“Arquitectura de Referencia para Anonimizar Documentos”
Prototipo: Aplicación DEMO

Anexo C: Documento de Arquitectura Complementario

Ing. Horacio Vico
horacio.vico@gmail.com
Versión 6.0 - Octubre de 2013.

Historial de Versiones

Versión	Fecha	Autor	Comentarios
1.0	Abril de 2013	Horacio Vico	Versión inicial.
2.0	Mayo de 2013	Horacio Vico	Versión corregida luego de revisión de D. Calegari.
3.0	Junio de 2013	Horacio Vico	Revisión general de formato.
4.0	Junio de 2013	Horacio Vico	Versión corregida luego de segunda revisión de D. Calegari.
5.0	Julio de 2013	Horacio Vico	Versión anexo para documento principal.
6.0	Octubre de 2013	Horacio Vico	Versión luego de revisión de A. Delgado

Índice

Índice de figuras

1. Introducción

1.1. Objetivo y Alcance

El presente documento presenta aspectos específicos de la arquitectura de la aplicación prototipo desarrollada como instancia de la Arquitectura de Referencia propuesta en el presente trabajo.
La arquitectura de la aplicación está modelada siguiendo los lineamientos de la propuesta genérica presentada en el S.A.D., por tanto la información que aquí se presenta es complementaria. El mencionado documento (S.A.D.) se considera parte integral de la documentación de la arquitectura de la aplicación prototipo.

2. Contexto del Sistema

2.1. Entorno del Sistema

Existe un sistema de gestión documental denominado "Base de Jurisprudencia Nacional" (BJN), el cual almacena las sentencias judiciales de Tribunales de Apelaciones y la Suprema Corte de Justicia de la República Oriental del Uruguay. Mediante una interfaz el sistema Aplicación DEMO, deberá interoperar con el sistema BJN para obtener los documentos a anonimizar, y luego de procesarlos almacenarlos en la propia base de datos del sistema BJN. Como actor se visualiza a los funcionarios del Dpto. de Jurisprudencia del Poder Judicial, quienes son los encargados de realizar el proceso de anonimización de las sentencias judiciales, para que puedan ser publicadas al público general.

Tomando como referencia el diagrama de contexto (Figura 1) que se presenta en el documento SAD de la arquitectura de referencia, se puede establecer la siguiente correspondencia:

1. El sistema de anonimización es concretamente la Aplicación Demo que se describe en el presente documento.
2. En este caso la interfaz se establece con una base documental, la base de datos del sistema BJN
3. El sistema de gestión documental es la Base de Jurisprudencia Nacional (BJN)
4. El experto del dominio son los técnicos jurídicos del Departamento de Jurisprudencia del Poder Judicial.

2.1.1. Stakeholders (interesados)

Los siguientes actores surgen como stakeholders del sistema prototipo Aplicación DEMO.

- **Usuarios potenciales del sistema**: Tal como se visualizó en la descripción del contexto del sistema, los funcionarios del Dpto. de Jurisprudencia del Poder Judicial surgen como primeros interesados en la Aplicación DEMO. Actualmente el proceso de anonimización se basa en el trabajo directo de estos funcionarios, quienes deben leer las sentencias y anonimizarlas manualmente. De existir una herramienta que automatice o asista en este proceso estos funcionarios serían los primeros beneficiarios.
- **Desarrolladores:** Los técnicos informáticos del Poder Judicial pueden valerse del sistema aquí descrito para contar con una primera aproximación a un sistema de anonimización integrable con la Base de Jurisprudencia Nacional.

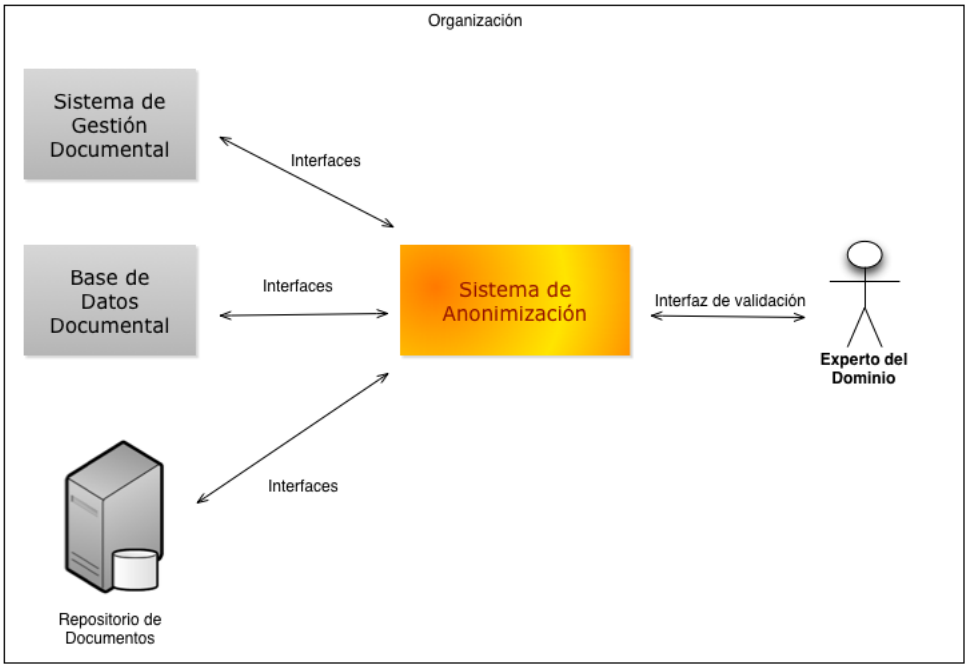

Figura 1: Diagrama de Contexto

- **Usuarios del servicio:** Los usuarios de la Base de Jurisprudencia Nacional (ciudadanía en general), podrá beneficiarse de existir un proceso automatizado de anonimización, dado que el tiempo de publicación de las sentencias se vería reducido sensiblemente.

2.2. Visión General de los Requerimientos

Se enumeran los requerimientos funcionales y no funcionales identificados para la Aplicación DEMO. Como notación se agrega la letra E a cada requerimiento específico de la Aplicación, de manera que se diferencien de los requerimientos genéricos identificados en la arquitectura de referencia descrita en el S.A.D. Cuando existe una vinculación de cada requerimiento con un requerimiento genérico, se especifica la misma en la columna "Req.Arq.Ref." de la tabla.

Referencia	Req. Arq. Ref.	Descripción del Requerimiento
RFE1	RF1	El sistema debe poder procesar sentencias judiciales almacenadas en el sistema Base de Jurisprudencia Nacional.
RFE2	RF3	El sistema debe permitir a los funcionarios de jurisprudencia validar las sentencias anonimizadas, y aprobar el documento o reprobarlo brindando feedback que retroalimente al sistema.
RFE3	RF7	El sistema debe almacenar el documento anonimizado en la Base de Jurisprudencia Nacional
RFE4	-	El sistema debe permitir ejecutar la anonimización de múltiples sentencias en un solo paso.
RFE5	RF2, RF4 y RF5	El sistema debe permitir, de forma configurable, un funcionamiento cien por ciento automático, partiendo de la selección de la/s sentencia/s a anonimizar, y almacenando las mismas en la Base de Jurisprudencia Nacional.
RFE6	RF6	El sistema debe permitir registrar exclusiones de términos jurídicos específicos que se presentan en las sentencias y que pueden generar falsos positivos en las herramientas NER.
RNFE1	-	Mantenibilidad: El sistema deberá desarrollarse sobre plataforma JAVA, herramienta de uso por parte de los técnicos del Poder Judicial, de forma que el sistema sea mantenible por los mismos.

2.3. Escenarios del Sistema

En esta sección se listarán algunos escenarios funcionales y no funcionales específicos.

2.3.1. Escenarios Funcionales

Referencia	ESRFE1
Requerimiento vinculado	RFE1
Visión General	Cómo el sistema obtiene sentencias judiciales almacenadas en el sistema Base de Jurisprudencia Nacional.
Estado del Sistema	Normal
Entorno del Sistema	El entorno del sistema opera normalmente.
Estímulo externo	El usuario selecciona un documento almacenado en la Base de Jurisprudencia Nacional, para ser anonimizado.
Respuesta	El sistema debe ser capaz de obtener el documento desde la BJN para su procesamiento.

