marcar el dominio como crítico ▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1000 (comando exitoso) ▪ el dominio se borra ▪ *check* indica que el dominio está disponible ▪ *info* indica que el objeto no existe EPP_2303

Datos de prueba

1	Entrada	1.01_delete_in.xml	borrar el dominio
	Salida	1.02_delete_out.xml	salida del primer *delete*
1	Entrada	2.01_delete_in.xml	borrar el dominio
	Salida	2.02_delete_out.xml	salida del segundo *delete*

A.17. Creación de un dominio con información de *DNSSEC*

Precondiciones	▪ no existe el dominio seguro.com.uy ▪ existen los contactos con handle ep1-uynic y pc1-uynic ▪ existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Crear un dominio que tenga asociado un registro *DS*	
Entrada	▪ *XML* para crear el dominio seguro.com.uy con un registro *DS* asociado
Acciones	▪ utilizar *Preppi* para conectarse con un registrar y enviar el *XML*
Salida	▪ xml de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1000 ▪ el dominio se creó en estado *EPP ok* ▪ *info* muestra la información de delegación asociada en la creación

Datos de prueba

1	Entrada	1.01_create_in.xml	crear el dominio
		1.03_info_in.xml	*info* del dominio
	Salida	1.02_create_out.xml	salida del primer *create*
		1.04_info_out.xml	salida del *info*
		1.05_write-zone.txt	volcado del *write-zone*

A.18. Agregar información de *DNSSEC* a un dominio

Precondiciones	▪ existe el dominio semiseguro.com.uy sin información de *DNSSEC* asociada ▪ existe un usuario administrador

1. Asociar un registro *DS* a un dominio que no tenga asociada dicha información	
Entrada	▪ *XML* para actualizar el dominio semiseguro.com.uy asociando un registro *DS*
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño de seguro.com.uy y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción retrasada) ▪ *ARCO* muestra la tarea *DNSSEC* asociada a la acción ▪ se crea el *ticket* con "pendiente de *DS*" ▪ *info* no muestra información de delegación
2. Denegar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, denegar
Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* no muestra la información de delegación
3. Asociar un registro *DS* al mismo dominio que en el paso 1	
Entrada	▪ *XML* para actualizar el dominio semiseguro.com.uy asociando un registro *DS*
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ Se crea el *ticket* con "pendiente por *DS*" ▪ *info* no muestra información de delegación
4. Aprobar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, aprobar
Salida	▪ volcado de *write-zone*

Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación ▪ *write-zone* genera un archivo de zona que asocia el registro *DS* al dominio

Datos de prueba

1	Entrada	1.01_update_in.xml	actualizar el dominio
		1.03_info_in.xml	*info* del dominio
	Salida	1.02_update_out.xml	salida del primer *update*
		1.04_info_out.xml	salida del *info*
		1.05_write-zone.txt	volcado del *write-zone*
2	Entrada	2.01_info_in.xml	*info* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.03_write-zone.xml	volcado del *write-zone*
3	Entrada	3.01_update_in.xml	segundo *update* al dominio
		3.03_info_in.xml	*info* del dominio
	Salida	3.02_update_out.xml	salida del segundo *update*
		3.04_info_out.xml	salida del *info*
4	Entrada	4.01_info_in.xml	*info* del dominio
	Salida	4.02_info_out.xml	salida del *info*
		4.03_write-zone.xml	volcado del *write-zone*

A.19. Modificar la información *DNSSEC* de un dominio

Precondiciones	▪ existe el dominio semiseguro.com.uy con un registro *DS* asociado ▪ existe un usuario administrador
1. Modificar el registro *DS* asociado al dominio	
Entrada	▪ *XML* para actualizar dominio semiseguro.com.uy modificando el registro *DS* asociado
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida ▪ volcado de la salida de *write-zone*

Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción retrasada) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ se crea el *ticket* con "pendiente de *DS*" ▪ *info* muestra la información de delegación anterior ▪ *write-zone* muestra la información de delegación anterior
2. Denegar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, denegar
Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación anterior
3. Modificar el registro *DS* asociado al dominio	
Entrada	▪ *XML* para actualizar dominio semiseguro.com.uy modificando el registro *DS* asociado
Acciones	▪ utilizar *Preppi* para conectarse con el *Registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción pendiente) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ Se crea el *ticket* con "pendiente por *DS*" ▪ *info* muestra la información de delegación anterior
4. Aprobar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, aprobar
Salida	▪ volcado de la salida de *write-zone*

Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación nueva ▪ *write-zone* muestra la información de delegación nueva

Datos de prueba

1	Entrada	1.01_update_in.xml	actualizar el dominio
		1.03_info_in.xml	*info* del dominio
	Salida	1.02_update_out.xml	salida del primer *update*
		1.04_info_out.xml	salida del *info*
		1.05_write-zone.txt	volcado del *write-zone*
2	Entrada	2.01_info_in.xml	*info* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.03_write-zone.xml	volcado del *write-zone*
3	Entrada	3.01_update_in.xml	segundo *update* al dominio
		3.03_info_in.xml	*info* del dominio
	Salida	3.02_update_out.xml	salida del segundo *update*
		3.04_info_out.xml	salida del *info*
4	Entrada	4.01_info_in.xml	*info* del dominio
	Salida	4.02_info_out.xml	salida del *info*
		4.03_write-zone.xml	volcado del *write-zone*

A.20. Agregar una segunda clave a un dominio con registro *DS* asociado

Precondiciones	▪ existe el dominio semiseguro.com.uy con información de *DNSSEC* asociada ▪ existe un usuario administrador
1. Agregar un segundo registro *DS* a un dominio	
Entrada	▪ *XML* para actualizar el dominio semiseguro.com.uy agregando un registro *DS* más
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida

Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción pendiente) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ Se crea el *ticket* con "pendiente de *DS*" ▪ *info* muestra la información de delegación anterior
2. Denegar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, denegar
Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación anterior
3. Agregar la segunda clave de nuevo	
Entrada	▪ *XML* para actualizar dominio semiseguro.com.uy agregando un registro *DS* adicional
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción pendiente) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ se crea el *ticket* con "pendiente por *DS*" ▪ *info* muestra la información de delegación anterior
4. Aprobar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, aprobar
Salida	▪ volcado de salida del *write-zone*
Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación anterior y la agregada ▪ *write-zone* muestra la información de delegación anterior y la agregada

Datos de prueba

1	Entrada	1.01_update_in.xml	actualizar el dominio
		1.03_info_in.xml	*info* del dominio
	Salida	1.02_update_out.xml	salida del primer *update*
		1.04_info_out.xml	salida del *info*
		1.05_write-zone.txt	volcado del *write-zone*
2	Entrada	2.01_info_in.xml	*info* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.03_write-zone.xml	volcado del *write-zone*
3	Entrada	3.01_update_in.xml	segundo *update* al dominio
		3.03_info_in.xml	*info* del dominio
	Salida	3.02_update_out.xml	salida del segundo *update*
		3.04_info_out.xml	salida del *info*
4	Entrada	4.01_info_in.xml	*info* del dominio
	Salida	4.02_info_out.xml	salida del *info*
		4.03_write-zone.xml	volcado del *write-zone*

A.21. Borrar uno de los registros *DS* asociados a un dominio

Precondiciones	▪ existe el dominio semiseguro.com.uy con dos registros *DS* asociados ▪ existe un usuario administrador
1. Borrar la primer clave asociada al dominio	
Entrada	▪ *XML* para actualizar dominio semiseguro.com.uy borrando uno de los registros *DS* asociados
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el xml de salida retorna un EPP_1001 (comando exitoso, acción retrasada) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ Se crea el *ticket* con "pendiente de *DS*" ▪ *info* muestra los dos registros *DS* asociados
2. Denegar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, denegar

Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación anterior
3. Borrar la primera clave	
Entrada	▪ *XML* para actualizar dominio semiseguro.com.uy borrando un registro *DS*
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción retrasada) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ se crea el *ticket* con "pendiente por *DS*" ▪ *info* muestra la información de delegación anterior
4. Aprobar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador' ▪ ver las tareas *DNSSEC* y seleccionar la asociada, aprobar
Salida	▪ volcado de salida del *write-zone*
Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación que no se borró ▪ *write-zone* muestra la información de delegación que no se borró

