Informe de Proyecto de Grado

Sistema para la gestión de la Clínica Notarial
Facultad de Derecho

Instituto de Computación,
Facultad de Ingeniería,
Universidad de la República,
Montevideo, Uruguay
2018

Integrantes:
Christian Favio Nuñez Machado
Horacio Gastón Borba Marins
Michell Andrés Vanrell Escalante

Tutor:
Ariel Sabiguero Yawelak

Tribunal:
Daniel Meerhoff
Federico Rodríguez
Omar Viera

Resumen

La Clínica Notarial de la Facultad de Derecho, brinda un servicio de asesoria Notarial a personas de bajos recursos, y también forma parte de la carrera de grado de Notariado en ella los estudiantes realizan prácticas con los casos que llegan a la clínica, actualmente se realiza toda la gestión de los casos a papel algo inaceptable para los tiempos que corren.
En el presente trabajo nuestro equipo introduce el estado del arte de las herramientas de desarrollo de software, la elección de dicha herramienta e informatizar la gestión de la Clínica. Para ello construimos el sistema de gestión de la Clínica Notarial y su implantación y puesta a punto para ser utilizado a comienzo del próximo año, esto será un gran beneficio para los funcionarios administrativos, los estudiantes, los docentes y los usuario finales.

Palabras Clave —OpenXava, gestión, notarial.

Agradecimientos

La realización de este proyecto no hubiese sido posible sin la colaboración de muchas personas, a quienes queremos agradecer en la presentación de este documento.
En primer lugar queremos agradecer especialmente a nuestros tutor Ariel Sabiguero por su apoyo incondicional durante el transcurso del proyecto, su experiencia y conocimiento fueron fundamentales para la realización del mismo.

En segundo lugar agradecer al Director Escribano Carlos Scirgalea y a la Profesora Escribana Estela Pena, por sus aportes de alto valor y su excelente disposición en el transcurso de este proyecto.

En tercer lugar a los Funcionarios de la Clínica Notarial de la Facultad de Derecho, en particular a Nicolás Guerra, Estela Moreira y Richard Carbajal, por la amabilidad y predisposición al permitirnos utilizar sus instalaciones para la realización de pruebas de campo.

Finalmente agradecemos a nuestras familias y amigos por la contención y el apoyo brindado durante todo el periodo del proyecto.
A todos ellos, muchas gracias.

Índice General

Introducción

La Clínica Notarial de la Facultad de Derecho de la Universidad de la República brinda un servicio de asesoría legal Notarial, este servicio es totalmente gratuito teniendo como usuarios del mismo a personas de bajos recursos y con distintas problemáticas sociales.
Este servicio actualmente se brinda todo a papel con escasa o nula intervención de la tecnología y los sistemas de Información.
A esta altura en los tiempos que corren es indispensable informatizar la gestión del servicio, nuestra tarea es construir desde cero e implantar dicho sistema informático haciendo que se relacione con el sistema del Consultorio Jurídico, también con otros sistemas de la Facultad de Derecho como el de Gestión de Funcionarios y Estudiantes.
Con este sistema informático se benefician directamente los funcionarios de la Clínica, el Equipo Docente y los Estudiantes pues se les simplifica las tareas y el trabajo en general haciéndolos mucho más eficientes y así pueden lograr brindar un servicio de mejor calidad.

Por otro lado y no menos importante se beneficiaran indirectamente todos los usuarios de la Clínica Notarial y la sociedad toda por tratarse de un servicio público y gratuito.

Objetivos

- Hacer un estudio de las tecnologías existentes y el estado de arte de las distintas herramientas tanto de desarrollo software como de las plataformas sobre las cuales es posible la implementación del sistema requerido. Teniendo en cuenta que dichas herramientas deben ser de carácter gratuito. En base a este relevamiento se elegirá el o las herramientas a utilizar que más se adecuen a las necesidades del cliente.
- Diseñar, construir e implantar una solución de software que permita gestionar la Clínica Notarial de manera informatizada, ágil, segura y eficiente.
- Elaboración de un Manual de Usuario, brindar un documento ordenado y sencillo para educar a los usuarios finales en el uso del Sistema Clínica Notarial.
- Elaboración de un Manual para administradores, brindar un documento técnico detallado sobre como instalar y configurar el sistema en un servidor. Así como para dar soporte a futuros desarrolladores del sistema.

Cronograma

Para organizar el trabajo y las tareas y así alcanzar los objetivos planteados, definimos la siguiente metodología de trabajo e identificamos las siguientes fases:

- Análisis
- Diseño
- Desarrollo
- Testing
- Implantación
- Documentación

	Abr	May	Jun	Jul	Ago	Set	Oct	Nov	Dic	Ene
Análisis										
Diseño										
Desarrollo										
Testing										
Implantación										
Documentación										

Prototipo

	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Set	Octr
Análisis										
Diseño										
Desarrollo										
Testing										
Implantación										
Documentación										

Versión 1

	Set	Oct	Nov	Dic
Análisis				
Diseño				
Desarrollo				
Testing				
Implantación				
Documentación				

Versión 2

A continuación se detalla la información de las distintas fases.