Referencia	ESRFE2
Requerimiento vinculado	RFE2
Visión General	Cómo el sistema permite a los funcionarios de jurisprudencia validar las sentencias anonimizadas, y aprobar el documento o reprobarlo brindando feedback que retroalimenta al sistema.
Estado del Sistema	Normal
Entorno del Sistema	El entorno del sistema opera normalmente.
Estímulo externo	Una sentencia procedente de la BJN es anonimizada y presentada al usuario de jurisprudencia.
Respuesta	El funcionario de jurisprudencia puede aprobar o rechazar el documento mediante controles específicos. En caso de rechazo se permite definir un patrón o regla para contemplar alguna casuística particular, a modo de feedback.

Referencia	ESRFE3
Requerimiento vinculado	RFE3
Visión General	Cómo el sistema almacena el documento anonimizado en la Base de Jurisprudencia Nacional
Estado del Sistema	Normal
Entorno del Sistema	El entorno del sistema opera normalmente.
Estímulo externo	El funcionario de jurisprudencia aprueba un documento anonimizado.
Respuesta	El sistema Aplicación DEMO almacena la sentencia anonimizada en la Base de Jurisprudencia Nacional mediante una interfaz adecuada.

Referencia	ESRFE4
Requerimiento vinculado	RFE4
Visión General	Cómo el sistema permite ejecutar la anonimización de múltiples sentencias en un solo paso.
Estado del Sistema	Normal
Entorno del Sistema	El entorno del sistema opera normalmente.
Estímulo externo	El usuario selecciona múltiples sentencias a anonimizar en el sistema Aplicación DEMO.
Respuesta	Las sentencias son anonimizadas una a una. En caso de que el sistema esté en modo cien por ciento automático tan solo se le presentará al usuario el resultado de la anonimización. En otro caso se irán procesando las sentencias seleccionadas una a una secuencialmente, solicitando la información que se requiera en cada paso.

Referencia	ESRFE5
Requerimiento vinculado	RFE5
Visión General	Cómo el sistema permite, de forma configurable, un funcionamiento cien por ciento automático, partiendo de la selección de la/s sentencia/s a anonimizar, y almacenando las mismas en la Base de Jurisprudencia Nacional.
Estado del Sistema	Normal
Entorno del Sistema	El entorno del sistema opera normalmente.
Estímulo externo	El usuario selecciona sentencias a anonimizar, y configura el modo cien por ciento automático del sistema, desde la interfaz de configuración.
Respuesta	Se procesan todas las sentencias seleccionadas sin pedir ningún tipo de información al usuario. Finalizado el proceso, se le presentan al usuario las sentencias anonimizadas, las cuales son automáticamente almacenadas en la BJN.

Referencia	ESRFE6
Requerimiento vinculado	RFE6
Visión General	Cómo el sistema permite registrar exclusiones de términos jurídicos específicos que se presentan en las sentencias y que pueden generar falsos positivos en las herramientas NER.
Estado del Sistema	Normal
Entorno del Sistema	El entorno del sistema opera normalmente.
Estímulo externo	El experto de dominio en la interfaz de revisión detecta que fue marcada como información a anonimizar el término CONSIDERANDO, que habitualmente se presenta en mayúsculas en las sentencias.
Respuesta	El usuario accede a una interfaz de configuración del sistema, define una nueva regla por la cual el término CONSIDERANDO no será considerado como nombre a anonimizar.

3. Vistas de la Arquitectura

3.1. Vista Funcional

Como se explicó en el SAD de la arquitectura de referencia, en la Vista Funcional se introducen los diferentes elementos funcionales del sistema de anonimización, sus responsabilidades y la forma en que se comunican entre sí. Habiéndose modelado el sistema como un proceso de negocios BPMNv2[1], se presentará el sistema en términos de dicho estándar, definiendo las tareas con sus entradas y salidas, así como las variables que condicionan el proceso.

3.1.1. Proceso

Como se explicó anteriormente, la Arquitectura de Referencia modela el funcionamiento del sistema de anonimización como un proceso de negocios. Siguiendo esta pauta, el proceso de la aplicación DEMO fue modelado utilizando el motor opensource Bonita Open Solution[2].

Uno de los puntos más importantes a destacar, es que el proceso de la Aplicación DEMO se define por sobre el proceso de Anonimización de Documentos tal cual fue diseñado en la arquitectura de referencia. Tal es así, que en primer lugar fue implementado el proceso BPMN principal definido en el SAD de la Arquitectura de Referencia, con sus tres subprocesos (Reconocer Entidades con Nombre, Agrupar Entidades con Nombre y Anonimizar Documento) sobre el motor Bonita. Una vez dicho procesos y sus correspondientes subprocesos fueron desarrollados y probados (validando de forma empírica además el proceso genérico definido), se implementó por sobre ellos un proceso adicional, que llamaremos Aplicación DEMO, el cual modela la instanciación del sistema concreta que se define en este SAD específico.

En la Figura 2 se puede visualizar el proceso Aplicación DEMO, que utiliza el proceso de la Arquitectura de Referencia como núcleo. Seguidamente se describen las entradas, salidas y responsabilidades de las distintas tareas específicas del proceso “Aplicación DEMO”.

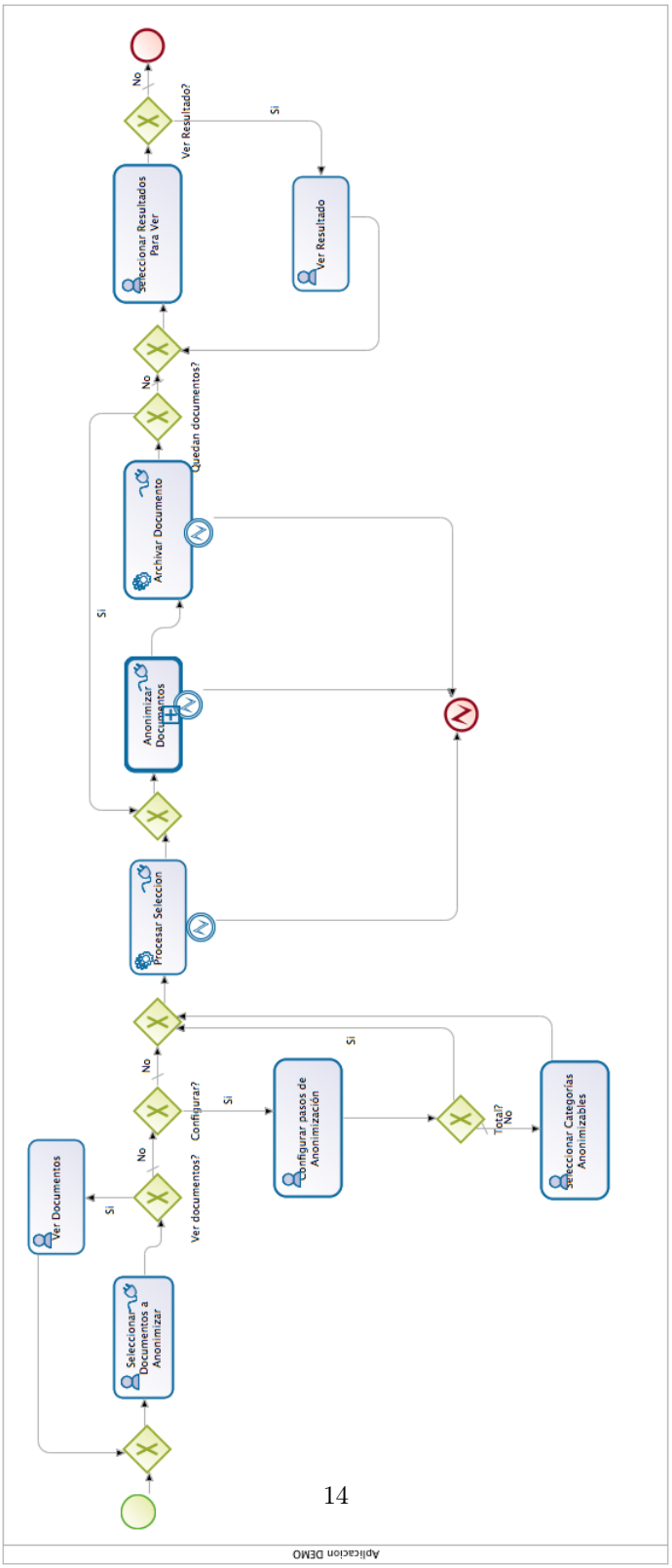

Figura 2: Proceso aplicación DEMO

Tareas

Nombre	**Seleccionar Documentos a Anonimizar**
Responsabilidades	En esta tarea el funcionario selecciona una o varias sentencias de la Base de Jurisprudencia Nacional a anonimizar.
Entrada	No tiene (Inicio del proceso)
Salida	Sentencias a ser anonimizadas seleccionadas en el sistema.