Datos de prueba

1	Entrada	1.01_update_in.xml	actualizar el dominio
		1.03_info_in.xml	*info* del dominio
	Salida	1.02_update_out.xml	salida del primer *update*
		1.04_info_out.xml	salida del *info*
		1.05_write-zone.txt	volcado del *write-zone*
2	Entrada	2.01_info_in.xml	*info* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.03_write-zone.xml	volcado del *write-zone*
3	Entrada	3.01_update_in.xml	segundo *update* al dominio
		3.03_info_in.xml	*info* del dominio
	Salida	3.02_update_out.xml	salida del segundo *update*
		3.04_info_out.xml	salida del *info*
4	Entrada	4.01_info_in.xml	*info* del dominio
	Salida	4.02_info_out.xml	salida del *info*
		4.03_write-zone.xml	volcado del *write-zone*

A.22. Borrar el único registro *DS* asociado a un dominio

Precondiciones	▪ existe el dominio semiseguro.com.uy con un único registro *DS* asociado ▪ existe un usuario administrador
1. Borrar el registro *DS* asociado al dominio	
Entrada	▪ *XML* para actualizar el dominio semiseguro.com.uy borrando el registro *DS* asociado
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción retrasada) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ se crea el *ticket* con "pendiente de *DS*" ▪ *info* muestra la información de delegación
2. Denegar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, denegar

Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* muestra la información de delegación
3. Borrar el único registro *DS* asociado al dominio	
Entrada	▪ *XML* para actualizar el dominio semiseguro.com.uy borrando el único registro *DS* asociado
Acciones	▪ utilizar *Preppi* para conectarse con el *registrar* dueño del dominio y enviar el *XML*
Salida	▪ *XML* de salida
Resultados esperados	▪ el *XML* de salida retorna un EPP_1001 (comando exitoso, acción retrasada) ▪ *ARCO* muestra la tarea *DNSSEC* asociada ▪ se crea el ticket con "pendiente por *DS*" ▪ *info* muestra la información de delegación
4. Aprobar la tarea	
Acciones	▪ acceder a *ARCO* con un usuario administrador ▪ ver las tareas *DNSSEC* y seleccionar la asociada, aprobar
Salida	▪ volcado de salida del *write-zone*
Resultados esperados	▪ se borra la tarea ▪ se borra el *ticket* ▪ *info* no muestra ninguna información de delegación ▪ el volcado de la salida del *write-zone* no muestra ninguna información de delegación

Datos de prueba

1	Entrada	1.01_update_in.xml	actualizar el dominio
		1.03_info_in.xml	*info* del dominio
	Salida	1.02_update_out.xml	salida del primer *update*
		1.04_info_out.xml	salida del *info*
		1.05_write-zone.txt	volcado del *write-zone*
2	Entrada	2.01_info_in.xml	*info* del dominio
	Salida	2.02_info_out.xml	salida del *info*
		2.03_write-zone.xml	volcado del *write-zone*
3	Entrada	3.01_update_in.xml	segundo *update* al dominio
		3.03_info_in.xml	*info* del dominio
	Salida	3.02_update_out.xml	salida del segundo *update*
		3.04_info_out.xml	salida del *info*
4	Entrada	4.01_info_in.xml	*info* del dominio
	Salida	4.02_info_out.xml	salida del *info*
		4.03_write-zone.xml	volcado del *write-zone*

A.23. Obtener un reporte válido

Precondiciones	▪ existen dominios con nombre que incluya ‘gratis’ ▪ no existe ningún dominio con nombre que incluya ‘porno’ ▪ existen dominios con titular ‘sr1-uynic’ ▪ no existe ningún dominio con titular ‘pr1-uynic’ ▪ existen dominios con contacto de tipo ‘admin’‘pc1-uynic’ ▪ no existe ningún dominio con contacto de tipo ‘tech’‘pc1-uynic’ ▪ no existe ningún dominio con contacto de tipo ‘admin’‘ep1-uynic’ ▪ existen algunos dominios con algún contacto ‘pr1-uynic’ ▪ no existe ningún dominio con ningún contacto ‘sr1-uynic’ ▪ existe algún dominio en estado *pendingCreate* ▪ no existe ningún dominio en estado *serverHold* ▪ existen dominios creados luego del 4/10/2010 ▪ existen dominios creados antes de 4/10/2010 ▪ existen dominios que expiren en 5/10/2012 ▪ no existe ningún dominio que expire en 5/10/1970 ▪ existen dominios asociados a hosts con ‘google’ ▪ no existe ningún dominio asociado a hosts con ‘fing’ ▪ existen dominios en *pendingDelete* del contacto ‘jm1-uynic’que venzan el 6/10/2012 ▪ no hay ningún dominio en *clientHold* del contacto ‘pr1-uynic’
1. Comprobar que se cumplen las precondiciones	
Entrada	▪ *XML* para hacer los reportes

Acciones	▪ usar *Preppi* para conectarse con un *registrar* que posea dominios que cumplen todas las pre-condiciones.
Salida	▪ *XML* de salida
Resultados esperados	▪ en cada consulta realizada se obtienen dominios pertenecientes al *registrar* que cumplen los criterios de las consultas, comprobando las pre-condiciones.
2. Probar la seguridad del comando *report*	
Entrada	▪ *XML* para hacer los reportes
Acciones	▪ usar *Preppi* para conectarse con un *registrar* que no posea dominios que cumplan alguna de las pre-condiciones.
Salida	▪ *XML* de salida
Resultados esperados	▪ el resultado es vacío
3. Comprobar las pre-condiciones usando filtros para las respuestas	
Entrada	▪ *XML* para hacer los reportes, eligiendo las columnas a mostrar en las respuestas
Acciones	▪ usar *Preppi* para conectarse con un *registrar* que posea dominios que cumplan alguna de las precondiciones
Salida	▪ *XML* de salida
Resultados esperados	▪ se muestran sólo los atributos pedidos de los dominios que cumplan los criterios de búsqueda

Datos de prueba

1	Entrada	1.01_find_gratis_in.xml – buscar dominios con nombre nombre que contenga *gratis*
		1.03_find_porno_in.xml – buscar dominios con nombre que contenga *porno*
		1.05_find_owner_sr1_in.xml – buscar dominios con titular *sr1-uynic*
		1.07_find_owner_pr1_in.xml – buscar dominios con titular *pr1-uynic*
		1.09_find_admin_pc1_in.xml – buscar dominios con contacto administrativo *pc1-uynic*
		1.11_find_tech_pc1_in.xml – buscar dominios con contacto técnico *pc1-uynic*

1	Entrada	1.13_find_admin_ep1_in.xml – buscar dominios con contacto administrativo *ep1-uynic*
		1.15_find_pr1_in.xml – buscar dominios con algún contacto *pr1-uynic*
		1.17_find_sr1_in.xml – buscar dominios con algún contacto *sr1-uynic*
		1.19_find_pendingCreate_in.xml – buscar dominios en estado *pendingCreate*
		1.21_find_serverHold_in.xml – buscar dominios en estado *serverHold*
		1.23_find_ge_crDate_in.xml – buscar dominios con fecha de creación mayor que una dada
		1.25_find_lt_crDate_in.xml – buscar dominios con fecha de creación menor que una dada
		1.27_find_eq_exDate_in.xml – buscar dominios con fecha de expiración igual a una dada
		1.29_find_eq_exDate_in.xml – buscar dominios con fecha de expiración igual a otra dada
		1.31_find_ns_google_in.xml – buscar dominios con algún *nameserver* cuyo *FQDN* contenga *google*
		1.33_find_ns_fing_in.xml – buscar dominios con algún *nameserver* cuyo *FQDN* contenga *fing*
		1.35_find_ct_status_exDate_in.xml – buscar dominios con determinado titular, estado y fecha de expiración
		1.37_find_ct_status_in.xml – buscar dominios con determinado titular y estado
	Salida	1.02_find_gratis_out.xml – dominios con nombre que contiene *gratis*
		1.04_find_porno_out.xml – dominios con nombre que contiene porno
		1.06_find_owner_sr1_out.xml – dominios con titular *sr1-uynic*
		1.08_find_owner_pr1_out.xml – dominios con titular *pr1-uynic*