Análisis

En esta etapa se obtuvieron los distintos requerimientos para poder realizar el proyecto.
En un principio tuvimos reuniones con el cliente y el usuario final, también el cliente nos otorgó copias de los formularios en papel que utilizan para llevar a cabo sus tareas. A partir de dichas reuniones nos fuimos familiarizando con las necesidades del cliente y con todos esos insumos se establecieron los objetivos y generamos la especificación de requerimientos. Luego de esto realizamos un estudio de las herramientas y las posibles plataformas a utilizar para poder llevar a cabo el proyecto.
Con toda esta información recabada y teniendo en cuenta el estudio realizado se define el alcance del proyecto.

Diseño y Construcción

A partir de los conocimientos generados en la etapa de Análisis elegimos las herramientas a utilizar para poder implementar la solución. En un principio elaboramos algunos mockups para asegurarnos de que habíamos entendido lo que se nos estaba solicitando y a partir de ellos generamos algunas pantallas y casos de usos.
Diseñamos la arquitectura del sistema, los mecanismos de comunicación y las interfaces con los distintos sistemas que están involucrados.
Por otra parte elaboramos algunos pequeños prototipos con distintas herramientas de desarrollo para evaluar la viabilidad y alcance del proyecto.

Organización del Documento

Aquí se detalla cómo está organizado el documento y se describen las actividades que fuimos realizando.

Estado del Arte: Mostramos el relevamiento de las herramientas realizado en la etapa de Análisis.

Requerimientos y alcance: Muestra los requerimientos del Cliente funcionales y no funcionales, se definen cuáles de estos requerimientos se enmarcan dentro del alcance y serán implementados en este proyecto.

Solución Propuesta: Se muestra el diseño del Sistema, sus interfaces y su arquitectura.

Implementación: Mostramos la metodología de desarrollo utilizada para la construcción de las distintas versiones de Prototipo y como se implementó la versión final.

Test: Se detallan las pruebas realizadas al Sistema.

Manuales: Se detallan los distintos manuales tanto para usuarios, para administradores y para desarrolladores.

Uso de OpenXava: Se muestra como se usó OpenXava desde el punto de vista del desarrollador.

Conclusiones y trabajo futuro: Se evalúan los resultados obtenidos, se desarrollan lo nuevos conocimientos y las lecciones aprendidas a lo largo del proyecto y por último se plantean los pasos a seguir para el complemento futuro del sistema.

Estado del Arte de las Herramientas

A continuación mostramos la evaluación de herramientas de desarrollo de software, si bien existen una gran variedad de estas, de las reuniones con el cliente obtuvimos algunos requerimientos funcionales, por otro lado también tenemos requerimientos funcionales desde nuestro lado como desarrolladores. Elegiremos la herramienta que a nuestro entender se adapte de mejor manera a estos requerimientos funcionales.
Requerimientos funcionales Cliente :

- La herramienta de desarrollo debe ser gratuita y con licencia LGPL de ser posible.
- La aplicación final desarrollada debe poder ejecutarse en ambientes con software libre instalado, tanto para servidores y clientes. Por ejemplo sistemas Linux o Android.
- Integración con LDAP.
- Interfaces con file system o agilidad para agregar documentos.

Requerimientos funcionales Desarrollo:

- Herramientas de desarrollo ágil.
- Curva de aprendizaje media o alta.
- Buena cantidad y calidad de documentación.

Motor de Base de Datos

El motor de base de datos es un servicio principal y primordial para almacenar, procesar y proteger los datos es por esto que un motor de base de datos debe tener las siguientes características: rendimiento, confiabilidad y seguridad. No menos importante es su método de licenciamiento este debe tener licencia pública, estudiaremos algunas de las herramientas que cumplen estas características.

PostgreSQL

PostgreSQL(1) es un motor de base de datos relacional de código abierto que utiliza SQL, se ejecuta en todos los sistemas operativos principales como Windows o Linux es compatible con ACID.
Alguna de su principales características son Tipo de datos, Integridad de datos, Concurrencia, Rendimiento, Confiabilidad, Seguridad.

MySQL

MySQL(2) es un motor de base de datos, que utiliza lenguaje SQL como NoSQL, es un sistema de con licencia dual Licencia pública/Licencia comercial y es considerada la base de datos de código abierto más utilizada en el mundo. MySQL funciona en múltiples sistemas operativos como Windows y GNU/Linux. Sus principales características son Amplio conjunto de lenguaje SQL, Disponibilidad en muchas plataformas, Transacciones, Conectividad segura.