Nombre	**Ver Documentos**
Responsabilidades	Esta tarea es opcional (se activa desde la tarea Seleccionar Documentos a Anonimizar), y su cometido es presentar al usuario el contenido de las sentencias seleccionadas previamente.
Entrada	Un conjunto de documentos seleccionados, y la opción "Ver documentos" activada.
Salida	No tiene. Retorna a la tarea "Seleccionar Documentos a Anonimizar" para que el usuario realice los cambios de selección que crea convenientes.

Nombre	**Configurar pasos de Anonimización**
Responsabilidades	Configuración del sistema. Entre otras opciones aquí se puede establecer que el proceso sea cien por ciento automático.
Entrada	Conjunto de documentos seleccionados para anonimizar.
Salida	Parámetros y variables de configuración del sistema actualizadas si corresponde. Se retorna a la tarea "Seleccionar Documentos a Anonimizar".

Nombre	**Procesar Selección**
Responsabilidades	Este componente inicializa una pila (stack) con el conjunto de sentencias a procesar.
Entrada	Conjunto de documentos seleccionados para anonimizar.
Salida	Stack inicializado con la lista de sentencias a procesar.

Nombre	Anonimización (Subproceso)
Responsabilidades	Este subproceso es el que se describe en el documento S.A.D., es decir es el proceso de anonimización definido para la Arquitectura de Referencia.
Entrada	Un documento del stack con una sentencia a procesar.
Salida	La sentencia anonimizada

Nombre	Guardar documento en base de datos
Responsabilidades	Guarda en la BJN la sentencia anonimizada
Entrada	Una sentencia anonimizada
Salida	La sentencia guardada en la base de datos. Si quedan sentencias en el stack retorna al subproceso anonimización. De otra forma se continúa hacia la tarea Seleccionar Resultado para Ver.

Nombre	Seleccionar Resultado para Ver
Responsabilidades	En esta tarea se le permite al usuario seleccionar una de las sentencias anonimizadas para su posterior visualización.
Entrada	Sentencias ya anonimizadas.
Salida	Sentencias a visualizar.

Nombre	Ver Resultado
Responsabilidades	Mostrar el documento anonimizado al usuario.
Entrada	Documento anonimizado seleccionado.
Salida	Ninguna. Retorna a la tarea Seleccionar Resultado para Ver.

Parámetros de Configuración

El proceso Aplicación DEMO maneja una serie de variables y parámetros para representar las distintas configuraciones que son admitidas por el sistema. Muchos de los parámetros de configuración, tienen un correspondiente mapeo en el subproceso "Anonimización de un Documento".

Particularmente, los parámetros de configuración que pueden ser seleccionados por el usuario en Aplicación DEMO en las tareas "Configurar pasos de Anonimización" y "Seleccionar Categorías Anonimizables", son mapeados con sus correspondientes variables del subproceso "Anonimización de un Documento". Por su parte, la selección del documento a anonimizar ocurre sólo en Aplicación DEMO, y ésta establece automáticamente la variable "Texto" del subproceso "Anonimizar un Documento" al invocarlo. Es decir que los parámetros que el usuario selecciona en Aplicación DEMO son automáticamente configurados en el subproceso, evitándose así cualquier duplicación de tareas. Para no generar

esa redundancia y que la configuración sólo ocurra en el proceso Aplicación DEMO, se establece en verdadero la bandera booleana "Automático" definida en el proceso "Anonimización de un Documento".

Para ilustrar todo esto con mayor claridad, cabe repasar el proceso Anonimización de un Documento que fuera diseñado en la Arquitectura de Referencia. El mismo se presenta en la Figura 3. Allí se puede ver cómo las tareas "Ingresar Documento", "Configurar pasos de Anonimización" y "Configurar Categorías Anonimizables", están todas condicionadas por la compuerta que valida la variable "Automático". Es decir que estas tareas pueden ser omitidas al configurarse dicha bandera, y tal es la estrategia utilizada por Aplicación DEMO.

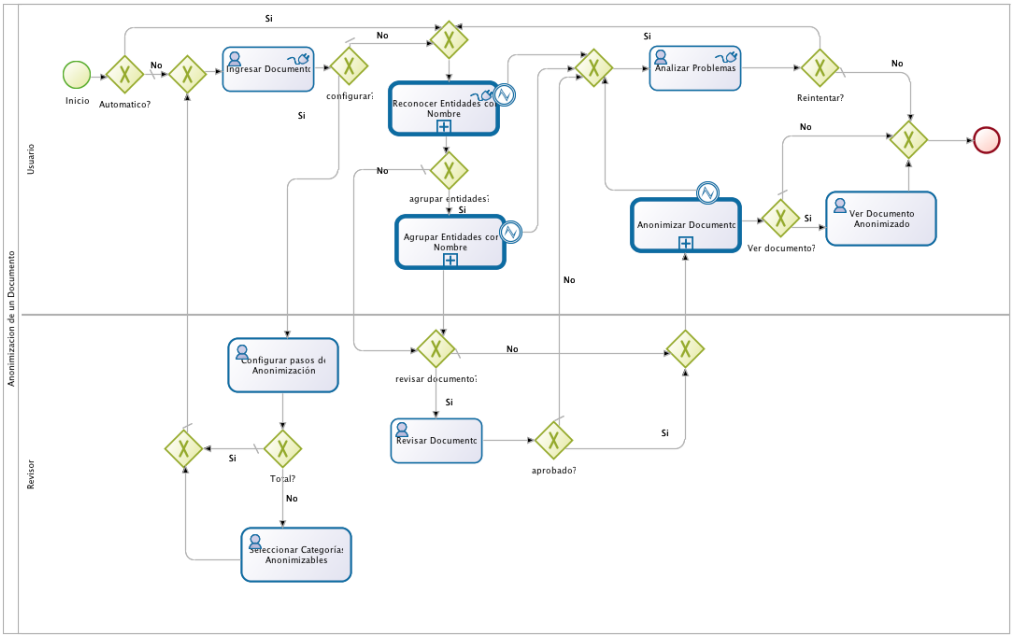

Figura 3: Proceso Anonimización de un Documento de la Arquitectura de Referencia

En la tabla siguiente se detallan los parámetros y variables de Aplicación DEMO, mencionando si la misma tiene una correspondencia (mapeo) en el subproceso "Anonimización de un Documento". Como convención las variables del proceso son homónimas en el subproceso.

Nombre	Tipo	Mapeada	Parámetro
Texto	JAVA: model.Text	Si	El documento a anonimizar
pilaDeTrabajo	JAVA: java.util.Stack	No	Estructura de pila que contiene los documentos a ser procesados por el sistema.
Automático	Booleano	Si	Define si el proceso de anonimización será automático o será supervisado.
Agrupar	Booleano	Si	Define si se deberán agrupar (clustering) las entidades con nombre.
Reversible	Booleano	Si	Define si la anonimización será reversible o irreversible.
Total	Booleano	Si	Define si la anonimización será parcial (false) o total (true).
heramientaNER	Texto	Si	Nombre de la herramienta NER que se utilizará.
herramientaClustering	Texto	Si	Nombre de la herramienta de clustering que se utilizará.
heuristicas	Booleano	Si	Define si se procesará el documento mediante reglas heurísticas, reglas y patrones.
Otra_herramienta	Booleano	Si	Define si se deberán invocar herramientas externas adicionales.

3.2. Vista de Información

En esta vista se detallarán las estructuras de datos específicas del sistema Aplicación DEMO, y las interfaces de datos con el sistema BJN.

3.2.1. Estructura de Datos

Las estructuras utilizadas en el sistema para modelar el documento a anonimizar es la descrita en el documento S.A.D. En la Figura 4 se presenta un diagrama para reflejar este modelo instanciado, donde se pueden visualizar algunas adiciones al modelo abstracto presentado en la arquitectura de referencia, a saber:

1. Se definió un modelo de NEClass específico para el motor NER OpenCalais, a través de la clase OpenCalaisNEClass que se visualiza en el diagrama. Esta clase agrega algunas categorías de Entidades con Nombre que aporta el motor OpenCalais, el cual es el más vasto en este sentido de las herramientas utilizadas con este fin.