1	Salida	1.10_find_admin_pc1_out.xml – dominios con contacto administrativo *pc1-uynic*
		1.12_find_tech_pc1_out.xml – dominios con contacto técnico *pc1-uynic*
		1.14_find_admin_ep1_out.xml – dominios con contacto administrativo *ep1-uynic*
		1.16_find_pr1_out.xml – dominios con algún contacto *pr1-uynic*
		1.18_find_sr1_out.xml – dominios con algún contacto *sr1-uynic*
		1.20_find_pendingCreate_out.xml – dominios en estado *pendingCreate*
		1.22_find_serverHold_out.xml – dominios en estado *serverHold*
		1.24_find_ge_crDate_out.xml – dominios con determinada fecha de creación
		1.26_find_lt_crDate_out.xml – dominios con fecha de creación menor a una dada
		1.28_find_eq_exDate_out.xml – dominios con fecha de expiración igual a una dada
		1.30_find_eq_exDate_out.xml – dominios con fecha de expiraciónigual a otra dada
		1.32_find_ns_google_out.xml – dominios con algún *nameserver* en *google*
		1.34_find_ns_fing_out.xml – dominios con algún servidor en *fing*
		1.36_find_ct_status_exDate_out.xml – dominios de un titular, en determinado estado y con determinada fecha de expiración
		1.38_find_ct_status_out.xml – dominios con determinado titular y en determinado estado

2	Entrada	2.01_find_pc1_filtrado_in.xml – buscar dominios con algún contacto *pc1-uynic*, aplicando filtros
		2.03_find_owner-sr1_filtrado_in.xml – buscar dominios con titular *sr1-uynic*, aplicando filtros
		2.05_find_status-pendingDelete_filtrado_in.xml – buscar dominios en estado *pendingDelete*, aplicando filtros
	Salida	2.02_find_pc1_filtrado_out.xml – algunos atributos de los dominios con algún contacto *pc1-uynic*
		2.04_find_owner-sr1_filtrado_out.xml – algunos atributos de los dominios con titular *sr1-uynic*
		2.06_find_status-pendingDelete_filtrado_out.xml – algunos atributos de los dominios en estado *pendingDelete*

A.24. Intentar realizar un reporte inválido

1. probar fechas inválidas en la consulta
2. probar fechas válidas con operador inválido
3. probar columnas inexistentes en el filtro

Datos de prueba

1	Entrada	1.01_find_fecha_invalida_in.xml	fecha inválida
	Salida	1.02_find_fecha_invalida_out.xml	fecha inválida
2	Entrada	2.01_find_oper_invalido_in.xml	operador inválido
	Salida	2.02_find_oper_invalido_out.xml	operador inválido
3	Entrada	3.01_find_filtro_invalido_in.xml	filtro inválido
	Salida	3.02_find_filtro_invalido_out.xml	filtro inválido

A.25. Prueba de performance

Para la ejecución de las pruebas de performance, se utiliza la herramienta *EPP Testing Tool*. Se crean lotes de comandos de creación de dominios que se encuentran en el sub-directorio *23_performance* del directorio *pruebas*. Son dos lotes de cien dominios cada uno, el primero con dominios comerciales y el segundo con dominios militares. Adicionalmente se crean dos lotes auxiliares

para el borrado masivo de los dominos creados. Se ejecutarán tanto contra un servidor *OpenReg* estándar, como contra *D-REX*.

Precondiciones	▪ no existe ninguno de los dominios a crear en ninguno de los dos servidores *EPP* ▪ en ambos servidores existen los contactos con handle ep1-uynic y pc1-uynic ▪ en ambos servidores existen los hosts google-public-dns-a.google.com y google-public-dns-b.google.com
1. Dominios comerciales (*OpenReg*)	
Entrada	▪ script de carga masiva de dominios comerciales
Acciones	▪ utilizar EPP Testing Tool para ejecutar el lote de comandos sobre el servidor *OpenReg* estándar
Salida	▪ log de ejecución
Resultados esperados	▪ el log de salida indica que los comandos fueron ejecutados exitosamente, indicando la duración total e individual
2. Dominios militares (*OpenReg*)	
Entrada	▪ script de carga masiva de dominios militares
Acciones	▪ utilizar EPP Testing Tool para ejecutar el lote de comandos sobre el servidor *OpenReg* estándar
Salida	▪ log de ejecución
Resultados esperados	▪ el log de salida indica que los comandos fueron ejecutados exitosamente, indicando la duración total e individual
3. Dominios comerciales (*D-REX*)	
Entrada	▪ script de carga masiva de dominios comerciales
Acciones	▪ utilizar EPP Testing Tool para ejecutar el lote de comandos sobre el servidor *D-REX*
Salida	▪ log de ejecución
Resultados esperados	▪ el log de salida indica que los comandos fueron ejecutados exitosamente, indicando la duración total e individual
4. Dominios militares (*D-REX*)	
Entrada	▪ script de carga masiva de dominios militares
Acciones	▪ utilizar EPP Testing Tool para ejecutar el lote de comandos sobre el servidor *D-REX*
Salida	▪ log de ejecución

Resultados esperados	▪ el log de salida indica que los comandos fueron ejecutados exitosamente, indicando la duración total e individual

Se registran, para cada una de las pruebas, 3 medidas de tiempo. Primero, interesa conocer el tiempo de ejecución del primer dominio creado, para medir el posible impacto del establecimiento de la conexión *XML-RPC* con *ARCO*. Segundo, interesa conocer el tiempo promedio de las siguientes ejecuciones, sin contar la primera, para obtener una estimación del tiempo en régimen. Finalmente, se registra el tiempo total de registro de los 100 dominios, para brindar una visión de tiempo global.

Los tiempos obtenidos fueron los siguientes:

1. Dominios comerciales (*OpenReg* estándar)	
Tiempo primer registro	0.099385s
Tiempo promedio siguientes	0.037450s
Tiempo total	3.806958s
2. Dominios militares (*OpenReg* estándar)	
Tiempo primer registro	0.0531s
Tiempo promedio siguientes	0.038269s
Tiempo total	3.841747s
3. Dominios comerciales (*D-REX*)	
Tiempo primer registro	1.732973s
Tiempo promedio siguientes	0.097086s
Tiempo total	11.344465s
2. Dominios militares (*D-REX*)	
Tiempo primer registro	0.163257s
Tiempo promedio siguientes	0.097280s
Tiempo total	9.79402s

Apéndice B

Instalación y configuración *D-REX*

B.1. Introducción

D-REX está probado sobre Debian GNU/Linux 4.0 (Debian 4.0 Release 8 "Etch And A Half", kernel 2.6.24). Se conocen incompatibilidades del módulo *OpenReg* que dificultan su ejecución en distribuciones más recientes. La causa principal de estas incompatibilidades es la versión de *Perl* incluida en las mismas.

Está compuesto por 4 aplicaciones que deben ser instaladas para su funcionamiento. Estas son:

- *OpenReg* en la versión modificada para este proyecto.
- *ARCO* para realizar la administración del repositorio y gestionar los procesos asociados a las acciones de los *registrars*.
- El prototipo de Agente legislador *ARCO-Legislador*.
- *ARCO-XMLRPCServer* para posibilitar la comunicación de *OpenReg* con el Agente Legislador.

B.1.1. Dependencias

Se requieren los siguientes paquetes instalados en el sistema operativo:

- build-essential
- stunnel4
- libexpat1-dev
- bind9
- libbind-dev
- expat
- postgresql-8.1
- postgresql-dev

Se deben instalar con el gestor de paquetes preferido. Para detalles sobre la instalación de los paquetes, consultar la página de manual del respectivo gestor.

Además, se deben instalar los siguientes módulos de *Perl*:

- Test::Harness
- Test::Simple
- Test::More
- ExtUtils::MakeMaker
- ExtUtils::ParseXS
- Probe::Perl
- IPC::Run3
- Test::Script
- File::Which
- Package::Constants
- Bundle::CPAN
- Event
- Time::HiRes
- MIME::Base64
- Text::Reform
- IO::Tee
- Bundle::DBI
- XML::NamespaceSupport
- HTML::Tagset
- HTML::Parser
- URI::Escape
- XML::Parser
- XML::SAX
- XML::SAX::Expat
- XML::Writer

- XML::Writer::String
- version
- DBD::Pg
- IDNA::Punycode
- Frontier::Client
- Clone

Estos pueden ser instalados de alguna de estas dos maneras:

```
$ perl -MCPAN -e 'install <nombre_de_paquete>'
$ cpan -i <nombre_de_paquete>
```

El ambiente necesario para instalar y ejecutar *ARCO* es el siguiente:

- *JBoss Application Server*. La aplicación fue probada, y está garantizado su funcionamiento, sobre un *JBoss* versión 4.2.3 GA *"out of the box"*. Se debe agregar el *driver* para *PostgreSQL* en el directorio *$JBOSS_HOME/$CONF/lib*.
- *PostgreSQL Server*. Se utilizará el mismo servidor que *OpenReg*. En la sección B.1.5 se detalla la creación de las dos bases de datos necesarias: *arco* y *srs*.
- *Eclipse IDE*. En caso de querer trabajar en el desarrollo, es necesario instalar los proyectos que componen *ARCO*. Los mismos fueron realizados utilizando *Eclipse Galileo* (3.5).