En definitiva ambos motores de base de datos cumplen con los requerimientos, nosotros escogimos MySQL más que nada por un conocimiento previo del uso de la aplicación y sus extensiones.
De todas maneras si por algún motivo es necesario utilizar otro motor de base de datos, perfectamente se puede usar PostgreSQL simplemente dentro del sistema se debe indicar que conector de base de datos se va utilizar y configurarlo con unos pocos parámetros

Herramientas para el Desarrollo

En la actualidad las herramientas para desarrollo y los lenguajes de programación que utilizan, las plataformas a la que están destinados son muchísimos y muy variables, la tarea de dominarlos todos es inabarcable para cualquier persona. Entonces la decisión de cuál herramienta utilizar se vuelve muy importante y no solo se debe tener en cuenta los requerimientos funcionales y no funcionales del sistema, sino también las particularidades del proyecto, los conocimientos de los distintos miembros del grupo y la curva de aprendizaje de cada herramienta. A continuación hacemos un análisis de distintas herramientas de desarrollo, como son Spring Boot, OpenXava, Core .Net, Genexus.

Spring Boot

Spring Boot es una herramienta que nos permite cumplir con todas las necesidades que a priori determinamos para la construcción del software para la clínica notarial. En un primer lugar se trata de una herramienta que es de software libre y nos permite trabajar con Java, tiene una excelente integración con otras tecnologías, en su uso se reducen los tiempos de configuración y despliegue del producto, haciendo que el centro de la actividades sean netamente sobre el desarrollo de la aplicación. Una de sus principales desventajas es que tiene una curva de aprendizaje alta.

Paradigma

Utiliza un Paradigma de microservicios[(2)], estos son autocontenidos es decir contienen todo lo necesario para ofrecer su funcionalidad, son fáciles de mantener gracias a lo abstractos que son, fáciles de testear y de desplegar, sencillo de configurar ya que no hay que preocuparse mucho por las dependencias y provee un asistente ágil e intuitivo y fácil de desplegar ya que posee sus propios servidores web.

Integración con otras tecnologías

Spring boot soporta los siguientes servidores web: Tomcat, Jetty o Undertow y pasar de uno a otro es muy transparente para la aplicación, además proporciona algunas características adicionales como inicializar el contenedor IoC de Spring, configuración, perfiles para diferentes entornos (desarrollo, pruebas, producción), monitorización y métricas del servidor de aplicaciones y soporte para la herramienta de automatización Gradle entre algunas más.

Desarrollo de aplicaciones Web

Nos ofrece versatilidad a la hora de elegir qué tipo de aplicación queremos construir siendo muy adecuado para el desarrollo de aplicaciones web, ya que puede crear fácilmente un servidor HTTP utilizando como servidor Tomcat, Jetty o Undertow integrado.

También se puede utilizar el framework MVC de spring Web (Spring MVC) con todas sus características, o si se prefiere también se puede usar JAX-RS y Jersey para un modelo de programación -JAX-RS para los endpoint REST.

Seguridad

Spring Security[7] provee un marco de autenticación y control de acceso muy potente y altamente configurable y personalizable:

- Integración API Servlet.
- Soporte integral tanto para autorización como autenticación.
- Protección contra ataques como clickjacking, fijación de sesión, falsificación de solicitud de cross site.
- Integración opcional con Spring Web MVC.

Acceso a Datos

Spring boot proporciona un vasto soporte para trabajar con base de datos SQL. Desde el JDBC directo o las tecnologías de mapeo relacional de objetos como hibernate.

- Datasource: el método standard para trabajar con conecciones de base de datos.
- JDBCTemplate: Clases que se auto configuran automáticamente.
- JPA: es una tecnología que permite asignar objetos a bases de datos relacionales, como son Hibernate, Spring Data JPA, Springs ORMs.
- Integración con MyBatis.

Integración con LDAP

Facilita la integración de las aplicaciones desarrolladas con Spring que usan LDAP[3], Spring LDAP es una biblioteca de JAVA para simplificar las distintas operaciones LDAP, está basada en el patrón JDBCTemplate de Spring. Libera al usuario de tareas comunes como buscar y cerrar contextos, valores de codificación, decodificaciones, filtros y mas.

Características:

- Proporciona una plantilla LDAP que elimina la necesidad de preocuparse por la creación y el cierre de LdapContext y el bucle a través de NamingEnumeration.
- Jerarquía de excepción completa no comprobada basada en DataAccessException de Spring.
- Contiene clases para la creación dinámica de filtros LDAP y Distinguished Names.
- Gestión de transacciones LDAP del lado del cliente.