2. La clase NamedEntity adiciona algunos servicios, con el fin de simplificar entre otras cosas su agrupamiento. Destaca un método isAcronym para determinar si una Named Entitiy es acrónimo de otra.

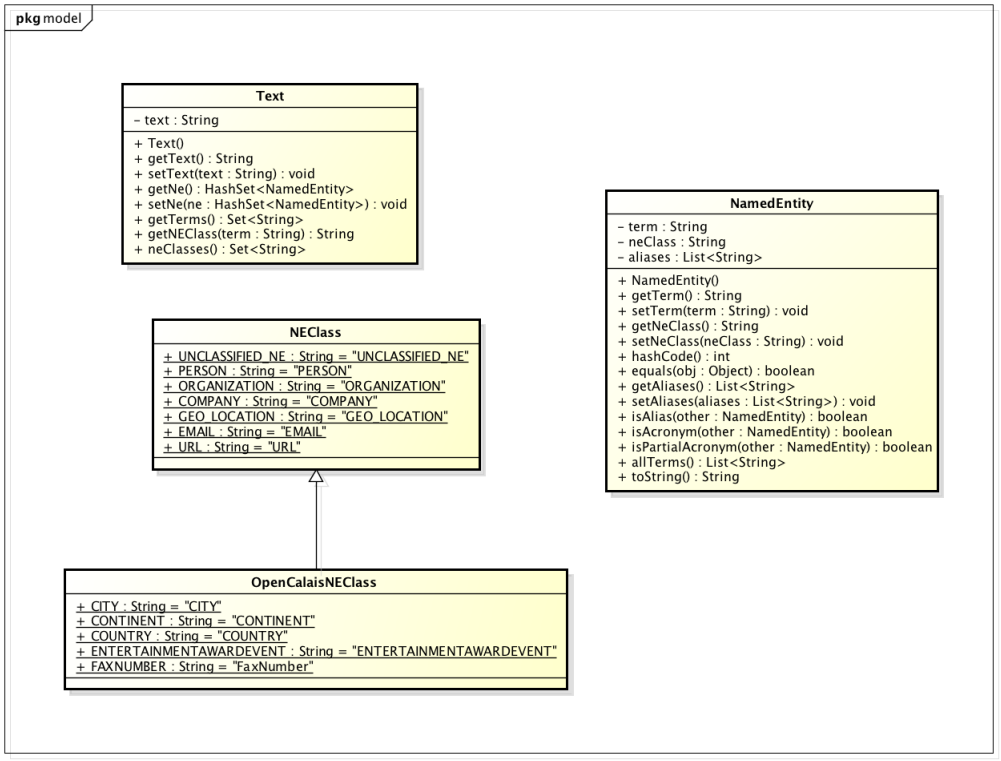

Figura 4: Modelo de Datos

3.2.2. Modelo Relacional e Interfaz con Base de Jurisprudencia Nacional

Como interfaz con el sistema Base de Jurisprudencia Nacional, se definió una tabla de acceso común entre ambos sistemas sobre una base de datos MySQL[3] 5.5. La estructura de la tabla se describe en la Figura 5. De la lista de campos, los relevantes a los efectos del sistema de anonimización son los siguientes:

1. id: Clave primaria

2. texto: Contiene el texto de la sentencia (documento a anonimizar).

3. textoSensible: En este campo se almacena el texto una vez la sentencia es anonimizada.

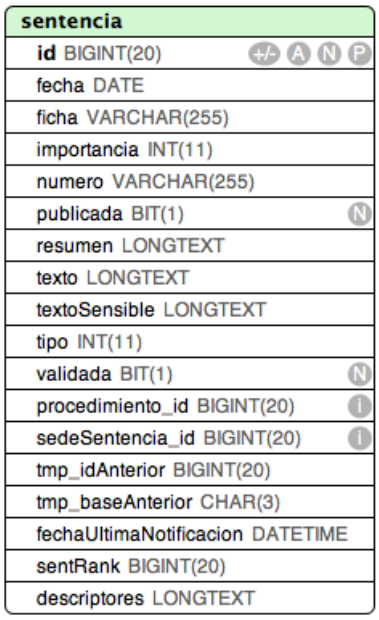

Figura 5: Tabla Sentencia de la base BJN

El acceso a la base de datos se establece mediante la JDBC, utilizando el driver para MySQL en su versión: mysql-connector-java-5.1.24-bin.jar.

3.2.3. Persistencia de reglas heurísticas

Para persistir las reglas heurísticas que se pueden definir dinámicamente en Aplicación Demo, se maneja una pequeña base de datos MySQL llamada "Anonimizacion", donde simplemente se tiene una tabla para representar cada una de estas reglas. La estructura de la tabla se puede visualizar en la Figura 6.

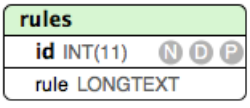

Figura 6: Tabla Rules de la base Anonimizacion

3.3. Vista de Despliegue

En esta sección se explica cómo se despliegan y cómo se comunican los distintos componentes del sistema para su ejecución.

3.3.1. Modelo de la plataforma de ejecución

La Figura 7 presenta los distintos componentes que ejecutan en Aplicación DEMO, dentro de sus respectivos contenedores de software y hardware.

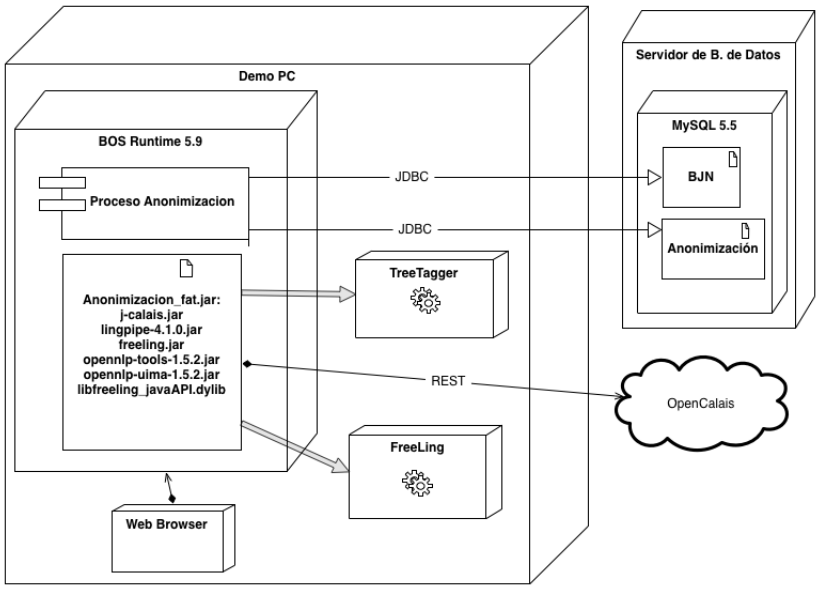

Figura 7: Modelo de Despliegue

3.3.2. Descripción de los nodos y componentes

- **Demo PC**: El PC o servidor donde se ejecuta el sistema Aplicación Demo.
- **BOS Runtime 5.9**: Motor de procesos Bonita Open Solution 5.9
- **Proceso Anonimización**: Es el proceso BPM de la aplicación demo, definido para su ejecución en BOS Runtime 5.9.
- **Anonimización_fat.jar**: Archivo JAR que contiene todas las clases y dependencias JAVA de la Aplicación Demo empaquetadas. Para simplificar el despliegue se empaquetan todas las librerías y dependencias junto con los paquetes JAVA desarrollados, utilizando el plugin para Eclipse "Fat Jar". Dentro del JAR se empaquetan los frameworks OpenNLP y LingPipe basados en JAVA, así como las APIs para interactuar con herramientas

externas (TreeTagger y FreeLing). También se incluye en el JAR la API (j-Calais) para dialogar vía REST con los web services de OpenCalais.

- **TreeTagger**: Instalación local ejecutable de la herramienta TreeTagger.
- **FreeLing**: Instalación local ejecutable de la herramienta FreeLing.
- **Web Browser**: Navegador web utilizado para acceder a la GUI por defecto de BOS 5.9 (Bonita User Experience).
- **Servidor de B. de Datos**: Servidor donde se encuentra alojada la base de datos del sistema BJN, sobre el motor RDBMS MySQL 5.5. Puede ejecutarse en el mismo servidor/PC que Aplicación Demo.