También depende de la política de seguridad *arcoWebAuthenticationPolicy*, la cual debe ser configurada en el servidor de aplicaciones (ver Sección B.1.2).

Es posible instalar el sistema en dos modos: con los proyectos *Eclipse* para desarrollo (ver Sección B.1.3, o con los binarios de la aplicación (ver Sección B.1.4.

B.1.2. Configuración de la política de seguridad

La autenticación y autorización de usuarios de *ARCO* está implementada utilizando el *framework JAAS*. Para que ésta funcione, es necesario agregar la política *arcoWebAuthenticationPolicy* en el archivo *login-config.xml* del directorio *$JBOSS_HOME/$CONF/conf*, donde *$JBOSS_HOME* es el directorio de instalación del servidor de aplicaciones, y *$CONF* es la configuración que se utiliza del servidor. El módulo de autenticación, debe configurarse con los siguientes atributos posee los siguientes atributos:

- *dsJndiName* con el nombre *JNDI* de la fuente de datos correspondiente a la política, la cual debe permitir el acceso a la base *arco* (ver Sección B.1.5).
- *principalsQuery* con la consulta que permite obtener las contraseñas de los usuarios.
- *rolesQuery* con la consulta que permite obtener los roles del usuario.

El archivo para configurar la fuente de datos, debe colocarse en el directorio *$JBOSS_HOME/$CONF/deploy*.

Los archivos adjuntos *arco-ds.xml* y *login-config.xml*, muestran la configuración utilizada durante las pruebas del sistema.

B.1.3. Instalación de desarrollo

Para realizar la instalación de los proyectos, deben existir una instalación funcional de *Eclipse IDE* y otra de *JBoss Application server*. Las versiones utilizadas durante el desarrollo y las pruebas fueron *Eclipse Galileo for JEE Developers SR2 (3.5)* y *JBoss 4.2.3-GA*. Sobre el *Eclipse* básico, se deben instalar los *plugins* para desarrollo en *C/C++* y *Perl*, llamados *Eclipse CDT* y *EPIC* respectivamente. Opcionalmente, es recomendable que se instalen, además, los *plugins* de *ICEFaces*, para facilitar el trabajo con la interfaz web.

Configuración de *Eclipse*

Los proyectos, dependen de las siguientes librerías:

ICEFaces-1.8.2 Los archivos `.jar` necesarios se encuentran en */bin/libraries/ICEFaces-1.8.2*.

JSF-1.1-RI Los archivos `.jar` necesarios se encuentran en */bin/libraries-/JSF-1.1-RI*.

epp-rtk-java-0.9.6 Los archivos `.jar` necesarios se encuentran en */bin/libraries/epp-rtk-java-0.9.6*.

libidn-1.9 Los archivos `.jar` necesarios se encuentran en *libidn-1.9*.

apache-xmlrpc-3.1.3 Los archivos `.jar` necesarios se encuentran en *apache-xmlrpc-3.1.3*.

Axis2-WebServices Los archivos `.jar` necesarios se encuentran en *Axis2-WebServices*

A modo de ejemplo, se muestra la creación de la librería *epp-rtk-java-0.9.6*.

1. En el menú *Preferencias->Java->Build Path->User Libraries* presionar *new* e indicar *epp-rtk-java-0.9.6* como nombre de librería.
2. Seleccionar la librería recién creada y presionar *Add JARs*.
3. Navegar hasta el directorio que contiene los archivos y seleccionarlos.
4. Presionar *Aceptar*. La figura B.1 muestra cómo debería haber quedado la librería.

Se puede realizar la importación de las librerías con el archivo *libraries* incluido en el directorio */bin/libraries* de la entrega. El mismo debe ser modificado para que las rutas apunten a los archivos correctos (hay que sustituir la variable *$LIB_DIR* por la ruta al directorio donde se encuentran los directorios con los `.jar`).

Una vez instaladas las librerías y creado el servidor de aplicaciones, se puede proceder a instalar y configurar los proyectos.

Instalación y configuración de los proyectos

Los proyectos se encuentran en el directorio */src/D-REX* de la entrega. Para instalarlos hay que recurrir a la utilidad de importación de *Eclipse*. Se selecciona *Proyectos Existentes* y se indica como carpeta raíz a la que contiene a las de los proyectos. Se eligen todos los proyectos y se presiona *Finish*.

Luego de instalados los proyectos, hay que configurar el *Build Path* de cada uno. Empezando desde las capas inferiores, a continuación se indica cómo hacerlo para cada proyecto.

- **ARCO-AdminData y ARCO-SRSData**

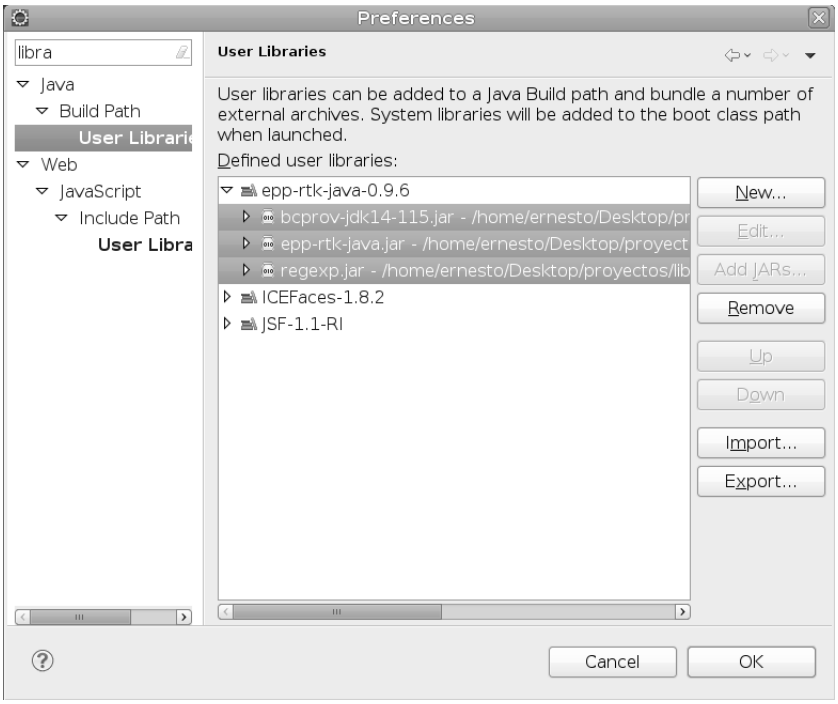


Figura B.1: Configuración de librerías en *Eclipse*

- Java JRE
- JBoss Server Runtime

- **ARCO-Data**
  - Java JRE
  - JBoss Server Runtime
  - Axis2-WebServices
  - Proyectos ARCO-AdminData y ARCO-SRSData
- **ARCO-Biz**
  - epp-rtk-java-0.9.6 marcada para exportar y como dependencia del módulo.
  - Axis2-WebServices marcada para exportar y como dependencia del módulo
  - JBoss Server Runtime
  - Proyectos ARCO-AdminData, ARCO-SRSData y ARCO-Data
- **ARCO-Web**

  - Todos los proyectos anteriores
  - JBoss Server Runtime
  - ICEFaces-1.8.2 marcada para exportar y como dependencia
  - libidn-1.19 marcada para exportar y como dependencia
  - JSF-1.1-RI marcada para exportar
- **ARCO-XMLRPCServer**
  - apache-xmlrpc-3.1.3 marcada para exportar y como dependencia
  - Axis2-WebServices marcada para exportar y como dependencia

En cuanto al prototipo del Agente Legislador (compuesto por ARCO-LegisladorData, ARCO-Legislador y LegisladorEAR), es necesario incluir la librería *JBoss Server Runtime*. En el caso de ARCO-Legislador, se debe referenciar a ARCO-LegisladorData. Para los proyectos *epp-frontend*, *msgq*, *write-zone* y *xaction*, no es necesario realizar ningún paso adicional.