3.3.3. Dependencias de Software

El sistema Aplicación Demo fue probado de forma exitosa sobre los siguientes sistemas operativos:

- Ubuntu Linux 12.10 64-bit
- Mac OS X 10.8 "Moutain Lion"

La plataforma BOS 5.9 se podría ejecutar también en plataforma Microsoft Windows. Sin embargo se deben tener en cuenta las siguientes consideraciones al cambiar de plataforma:

1. Las herramientas externas TreeTagger y FreeLing son dependientes de la plataforma, ya que se basan en binarios compilados para cada sistema operativo. Existen versiones de dichas herramientas para las tres plataformas mencionadas, Linux, Windows y Mac OS X. Una vez instaladas dichas herramientas se deberían configurar las rutas que correspondan en los wrappers correspondientes dentro del sistema Aplicación Demo.

2. La API JAVA herramienta FreeLing además utiliza una librería nativa a la cual se accede por la vía de JNI. Dicha librería debe ser compilada en cada plataforma. El sistema Aplicación DEMO tiene incorporadas las versiones Linux 64-bit y Mac OS X 10.8 de dicha librería pre-compiladas.

Dependencias Adicionales JAVA

Para el desarrollo de los componentes JAVA del sistema Aplicación Demo, se incorporaron diversas librerías de terceros para proveer servicios y funcionalidades genéricas y reutilizables. A continuación se listan las librerías que se incorporan en el JAR de Anonimización_fat.jar:

1. Apache Commons: Biblioteca de componentes JAVA reutilizables. Se utiliza para la gestión de la configuración XML, a través de librería commons-configuration.

2. freeling.jar: API JAVA de FreeLing

3. j-calais.jar: API JAVA para acceso a OpenCalais
4. log4j: Librería que brinda servicios para log/auditoría.
5. opennlp: Framework Apache OpenNLP para el procesamiento de lenguaje natural.
6. lingpipe.jar: Framework LingPipe
7. org.annolab.tt4j: API JAVA para TreeTagger

3.4. Vista Operacional

En esta sección se explica cómo se instala y ejecuta el sistema, teniendo en cuenta los distintos componentes detallados en la Vista de Despliegue precedente.

3.4.1. Instalación del Sistema

El sistema Aplicación Demo se distribuye en el formato de una máquina virtual versión 9 de VMWare, con el software necesario para ejecutarse preinstalado. Por tanto la instalación implica instalar el software VMWare Player 5 o superior (disponible para plataforma Windows, y Linux en forma gratuita, o el software comercial VMWare Fusion para Mac). La máquina virtual también es compatible con los productos comerciales VMWare ESXi 5.1 y Workstation 9.x.

Las últimas versiones del software gratuito VMWare Player para Linux y Windows a la fecha, se distribuyen con el sistema Aplicación Demo, en la carpeta "VMWARE" del disco de instalación del sistema.

La máquina virtual se encuentra en el directorio VM_APLICACION_DEMO.

Para instalar el sistema se deberá copiar el contenido de dicho directorio en alguna ubicación del disco duro, y luego abrir dicha máquina con el software VMWARE que se tenga instalado (o con el VMWARE Player que se incluye en el disco).

3.4.2. Software Incluido

La máquina virtual incluye el siguiente software de base preinstalado:

- Ubuntu Linux Desktop 12.10 "Quantal Quetzal" 64-bit
- Oracle JDK 1.6.0_45
- Bonita Open Solution (Studio) 5.9
- Eclipse "Juno" 4.2
- FreeLing 3.0 (/usr/local/share/freeling)
- TreeTagger 3.2 (/usr/local/treetagger)

- MySQL Server 5.5
- MySQL Workbench 5.2.47

Se distribuye la Aplicación Demo como un proyecto Bonita Open Solution, el cual se debe iniciar desde el BOS Studio.

Los fuentes del sistema Aplicación Demo, se encuentran accesibles en el IDE Eclipse Juno que se encuentra en el Escritorio de Ubuntu.

3.4.3. Inicio del Sistema Aplicación Demo

A continuación se describen los primeros pasos necesarios para iniciar el sistema Aplicación Demo. El objetivo de esta sección no es el de explicar el funcionamiento y operación del sistema, sino dar las indicaciones necesarias para iniciar el ambiente del mismo.

Una vez iniciada la máquina virtual, se debe acceder a la carpeta BOS 5.9 que se encuentra disponible en el escritorio de Ubuntu, como se puede apreciar en la Figura 8.

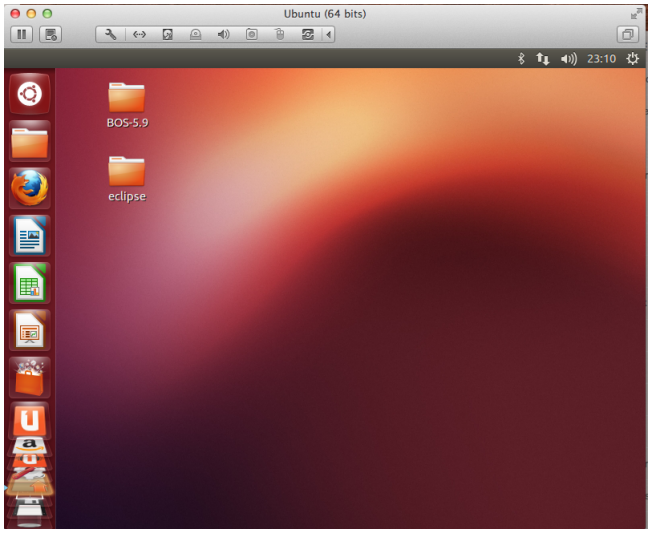

Figura 8: Escritorio de Ubuntu

Se deberá iniciar el script ejecutable BonitaStudio.sh, como se aprecia en la Figura 9. Seleccionar la opción “Ejecutar”.

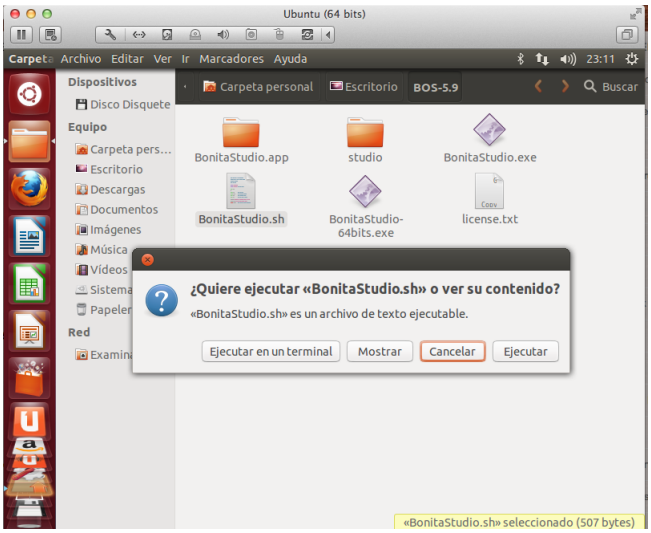

Figura 9: Ejecutar script

Dentro del software Bonita Studio, seleccionar la opción “Abrir”, para que el sistema presente el conjunto de procesos “AnonimizaciónDeDocumentos(1.0)”, el cual se deberá seleccionar y abrir, como se aprecia en la Figura 10.

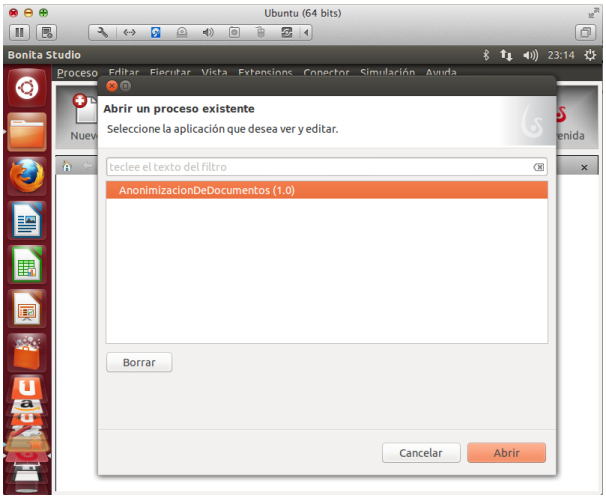

Figura 10: Bonita Studio

Una vez Bonita Studio finaliza la carga de los procesos, se presentarán los distintos procesos y subprocesos que componen el sistema Aplicación Demo. Se deberá seleccionar el proceso “Aplicación Demo” que se encuentra al final de todos los procesos, y seleccionar la opción “Ejecutar”. Ver Figura 11.

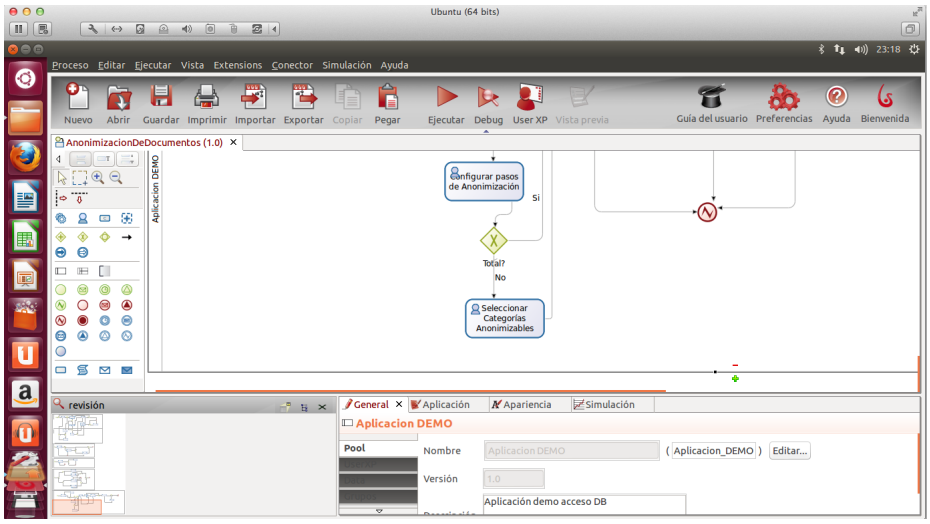

Figura 11: Proceso Aplicación Demo

El proceso de despliegue de los procesos puede tardar varios minutos. Una vez finaliza se iniciará automáticamente el navegador Firefox, presentando una nueva instancia del proceso “Aplicación DEMO” como se aprecia en la Figura. El sistema automáticamente inicia sesión con el usuario “admin”. Se pueden realizar pruebas con dicho usuario, o si se desea se puede utilizar el enlace “Salir”, e iniciar sesión posteriormente con el usuario “demo” con contraseña “demo” definido específicamente para tal fin.

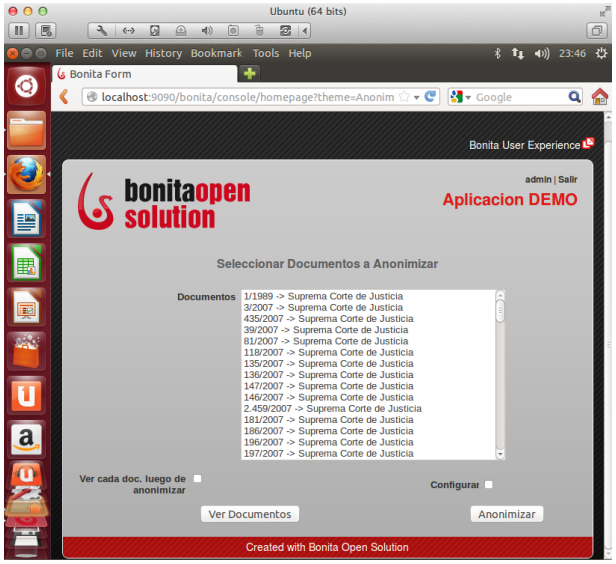

Figura 12: Sistema Aplicación Demo

3.5. Vista de Desarrollo

La presente vista profundiza en la organización de los componentes internos del sistema. Sobre la base de la estructura de paquetes "macro" definida en la Arquitectura de Referencia, se visualiza aquí su puesta en práctica en Aplicación DEMO, y los componentes adicionales que fueron definidos para implementar este sistema de anonimización específico.

3.5.1. Estructura de Paquetes

En la figura Estructura de Paquetes se describe la estructura de paquetes definida en la Aplicación DEMO.