Si se quisiera utilizar el cliente de prueba para el *WebService* de los agentes ejecutores, hay que instalar los proyectos *WS-Test* y *WS-TestEAR*. El primero depende de las librerías para aplicaciones Web y del *Runtime* de *JBoss*.

Por otro lado, si se quisiera utilizar el cliente de prueba por lotes de comandos *EPP*, se debe instalar el proyecto *epptt*.

B.1.4. Instalación binaria

La instalación binaria se compone de cinco archivos, los cuales se encuentran en el directorio */bin/D-REX* de la entrega:

- *ARCO-EAR.ear* contiene toda la aplicación de administración de *D-REX*.
- *ARCO-XML-RPC-Server.jar* es el servidor *XML-RPC* que permite la comunicación de *OpenReg* con el Agente Legislador.
- *Legislador-EAR.ear* es un prototipo de Agente Legislador, necesario para las pruebas realizadas.
- *WS-TestEAR.ear* es un cliente (generado por *Eclipse*) para el Agente Legislador.
- *Openreg.tar.gz* contiene la versión de *OpenReg* de *D-REX*.

Para instalar el sistema y poder realizar las pruebas, es necesario colocar todos los archivos **.ear* en el directorio *$JBOSS_HOME/$CONF/deploy*. Por otro lado, se debe descomprimir el archivo *OpenReg.tar.gz* incluído en el directorio */bin/D-REX/OpenReg*, a un directorio temporal (*$TEMP_DIR*), y ejecutar desde una consola los siguientes comandos:

- cd $TEMP_DIR
- ./configure
- make
- make install

Se deben agregar dos variables de entorno: una llamada *PERL5LIB*, que apunte al directorio *share/perl/$VERSION_PERL* dento del directorio de instalación (por defecto */usr/local/openreg/share/perl/5.8.8*), y otra llamada *ISC_SRS_DBPASS*, conteniendo la clave de la base de datos *srs* (por defecto *srs*).

B.1.5. Instalación de Bases de Datos

Para la instalación, utilizando el usuario del sistema para la base de datos, ir al sub-directorio *openreg-pgepp/db* del temporal creado antes, y ejecutar `make createdb`. Esto crea las bases de datos *arco* y *srs* y los roles *arco* y *srs*. Se debe permitir el acceso a ambas bases (*INSERT*, *SELECT*, *UPDATE* y *DELETE*) para ambos usuarios. Respecto a autenticación de usuarios, se debe configurar que el acceso a las bases de datos locales se realice utilizando contraseñas (por defecto son "arco" y "srs" respectivamente).

B.1.6. Configuración de *ARCO*

Para que *ARCO* funcione, deben existir las siguientes fuentes de datos:

- *ARCO-Data* apuntando a la base de datos *arco*, accediendo con el usuario creado durante la instalación, o con otro creado a tales efectos.
- *SRS-Data* apuntando a la base de datos *srs*, accediendo con algún usuario autorizado.

Ambas fuentes de datos deben permitir lectura y escritura de datos.

También es necesario configurar una cola de mensajes *JMS* llamada *tasksQueue*.

Estas configuraciones se realizan mediante los archivos *.xml* incluidos en los directorios *datasources* y *messaging* de la aplicación. En la instalación por proyectos, se encuentran en el proyecto *ARCO-EAR* y en la binaria en la raíz del archivo *ARCO-EAR.ear*.

B.1.7. Corriendo el sistema

ARCO entra en funcionamiento cuando se inician los servidores. En el caso de la instalación de proyectos, es necesario crear (en el *IDE*) un servidor de tipo *JBoss* que apunte a la instalación empleada, agregar los proyectos *EAR* al mismo, e iniciar el servidor. Para que funcione, se deben realizar las configuraciones que se describen en la Sección B.1.6.

Para ejecutar el servidor *XML-RPC* se puede iniciar el proyecto haciendo *click* derecho sobre el mismo en el *Project Explorer* de *Eclipse*, y seleccionando *Run as->Java Application*. También se puede iniciar la versión binaria, ejecutando el siguiente comando:

```
$ java -jar ARCO-XML-RPC-Server.jar
```

Para ejecutar los diferentes procesos que componen *OpenReg*, se deben ejecutar los siguientes comandos, en el orden indicado, y preferiblemente cada uno en su propia terminal de consola. Estos comandos se encuentran en el directorio de instalación, el cual por defecto es */usr/local/openreg*

```
$ sbin/msgq
$ sbin/log-monitor
$ sbin/stats-monitor
$ sbin/whois
$ sbin/xaction
$ sbin/epp-frontend
```

Si se desea correr los proyectos del ambiente de desarrollo, los proyectos se deben iniciar en el siguiente orden:

1. *msgq*
2. *xaction*
3. *epp-frontend*

El servidor *XML-RPC* se finaliza enviando las señales *SIG_INT* (por ejemplo, presionando *Ctrl.+C*) o *SIG_TERM* (con el comando *kill* de *GNU/*

Linux) al proceso. En caso de *"colgarse"* la terminal donde se esté ejecutando, el servidor se finaliza automáticamente.

La aplicación web finaliza una vez cerrado el servidor *JBoss*. Para finalizar *OpenReg*, en ambos tipos de instalación, hay que interrumpir (enviando la señal *SIG_INT*) el proceso del módulo *msgq*, lo cual causa la caída inmediata del resto.

Apéndice C

Manual de uso de las aplicaciones

C.1. ARCO

En la página *$RUTA/ARCO-Web*, se encuentra disponible la aplicación de administración de *D-REX*. Para ingresar por primera vez, se dispone del usuario 'root'con password '12345678'. Luego del primer ingreso, es recomendable dar de alta otro superusuario y borrar el primero de la base de datos.

C.1.1. Alta de nuevos usuarios

Accediendo con un superusuario, es posible dar de alta los usuarios del sistema. Para esto se debe acceder mediante el menú al área de usuarios y seleccionar "Crear Nuevo". Se ingresan los datos pedidos, se seleccionan los roles y se presiona "Enviar". En la figura C.1 se puede ver la pantalla para registro de usuarios.

Figura C.1: Formulario para crear usuarios *ARCO*

Creación de un Agente

Cuando se cree un usuario de tipo agente en *ARCO*, debido a limitaciones en el sistema, se lo deberá registrar, manualmente, en la Tabla *agent* de la base *SRS*. Para esto, se puede ejecutar la siguiente sentencia SQL:

```
INSERT INTO agent(agent_id, name, description)
VALUES(
    nextval('agentid_seq'),
    $AGENT_NAME$,
    $AGENT_DESC$
)
```

El valor para *AGENT_NAME*, debe coincidir con el nombre de inicio de sesión del Agente creado.

C.1.2. Área de Dominios

En el área de dominios es posible visualizar los dominios registrados en el sistema. Según el rol del usuario *loggeado*, se podrán realizar las siguientes tareas:

Administrador y superusuario Puede ver todos los dominios y revisar el detalle de cada uno. Además, puede cambiar la calidad de crítico para un conjunto de dominios.

Agent Puede ver los dominios que tiene registrados y revisar el detalle de cada uno.

Cambio de calidad de crítico

Para cambiar la calidad de crítico de uno o más dominios, se deben seleccionar los dominios a cambiar en el área de usuarios y presionar "Cambiar a Crítico". Luego de esto, se pasa a una pantalla como la que se muestra en la figura C.2, donde se pide que se confirmen los cambios.

Luego de aceptados los cambios, la calidad de crítico se negará en la base de datos, es decir, que si un dominio no era crítico pasará a serlo, y viceversa.

C.1.3. Área de Tareas

El Área de Tareas es la destinada a las verificaciones que se realizan durante el proceso de verificación de registro de dominios "especiales". Según el rol adquirido en el ingreso al sistema, se pueden realizar las siguientes acciones:

Superusuario puede ver la lista completa de tareas e inspeccionar cada una.

Agente puede ver la lista de tareas asignadas, inspeccionar cada una y resolverlas.

Figura C.2: Confirmar cambios de criticidad

Resolución de tareas

Para resolver una tarea, un usuario con rol Agente, debe acceder al Área de Tareas y seleccionar la tarea a resolver, presionando sobre la lupa que aparece sobre la derecha. En la siguiente pantalla, como se muestra en la figura C.3, podrá marcar el paso como aceptado o no.

La contestación de la tarea, se hace de manera asíncrona, por lo cual no se borrará automáticamente de la lista, y se deberá comprobar su resolución recargando la página. Si la tarea permanece un tiempo excesivo luego de su contestación, es un indicio de que surgieron problemas en el procesamiento.