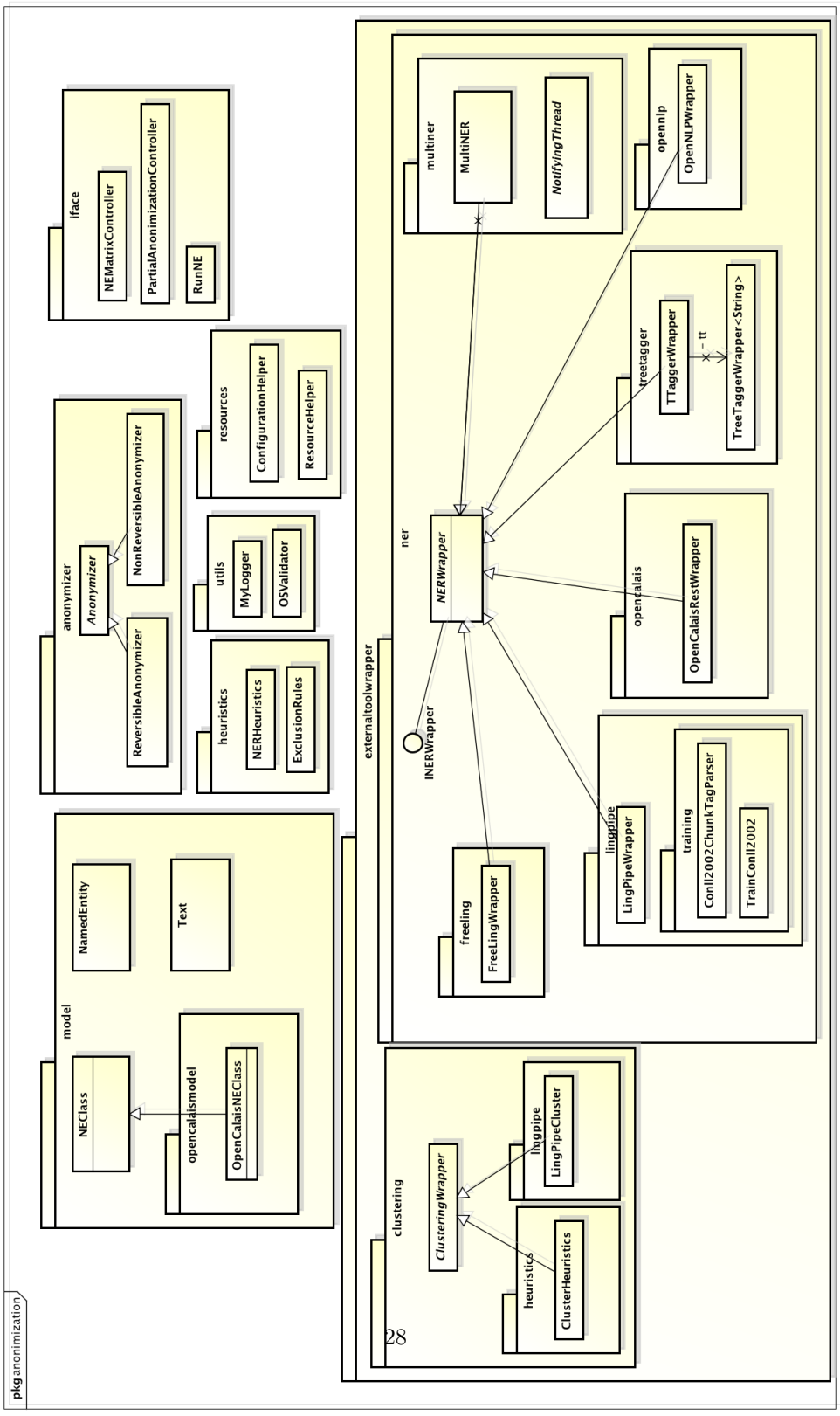

Figura 13: Estructura de Paquetes

3.5.2. Descripción del alcance funcional de los paquetes

- **model**: Define las estructuras de datos que representan el documento a ser procesado por el sistema así como las entidades con nombre y su clasificación.
  - Clase **model.NEClass**: Esta clase contiene un conjunto de constantes de tipo String, que representan tipos de entidades con nombre genéricos que pueden clasificar las herramientas NER. Esta clase debe ser extendida en caso de que una herramienta clasifique un conjunto más extenso de entidades con nombre. Se presenta un ejemplo de esta extensibilidad, en el paquete opencalaismodel, en la clase OpenCalaisNEClass, la cual añade algunas categorías específicas de la herramienta OpenCalais. Provee un método genérico getSupportedNEClasses, que utiliza introspección (reflection), y devuelve el conjunto de categorías de entidades con nombre en formato String[].
  - Clase **model.NamedEntity**: Modela las entidades con nombre. Guarda el término en el atributo "term", la clasificación de la entidad en neClass, y los términos, acrónimos que son equivalentes al término (ejemplo: R.O.U. = República Oriental del Uruguay).
  - Clase **model.Text**: Es el componente central del paquete. Text modela el documento que es procesado por el sistema de anonimización. Presenta un atributo de tipo String text, donde se almacena el documento propiamente dicho, y una estructura de tipo HashSet<NamedEntity>, donde se almacenan las entidades con nombre que son reconocidas en el documento por el sistema de anonimización.
- **externaltoolwrapper.clustering**: Contiene los adaptadores para las herramientas de clustering utilizadas para agrupar Entidades con Nombre.
  - Clase **externaltoolwrapper.clustering.ClusteringWrapper**: Implementación genérica de un wrapper (adaptador) para herramientas de clustering. Define un método abstracto buildClusters(Text), que deberá implementar cada wrapper específico. También provee la implementación de un método genérico "instantiate", el cual mediante introspección permite instanciar un wrapper mediante el nombre de su clase. De esta manera es posible obtener la definición de wrappers desde archivos de configuración XML, e instanciarlos dinámicamente en tiempo de ejecución.
  - Clase **externaltoolwrapper.clustering.lingpipe.LingPipeCluster**: Implementación de un wrapper específico para utilizar las capacidades de clustering del framework LingPipe.
  - Clase **externaltoolwrapper.clustering.heuristics.ClusterHeuristics**: Wrapper que genera los clusters de entidades con nombre utilizando reglas y patrones, por ejemplo reconociendo acrónimos. Este wrapper está autocontenido, es decir que no tiene otras dependencias de herramientas externas.

- **externaltoolwrapper.ner**: Encapsula los adaptadores para las diferentes herramientas NER.
  - Clase **externaltoolwrapper.NERWrapper**: Wrapper genérico para herramientas NER. Define un método abstracto tagNE(Text), el cual deberá procesar un documento y cargar las entidades con nombre identificadas dentro del mismo.
  - Clase **externaltoolwrapper.freeling.FreeLingWrapper**: Adaptador de la herramienta FreeLing. Ver sección 4.1
  - Clase **externaltoolwrapper.lingpipe.LingPipeWrapper**: Adaptador de la herramienta LingPipe. Ver sección 4.2
  - Clase **externaltoolwrapper.treetagger.TTaggerWrapper**: Adaptador de la herramienta TreeTagger. Ver sección 4.5
  - Clase **externaltoolwrapper.opencalais.OpenCalaisRestWrapper**: Adaptador de la herramienta OpenCalais. Ver sección 4.4
  - Clase **externaltoolwrapper.opennlp.OpenNLPWrapper**: Adaptador de la herramienta OpenNLP. Ver sección 4.1
  - Clase **externaltoolwrapper.multiner.MultiNER**: Adaptador MultiNER. Ver sección 5.
  - El paquete **externaltoolwrapper.lingpipe.training** contiene herramientas JAVA utilitarias para entrenar modelos de LingPipe a partir de un corpus específico.
- **anonymizer**: Módulo anonimizador.
  - Clase **anonimizer.Anonymizer**: Representa la abstracción genérica de un anonimizador. Define el método abstracto anonimize(Text), el cual deberá ser implementado por los anonimizadores específicos que especialicen esta clase. Provee una estructura HashSet<String> llamada partialList, donde se pueden especificar clases de entidades con nombre a ser anonimizadas, para contemplar la posibilidad de realizar anonimización parcial.
  - Clase **anonimizer.ReversibleAnonymizer**: Implementación de un anonimizador reversible. Utiliza cifrado AES para ocultar la información sensible.
  - Clase **anonimizer.NonReversibleAnonimizer**: Implementación de un anonimizador no reversible. Sustituye las entidades con nombre por etiquetas genéricas basadas en su clasificación (NEClass).
- **heuristics**: Gestiona los motores de reglas y patrones utilizados para mejorar el reconocimiento de entidades con nombre.
  - Clase **heuristics.NERHeuristics**: Realiza identificación de entidades con nombre por aplicación de reglas heurísticas. Las reglas se

cargan dinámicamente desde un archivo de texto, en el cual cada línea es una regla heurística. La primera columna en cada línea indica la clasificación de la entidad con nombre que sea reconocida por esta regla, la segunda columna es una expresión regular (REGEXP) en formato JAVA, y la última columna representa al subgrupo dentro de la expresión regular que debe considerarse la entidad con nombre. En caso de que la REGEXP represente a toda la entidad, dicha columna lleva el valor "0" (cero), en otro caso el valor del subgrupo que corresponda. Por ejemplo:

    - DATE (0[1-9]|[12][0-9]|3[01])([- /.])(0[1-9]|1[012])\2(19|20)\d\d 0
    - EMAIL ([\w\\-]([\.\w])+[\w]+@([\w\-]+\.)+[A-Za-z]{2,4}) 1
    - URL \b(https?|ftp|file)://[-a-zA-Z0-9+&@#/%?=~_|!:,.;]*[-a-zA-Z0-9+&@#/%=~_|] 0

- Clase **heuristics.ExclusionRule**: Gestiona reglas de exclusión, para no identificar como entidades con nombre términos específicos, que pueden inducir a errores a las distintas herramientas. Los términos a excluir se toman de un archivo de texto, donde cada línea es simplemente el término o frase a excluir.

- **iface**: Este paquete contiene clases que ofician de interfaz del sistema con el motor de procesos.

    - Clase **iface.RunNE**: Esta clase es la principal que oficia como interfaz para el motor de BPM Bonita. Los métodos de ésta clase son invocados por el motor de BPM para iniciar los distintos componentes del sistema de anonimización (NER, herramientas de clustering, heurísticas, anonimizadores).
    - Clase **iface.NEMatrixController**: Se utiliza para modelado específico de las interfaces de edición manual de clusters.
    - Clase **iface.PartialAnonimizationController**: Permite al proceso Bonita obtener la lista de clases de entidades con nombre soportadas por una determinada herramienta NER, con el fin de seleccionar categorías a anonimizar en el caso de una anonimización parcial.

- **utils**: Contiene servicios utilitarios utilizados por los diversos componentes.

    - Clase **utils.MyLogger**: Una especialización del Logger Apache Log4J. Se utiliza para registrar bitácoras en todo el sistema. Cabe destacar que desde el proceso Bonita no es posible visualizar la salida estándar JAVA al ejecutar los distintos componentes. Por tanto resulta particularmente útil esta clase para registrar el comportamiento de los distintos componentes, y realizar debugging.

    - Clase **utils.OSValidator**: Permite identificar el sistema operativo en tiempo de ejecución. Utilizada para soportar algunas diferencias entre plataformas que requieren ser contempladas en algunos componentes del sistema.

- **resources**: Contiene servicios utilizados para la gestión de recursos externos utilizados por las distintas herramientas (archivos binarios y de texto, modelos, etc).
- Clase **resources.ConfigurationHelper**: Este objeto permite gestionar la configuración XML del sistema. Se basa en Apache Commons Configuration.
- Clase **resources.ResourceHelper**: Esta clase realiza la carga desde sistema de archivos hacia objetos File de JAVA, de los artefactos binarios (recursos), tales como modelos binarios de las herramientas, librerías, etc.

4. Particularidades de los Adaptadores construidos

4.1. FreeLing

FreeLing[4] es una librería pensada para ser integrada e invocada desde otras aplicaciones, y tal es así que se distribuye con distintas APIs para integrarla en desarrollos sobre diversas plataformas, entre ellas JAVA. Es decir, no se trata una aplicación "standalone", si bien provee de una herramienta wrapper llamada analyzer que permite invocar a las distintas funciones de FreeLing desde la línea de comandos.