Los Superusuarios, podrán ver las tareas e inspeccionarlas, pero no podrán decidir sobre las mismas.

C.1.4. Área *IDN*

En este área, los Administradores y Superusuarios, se pueden ver los conflictos que surgen por similaridad de nombres *IDN* y resolverlos. Del listado se selecciona un ítem, y se accede a una pantalla como la mostrada en la figura C.4

Presionando el botón “Aceptar”, el conflicto se resuelve favorablemente, es decir, que se acepta el registro del nombre que creó el conflicto. Para ayudar a la decisión, se puede ver la información de contacto para los dos dominios involucrados.

Figura C.3: Contestar una tarea

C.1.5. Área *DNSSEC*

Aquí se realiza la autorización de los cambios en la información de *DNSSEC*. Administradores y Superusuarios podrán ver los datos relativos al cambio en una pantalla como la que se muestra en la figura C.5, y aceptar o denegar el cambio.

C.1.6. Área Registro

Finalmente, en el área Registro, los usuarios podrán ver el registro de incidencias del sistema. Los usuarios de tipo Agente no tienen derecho a esta funcionalidad, mientras que los *Registrars* podrán ver el registro, únicamente, las acciones (comandos *EPP*) que haya realizado. Los Administradores y Superusuarios tienen acceso al registro completo.

Figura C.4: Resolver un conflicto *IDN*

C.2. *Preppi*

Preppi es un cliente gráfico para *EPP*, el cual fue identificado como muy útil para trabajar con este protocolo. Sus características principales fueron detalladas en la Sección 6.7.

C.2.1. Dependencias

Se requieren los siguientes paquetes instalados en el sistema operativo:

- build-essential
- pkg-config
- libgconf2-dev
- libgnomeprintui2.2-dev
- libgtksourceview-dev

Figura C.5: Análisis de cambio en *DNSSEC*.

- libcurl3-openssl-dev
- libglade2-dev

Además, se deben instalar los siguientes módulos de *Perl*:

- ExtUtils::Depends
- ExtUtils::PkgConfig
- ExtUtils::MakeMaker
- Gnome2::GConf
- Gnome2::Print
- Gtk2::GladeXML
- Gtk2::GladeXML::Simple

- Gtk2::SourceView
- Gtk2::Ex::Simple::List
- Gtk2::Ex::Simple::Tree
- HTML::Tagset
- HTML::Entities
- XML::NamespaceSupport
- XML::SAX
- XML::LibXML
- Digest::SHA1
- Net::SSLeay
- IO::Socket::SSL
- Net::EPP
- URI::Escape

C.2.2. Instalación

Preppi puede obtenerse desde el directorio */bin/Herramientas* de la entrega, o puede descargarse desde *CentralNic*[1] empaquetado como *Tarball* o como *RPM*. Debido a que el entorno de trabajo seleccionado utiliza *Debian GNU/Linux*, se mostrará únicamente como instalarlo a partir del *Tarball*.

Se debe descomprimir el archivo antes indicado en algún directorio temporal, y ejecutar:

- ./configure
- make
- make install

[1] https://www.centralnic.com/company/labs/preppi/

Si se utiliza la versión que se encuentra en la entrega, no es necesario hacer nada más. Sin embargo, si se descarga el instalador desde el sitio de *CentralNic*, debido a que *Preppi* fue diseñado para ser utilizado sobre *SSL*, para poder utilizarlo con *OpenReg* es necesaria una pequeña modificación en su fuente. Luego de instalado, se debe editar el archivo */bin/preppi* comentando la línea 605:

```
$self->parse_peer_certificate;
```

C.2.3. Utilización

Al ejecutar el comando `preppi` se muestra la pantalla de login, como puede verse en la figura C.6. Deben ingresarse los datos del nombre del servidor (o dirección *IP*), puerto, usuario y contraseña. Puede especificarse el *timeout* deseado para la conexión (por defecto en 10 segundos) y si debe utilizarse o no *SSL* (para *OpenReg* dejar desactivado). La pantalla de login además permite cambiar la contraseña. Si se especificó una nueva contraseña además de todos los datos anteriores, el cambio se hará efectivo al momento de loguearse. Si no se especifica una nueva contraseña, se producirá un login normal.

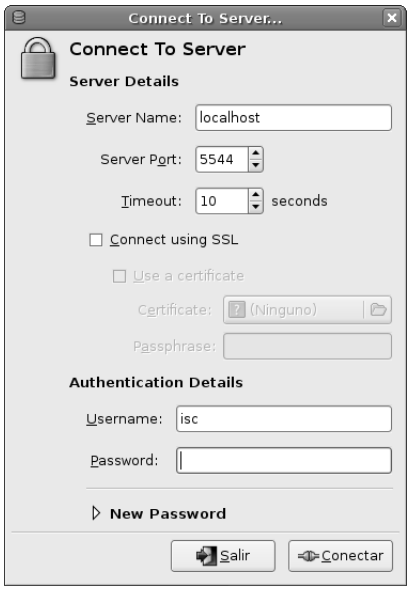


Figura C.6: Pantalla de login de *Preppi*

Una vez autenticado el usuario, se muestra la pantalla principal de la aplicación, la cual puede verse en la figura C.7. Se pueden distinguir 4 áreas principales dentro de esta ventana:

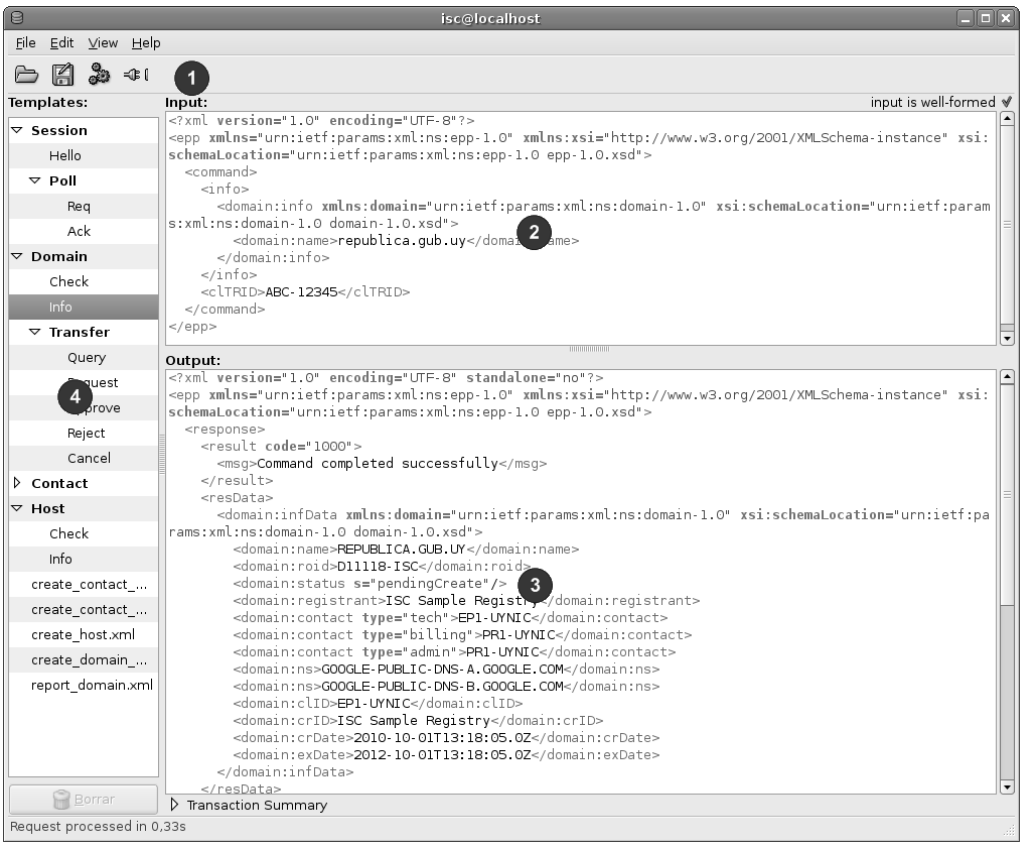


Figura C.7: Pantalla principal de *Preppi*

1. Menú y Barra de Herramientas: Contiene las funciones habituales, como ser abrir, guardar, enviar, salir, etc.

2. Área de *XML* de entrada: muestra el *XML* a enviar al servidor, que puede haber escrito el usuario o haber abierto desde un archivo. Inmediatamente arriba de la esquina superior derecha de esta área, se muestra un tick o una cruz acompañados del texto *input is (not) well formed* para indicarle al usuario si el *XML* está bien formado o no.

3. Área de *XML* de salida: Muestra el *XML* recibido del servidor. En la parte inferior cuenta con una solapa que permite consultar el historial de comandos enviados.

4. Árbol de Plantillas: Contiene la lista de plantillas disponibles con los comandos habituales, permitiendo ahorrar gran parte de la escritura. Las primeras son plantillas predefinidas, y el usuario puede agregar más desde la ventana de abrir *XML*, presionando el botón *Add Template*

El uso habitual de *Preppi* consiste en realizar el login, escribir los *XML* a enviar (opcionalmente abriéndolos de un archivo existente o desde una de las plantillas) y presionar el botón enviar. Cuando el servidor haya procesado la petición, se mostrará la salida. Se repetirá este proceso todas las veces que el usuario necesite, y se finalizará la sesión haciendo clic en el botón Salir.

Glosario

A

American Standard Code for Information Interchange (*ASCII*) Código utilizado por casi todos los ordenadores y sistemas para representar las letras, los números y los caracteres especiales. Este código asigna un valor alfanumérico a 128 números, utilizando 7 bits para cada uno de ellos. El código *ASCII* ampliado utiliza 8 bits, y puede representar 256 caracteres distintos. Se pronuncia **áski**.

B

Codificación Base64 (*Base64*) Sistema de numeración posicional que usa 64 como base. 64 es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de *ASCII*. Esto ha propiciado su uso para codificación de correos electrónicos, *PGP* y otras aplicaciones. Todas las variantes famosas que se conocen con el nombre de *Base64* usan el rango de caracteres A-Z, a-z y 0-9 en este orden para los primeros 62 dígitos, pero los símbolos escogidos para los últimos dos dígitos varían considerablemente de unas a otras.

C

Comprehensive Perl Archive Network (*CPAN*) Es un enorme archivo de software escrito en *Perl*, así como de documentación sobre el mismo. Está disponible a través del sitio www.cpan.org y sus 236 espejos distribuidos por todo el mundo.

D

Design Patterns GoF Los Patrones de Diseño, son soluciones de diseño a problemas comunes que pueden ser reutilizadas para solucionar

nuevos problemas similares. Se conoce por *GoF* (Pandilla de los Cuatro en inglés), a los autores del libro *"Design Patterns"*(ISBN 0-201-63361-2), de referencia en la programación orientada a objetos.

E

ENUM El Mapeo de Números Telefónicos a Nombres de Dominio (*ENUM*), es un protocolo que analiza el uso de *DNS* para almacenar números telefónicos.*ENUM* facilita servicios como *VoIP* (Voz sobre *IP*), y permite que los elementos de una red encuentren servicios en Internet usando un número telefónico. Está definido en las *RFC* 3761 y 2916.

F

Fully Qualified Domain Name (*FQDN*) Nombre de dominio que especifica la posición de un nodo en la jerarquía del espacio de nombres del *DNS*.

File Transfer Protocol (*FTP*) Protocolo para intercambiar archivos en Internet. Transfiere los datos con los protocolos de Internet *TCP/IP*, de la misma manera que el *HTTP* en la transferencia de páginas web desde un servidor al navegador de un usuario y el *SMTP* para transferir correo electrónico a través de Internet. Se utiliza principalmente para descargar un archivo de un servidor o para subir un archivo a un servidor a través de Internet.

G

General GNU Public License (*GNU GPL*) Licencia creada por la *FSF (Free Software Foundation)* en 1989 (la primera versión), orientada principalmente a proteger la libre distribución, modificación y uso de *software*. Su propósito es declarar que el *software* cubierto por esta licencia es *software* libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Existen varias licencias "hermanas" de la *GPL*, como la licencia de documentación libre de *GNU (GFDL)*, y otras menos restrictivas, como la *MGPL*, o la *LGPL (Lesser General Public License*, que permiten el enlace dinámico de aplicaciones libres a aplicaciones no libres.

I

Internet Assigned Numbers Authority (*IANA*) Entidad responsable de la coordinación global de la raíz *DNS*, asignación del espacio *IP*, y otros recursos de protocolos de Internet.

Internet Corporation for Assigned Names and Numbers (*ICANN*) Organización que opera a nivel internacional, siendo la responsable de asignar las direcciones del protocolo *IP*, de los identificadores de protocolo, de las funciones de gestión del sistema de dominio y de la administración del sistema de servidores raíz. *ICANN* se dedica a preservar la estabilidad de Internet por medio de procesos basados en el consenso. Coordina la administración de los elementos técnicos del *DNS* para garantizar la resolución univoca de los nombres, para que los usuarios puedan encontrar todas las direcciones.

Internet Engineering Task Force (*IETF*) Organización dedicada a la normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, ruteo, seguridad, etc. La *IETF* es mundialmente conocida por ser la entidad que regula las propuestas y los estándares de Internet, conocidos como *RFC*.

Internationalized Domain Name (*IDN*) Los nombres de dominio internacionalizados, son nombres de dominio que, potencialmente, contienen caracteres *no-ASCII*, como por ejemplo, eñes, caracteres orientales, cirílicos, griegos, etc.

Dirección IP (*IP*) Una dirección *IP* es una etiqueta numérica que identifica, de manera lógica y jerárquica, a una interfaz (elemento de comunicación/conexión) de un dispositivo (habitualmente una computadora) dentro de una red que utilice el protocolo *IP* (*Internet Protocol*), que corresponde al nivel de red del protocolo *TCP/IP*. El su versión 4 (*IPv4*) las direcciones son de 32 bits, mientras que en la 6 (*IPv6*) son de 128.

Internet Systems Consortium (*ISC*) Distribuidor de *software open source* para la comunidad de Internet. Se encarga de desarrollar *BIND* y otras implementaciones de referencia para diversos protocolos. También provee otros sistemas para dar soporte a la infraestructura de Internet (por ejemplo, *OpenReg* es un desarrollo originado por *ISC*).

J

Java Database Conectivity (*JDBC*) *API Java* para el acceso a bases de datos.

Java Message Service (*JMS*) Estándar de mensajería que permite a los componentes de aplicaciones basados en la plataforma *Java2* crear, enviar, recibir y leer mensajes. Hace posible la comunicación confiable de manera síncrona y asíncrona.

G

GNU Lesser General Public License (*LGPL*) *véase General GNU Public License.*

L

Licencia *BSD* Licencia de *software* otorgada principalmente para los sistemas *BSD* (*Berkeley Software Distribution*). Esta licencia tiene menos restricciones en comparación con otras como la *GPL* estando muy cercana al dominio público. La licencia *BSD*, al contrario que la *GPL*, permite el uso del código fuente en software no libre.

N

Network Information Center (*NIC*) Grupo de personas, entidad o institución encargada de asignar dominios de Internet bajo su dominio de red, sean genéricos o de países, a personas naturales o empresas que mediante un *DNS* pueden montar sitios de Internet en un proveedor de hospedaje.

O

OpenSSL (*OpenSSL*) Librería *open source* que implementa funciones criptográficas para ser utilizadas desde otros programas.

R

Request For Comments (*RFC*) Conjunto de documentos que sirven de referencia para la comunidad de Internet, que describen, especifican y asisten en la implementación, estandarización y discusión de la

mayoría de las normas, los estándares, las tecnologías y los protocolos relacionados con Internet y las redes en general.

S

Simple Authentication and Security Layer (*SASL*) *Framework* para autenticación y autorización en protocolos de internet. Separa los mecanismos de autenticación de los protocolos de la aplicación permitiendo, en teoría, a cualquier protocolo de aplicación que use *SASL* usar cualquier mecanismo de autenticación soportado por *SASL*. En la variante *PLAIN* las contraseñas se intercambian en texto plano (sin encriptar).

Simple API for XML (*SAX*) La *API* simple para *XML*, es una *API* de decodificación secuencial de *XML*. *SAX* proporciona un mecanismo para la lectura de datos de un documento *XML* orientado a eventos.

Sistemas *NIX Manera de denominar a los sistemas operativos *unix-like*. Los sistemas **NIX*, son sistemas que se comportan de manera similar a *Unix*, aunque no necesariamente se ajusten o estén certificados por alguna versión de la *Single UNIX Specification*.

Simple Mail Transfer Protocol (*SMTP*) Protocolo simple para la transferencia de correos electrónicos en Internet.