Al momento de utilizar FreeLing, se evaluaron entonces dos alternativas para interoperar con esta librería:

1. Una opción es invocar el wrapper de línea de comandos mencionado (analyzer), y encapsular su complejidad dentro de un wrapper propio dentro del sistema.

2. La segunda opción es utilizar la API para JAVA que provee esta librería. Esta API interactúa con FreeLing utilizando una biblioteca nativa la cual es invocada desde JAVA mediante JNI (Java Native Interface).

   Se consideró más "elegante" la segunda estrategia, es decir utilizar la API para integrar FreeLing al sistema de anonimización. Sin embargo, cabe decir que la integración de esta herramienta en particular revistió especial dificultad. Para poder utilizar la API fue necesario compilar la librería JNI nativa, para cada una de las plataformas en las que se probó el sistema. Para ejemplificar esto, para compilar esta librería en Mac OS X, fue necesario previamente compilar e instalar librerías de C que son dependencia de la misma: ICU4C, BOOST. Una vez se logra compilar la librería nativa, es necesario agregarla en CLASPATH de la aplicación, junto con la API JAVA de FreeLing que se distribuye en formato JAR.

4.2. LingPipe

LingPipe[5] es un framework desarrollado en JAVA puro, por tal motivo su integración al sistema fue muy sencilla, consistiendo en añadir el JAR correspondiente en el CLASSPATH, así como los recursos necesarios para operar con el módulo NER (un modelo binario pre-entrenado).

En el marco de la Aplicación DEMO, LingPipe tuvo una aplicación más amplia que el resto de las herramientas, ya que para esta librería se construyeron dos adaptadores. Uno para encapsular el manejo de su módulo NER, y otro para interoperar con sus herramientas de clustering.

4.3. OpenCalais

OpenCalais[6] es una herramienta disponible en línea, la cual provee interfaces de web services tanto para SOAP como para acceso REST.

Para desarrollar el wrapper se optó en primera instancia por desarrollar un cliente REST utilizando la API JAX-WS. Sin embargo posteriormente se encontró que existía un desarrollo de un tercero llamado J-Calais, el cual provee una API JAVA que encapsula la invocación a los web services. El adaptador para OpenCalais utiliza entonces J-Calais para interoperar con los web services.

4.4. OpenNLP

OpenNLP[7], al igual que LingPipe, es un framework desarrollado cien por ciento en JAVA. Por tal motivo su integración también consistió en simplemente agregar el JAR correspondiente junto con sus recursos en el CLASSPATH.

4.5. TreeTagger

La integración de TreeTagger[8] a una aplicación JAVA es similar a la de FreeLing, con la salvedad de que no se requiere interoperar mediante JNI ni compilar módulos y dependencias adicionales. TreeTagger provee una API Java, que no hace otra cosa que encapsular las invocaciones al ejecutable de TreeTagger que tengamos instalado en el sistema. La API es básicamente un adaptador tal como el que se pretendía construir. De todas maneras, se encapsularon las particularidades de dicho adaptador dentro de un wrapper análogo a los construidos para las demás herramientas, el cual sigue la firma y estructura definida en la arquitectura de referencia.

5. El adaptador MultiNER

El adaptador MultiNER merece un capítulo aparte, ya que no se trata de un adaptador para una herramienta en particular, sino que es un componente "original" del prototipo desarrollado, el cual permite integrar y utilizar en paralelo múltiples herramientas NER.

A la vista de la variable efectividad de las herramientas, surgió la idea de implementar un wrapper adicional y genérico que permitiera utilizar al mismo tiempo todas (o un subconjunto de) las herramientas NER sobre cada documento.

Este wrapper fue denominado "MultiNER", y funciona de la siguiente manera:

- Se procesa en paralelo (utilizando "threads") el documento con cada una de las herramientas disponibles o seleccionadas. Para ello se utiliza una especialización de la clase Thread de JAVA implementada en la clase "NotifyingThread".
- Las herramientas comparten una estructura de tipo "HashTable" llamada rankTable. En dicha tabla se guarda como clave la entidad con nombre identificada, y como valor la cantidad de herramientas que identificaron dicha entidad con nombre.
- Una vez finaliza la ejecución de todas las herramientas, se realiza un análisis de la tabla rankTable resultante, y se toman como válidas solamente aquellas entidades con nombre que hayan sido identificadas por un cierto número de herramientas (el umbral mínimo de herramientas para que una entidad sea considerada es configurable, aportando una vez más al atributo Configuración identificado en la arquitectura).

  La secuencia de llamadas, asincrónicas y sincrónicas se ilustra en la Figura 14.

En lo que refiere a la clasificación de las entidades en el módulo MultiNER, dado que como se explicó las herramientas presentan diversas capacidades de clasificación, cuando se determinan distintos tipos de clase por parte de distintas herramientas prevalece la primera simplemente como convención. Pero cuando una herramienta no clasifica a una entidad o la clasifica con alguna categoría poco específica, si otra herramienta realiza una mejor clasificación se toma la de ésta.

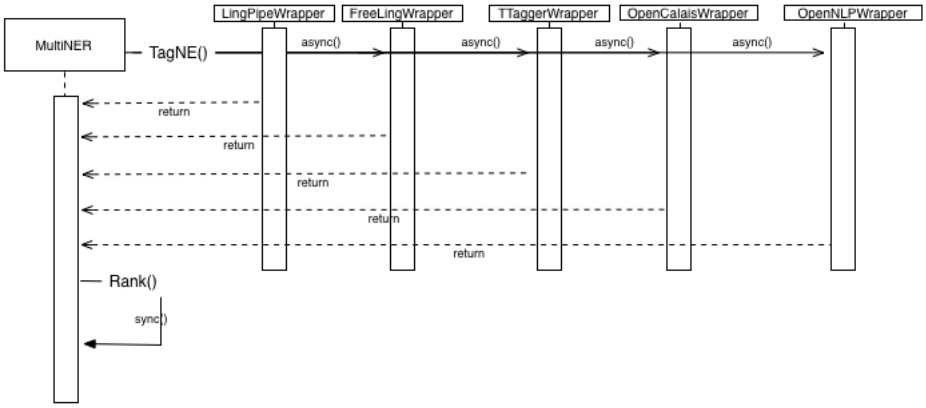

Figura 14: Diagrama de Secuencia - MultiNER

6. Uso de Introspección (Reflection)

En el sistema de anonimización "Aplicación DEMO" se hace un uso extensivo de introspección de tipos, una herramienta poderosa que proveen lenguajes orientados a objetos maduros como JAVA. En particular en MultiNER se utiliza introspección para gestionar todo el manejo de los múltiples adaptadores que se pueden llegar a invocar desde éste módulo. Pero en general todos los adaptadores NER se manejan de forma "abstracta" mediante la superclase NERWrapper, y son instanciados en su tipo específico en tiempo de ejecución. Esto permite que los adaptadores puedan ser incorporados dinámicamente, sin necesidad de recompilar código alguno. El resultado es que se simplifica enormemente la configuración y posterior invocación desde el motor de procesos de los distintos adaptadores, con cero acoplamiento entre el motor y las herramientas externas que se terminan invocando para cada tarea particular. Análogamente los adaptadores para las herramientas de clustering, también son instanciados mediante introspección desde la superclase ClusteringWrapper.

Referencias

[1] OMG. (2011) Business process model and notation (bpmn) versión 2.0. OMG. [Online]. Available: http://www.omg.org/spec/BPMN/2.0/PDF

[2] BonitaSoft. (2001) Bonita open solution. [Online]. Available: http://es.bonitasoft.com/

[3] O. Corporation. (1995) Mysql. [Online]. Available: http://dev.mysql.com

[4] P. L. (2003) Freeling. TALP Research Center - Universitat Politècnica de Catalunya. [Online]. Available: http://nlp.lsi.upc.edu/freeling/

[5] Alias-I. (2003) Lingpipe. [Online]. Available: http://alias-i.com/lingpipe/

[6] T. Reuters. (2008) Opencalais. Thomson Reuters. [Online]. Available: http://www.opencalais.com

[7] A. S. Foundation. (2000) Apache opennlp. Apache Software Foundation. [Online]. Available: http://opennlp.apache.org

[8] S. H. (1994) Treetagger. Institute for Computational Linguistics of the University of Stuttgart. [Online]. Available: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

[9] D. T. Informática. (2011, 12) Base de jurisprudencia nacional pública. Poder Judicial - República Oriental del Uruguay. [Online]. Available: http://bjn.poderjudicial.gub.uy/