T

Transfer Control Protocol (*TCP*) Es uno de los principales protocolos en las redes *TCP/IP*. *TCP* le permite a dos computadoras anfitrionas establecer una conexión e intercambiar flujos de datos, garantizando la entrega de los datos y que los paquetes sean despachados en el mismo orden en que fueron enviados.

Modelo *TCP/IP* (*TCP/IP*) Es un modelo de descripción de protocolos de red creado en la década de 1970 por *DARPA*, una agencia del Departamento de Defensa de los Estados Unidos. Está estructurado en capas jerarquizadas. La capa de interred, que define como su protocolo oficial al *IP*, permite que los *hosts* inyecten paquetes en cualquier red y los hagan llegar a su destino. Sobre ésta, se encuentra la capa de transporte, cuyo cometido es permitir la conversación entre los extremos de una conexión. Define

dos protocolos oficiales: *TCP* y *UDP*. Lo servicios de la capa de transporte son utilizados por la capa de aplicación, que contiene los protocolos de alto nivel, como *DNS*, *HTPP*, *FTP*, etc.

U

User Datagram Protocol (*UDP*) Es un protocolo no orientado a conexión de la capa de transporte del modelo *TCP/IP*. *UDP* proporciona un servicio de datagramas sin conexión que ofrece "entrega de mejor esfuerzo", lo que significa que *UDP* no garantiza la entrega ni comprueba la secuencia de los datagramas.

UNICODE Estándar de codificación de caracteres diseñado para facilitar el tratamiento informático, transmisión y visualización de textos de múltiples lenguajes y disciplinas técnicas, además de textos clásicos de lenguas muertas.

UTF Formato de codificación de caracteres *UNICODE* e *ISO* 10646 utilizando símbolos de longitud variable.

Uniform Resource Identifier (*URI*) Cadena corta de caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente estos recursos son accesibles en una red o sistema.

Uniform Resource Identifier (*URL*) Secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos, presentaciones digitales, etc. Es un tipo particular de *URI*.

W

Web Services XML conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos como *XML*.

WHOIS Protocolo de aplicación de Internet que permite conocer información detallada acerca de la entidad/persona propietaria de un dominio, red o host de Internet.

X

Extensible Markup Language (*XML*) Es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados. Los datos se organizan en un árbol donde los nodos pueden tener atributos. A su vez, cada nodo puede tener un valor, si es hoja, o cualquier cantidad de sub-árboles.

Índice de figuras

2.1. Árbol de *DNS* 13
2.2. Archivo de zona en BIND 30

3.1. Arquitectura del SIGD 39

4.1. Máquina de estados del *EPP* 51
4.2. Proceso de transferencia de objetos 54
4.3. *EPP* aplicado a la provisión de Nombres de Dominio de Internet 63

5.1. Proceso de verificación para *registrants* del TLD .COOP . . . 67
5.2. Relación entre *DNSSEC*, *EPP* y registros DS 70
5.3. Diagrama de estados del *RGP* 81

6.1. Módulos *OpenReg* 86
6.2. Ejemplo de entorno *CoCCA* 89
6.3. Módulos *CoCCA* 90
6.4. Módulos *FRED* 92
6.5. *EPP* utilizado por *FRED* 93

7.1. Implementación del Manejo de Dominios Críticos 111
7.2. Modelo de dominio para registro IDN 118
7.3. Extensión del objeto dominio para la gestión de información *DNSSEC* 120
7.4. Chequeo de una acción de transformación 123
7.5. El *agente ejecutor* cumple una tarea pendiente. 124
7.6. El *Agente ejecutor* debe consultar al *registry* sus tareas pendientes. 124
7.7. Modelo de datos del *registry* extendido para poder completar las tareas de verificación 128
7.8. Componentes involucrados en el proceso de validación de solicitudes 131
7.9. Gestión del Período de Gracia 139

7.10. Implementación de la gestión del Período de Gracia 144
7.11. Extensión del objeto dominio para la gestión del Período de Gracia 145

8.1. Actores de los Casos de Uso 151
8.2. Casos de Uso para usuario Web 151
8.3. Casos de Uso para usuario Administrador 152
8.4. Casos de uso para Usuario Registrador 153
8.5. Casos de uso para Usuario Ejecutor 154
8.6. Diagrama de secuencia para resolución de conflicto IDN 156
8.7. Creación del dominio 158
8.8. Ejecución de una tarea asociada a un ticket 158
8.9. Tipos de datos de *ARCO* 160
8.10. Tipos de datos de *EPP-RTK* 160
8.11. Vista lógica y de componentes 161
8.12. Vista de subsistemas para el componente de negocios 162
8.13. Vista de implementación y despliegue 164
8.14. Interfaz exigida a los Agentes Legisladores 171
8.15. Tipo de datos relativo al estado de validación 172
8.16. Interfaz para agentes ejecutores 172

B.1. Configuración de librerías en *Eclipse* 247

C.1. Formulario para crear usuarios *ARCO* 253
C.2. Confirmar cambios de criticidad 255
C.3. Contestar una tarea 256
C.4. Resolver un conflicto *IDN* 257
C.5. Análisis de cambio en *DNSSEC*. 258
C.6. Pantalla de login de *Preppi* 260
C.7. Pantalla principal de *Preppi* 261

Índice de cuadros

2.1. Ejemplo de un registro *A* 16
2.2. Ejemplo de un registro *AAAA* 16
2.3. Ejemplo de un registro *CNAME* 16
2.4. Ejemplo de un registro *NS* 17
2.5. Ejemplo de un registro *PTR* 17
2.6. Registro *DNSKEY* 20
2.7. Detalle de los campos de un registro *DNSKEY* 21
2.8. Registro *RRSIG* 21
2.9. Detalle de los campos de un registro *RRSIG* 22
2.10. Registro *NSEC* 23
2.11. Detalle de los campos de un registro *NSEC* 23
2.12. Ejemplo de un registro *NSEC* 23
2.13. Registro *DS* 24
2.14. Detalle de los campos de un registro *DS* 24
2.15. Ejemplo de un registro *DS* 25
2.16. Ejemplo de un registro *DNSKEY* correspondiente al registro *DS* 25

4.1. Categorías de resultados según código 56
4.2. Códigos de resultado y descripciones 57

5.1. Respuesta al comando *info* para un nombre con información *DNSSEC* 71
5.2. Respuesta al comando *info* para un nombre con información *DNSSEC* y nodos opcionales. 72
5.3. Comando *create* para un nombre con información *DNSSEC* . 73
5.4. Comando *create* para un nombre con información *DNSSEC* y nodos opcionales 74
5.5. Extensión al comando *update* para agregar información *DNSSEC* 75
5.6. Extensión al comando *update* para quitar información *DNSSEC* 76

5.7. Extensión al comando *update* para modificar información *DNSSEC* de manera urgente. 76
5.8. Extensión al comando *update* para modificar información *DNSSEC* de manera urgente con información opcional. 77

7.1. Pedido de registro nombre IDN 113
7.2. Errores en el registro de un nombre *IDN* 114
7.3. Solicitud de chequeo de nombre internacionalizado 115
7.4. Respuesta a chequeo de nombre internacionalizado 116
7.5. Caracteres *no-ASCII* soportados por el registro .UY 117
7.6. Criterios de equivalencia 117
7.7. Algoritmo de similitud 118
7.8. Algoritmo de registro de nombres 119
7.9. Respuesta extendida a un comando transformador, que incluye un *ticket* y una lista de pendientes 126
7.10. Mensaje enviado al *registrar* cuando se completa un paso del proceso de verificación de una solicitud (sólo se incluyen los nodos más relevantes 129
7.11. Mensaje enviado al *registrar* cuando culmina el proceso de verificación de una solicitud 130
7.12. Ejemplo de solicitud de *report* 133
7.13. Respuesta exitosa a un comando *report* 133
7.14. Solicitud de reporte de objetos según condiciones 134
7.15. Ejemplo de solicitud de reporte de objetos según condiciones para dominios 135
7.16. Solicitud de reporte de tipo consulta nombrada 136
7.17. Solicitud de reporte de tipo consulta libre 138
7.18. Extensión para mostrar el período de gracia en el comando `<info>` 140
7.19. Mensaje enviado al *registrar* cuando se da la baja automática de un dominio 142
7.20. Mensaje enviado al *registrar* cuando se da la baja definitiva de un dominio 143
7.21. Posible modelo de datos 147

10.1. Extensiones realizadas a *EPP* 187