Testing

Spring Boot Testing[4] provee un gran número de utilitarios y “annotations” para ayudar en el testeo de la aplicación. Se proveen 2 módulos spring-boot-test y spring-boot-test-autoconfigure, el primero contiene elementos básicos y el Segundo admite auto-configuración para pruebas.

- La mayoría de los desarrolladores utilizan spring-boot-starter-test que importan ambos módulos así como también Junit, Spring Test & Spring Boot Test, AssertJ, Hamcrest, Mockito, JSONassert, JSONPath.
- WebSockets.

MyBatis-Spring-Boot-Starter

Ayuda a crear rápidamente aplicación mybatis sobre Spring Boot[5] y permite:

- Crear aplicaciones independientes.
- Menos configuración XML.
- Provee configuración Rápida.
- Escaneo avanzado.
- Uso de SQLSession.
- Detección de componentes MyBatis.
- Compatibilidad con Hibernate.

Trabaja con tecnologías NoSQL

Spring data[6] provee proyectos adicionales para trabajar con tecnologías NoSQL como MongoDB, Neo4J, Elasticsearch, Solr, Redis, Gemfire,Cassandra, Couchbase y LDAP.

Spring Framework

Ventajas

- IDE propio Spring Tools Suite basado en eclipse que facilita el desarrollo.
- Open Source.
- Tecnología Madura.
- Buena Documentación en el sitio y en la web.
- Mucha cantidad de usuarios en Stack Overflow y en la web.
- Gran comunidad.
- Cuenta con proyectos-módulos para las necesidades de infraestructura de la aplicación (Spring Boot, Spring MVC, Spring Security, Spring Data y Spring LDAP)
- No pone restricciones en el diseño de la interfaz de usuario.

Desventajas

- Instalación y configuración compleja.
- Curva de aprendizaje alta.
- Mucho código repetitivo para implementar ABMs con listados, filtros y reportes.
- Mucho código client-side para mostrar y manejar las relaciones entre las entidades.
- Demoramos una semana aproximadamente en crear un prototipo.

Hibernate

Ventajas

- Open Source
- Se integra bien con Spring Boot
- Generación automática de las tablas a partir de los objetos POJO
- Buena integración con Spring Data

Desventajas

- Issues de memoria cuando se utiliza gran cantidad de datos.
- El mapeo a los esquemas puede ser tedioso.
- Consultas complejas a la base de datos se tienen que hacer con SQL

MyBatis

Ventajas

- Open Source.
- MyBatis elimina casi todo el código JDBC.
- Creación de los objetos POJO a partir de las tablas de la base de datos
- El mapeo se puede hacer a través de XML o anotaciones.
- Integración con Spring Boot.
- Generador de código MyBatis Generator, genera los objetos POJO, los mapeos y un conjunto de operaciones y consultas a partir de las tablas en la base de datos.
- Mejor performance que Hibernate

Desventajas

- Difícil configuración e instalación
- Curva de aprendizaje alta
- No es compatible con Spring Data.
- No tiene validadores, se puede usar Hibernate Validator o Bean Validations
- Si se edita el código generado por MyBatis Generator y luego se cambia una tabla en la base de datos, hay que adaptar el código a el nuevo cambio.
- Si no se edita el código generado por MyBatis y se quieren agregar funcionalidades a los objetos POJO (como validadores y mapeos JSON) se puede hacer externamente con configuración XML o usando mapeadores, no hay una forma estándar de hacer esto.
- Demoramos una semana aproximadamente en adaptar el prototipo anterior para que use MyBatis.

OpenXava

Este es el paradigma de OpenXava[8], un marco de trabajo AJAX para desarrollo rápido de aplicaciones web empresariales, sólo hay que escribir las clases del dominio con Java para obtener una aplicación web lista para producción.

A continuación se detallan la principales características de OpenXava:

- Alta productividad: Solo se escribe la lógica de negocio y las estructuras de datos, no se escribe código HTML, SQL, CSS, JavaScript. La interfaz de usuario y la lógica de la base de datos las provee OpenXava automáticamente.
- Curva de aprendizaje corte: si uno sabe escribir clases de java simples, prácticamente esto es suficiente para desarrollar aplicaciones completas. Por otro lado la distribución de OpenXava viene preparada para un inicio agil y rapido.
- Alta Funcionalidad, OpenXava genera automáticamente:
    - La interfaz de usuario (AJAX) sin recarga de página.
    - Modo de lista con paginación, ordenado, filtrado.
    - Exportación a PDF y planillas de cálculo.
    - Modo detalle con pestañas, marcos y diálogos adaptables.
- Multiplataforma:
    - Funciona en los principales navegadores web como (Internet Explorer, Firefox, Chrome y Safari)
    - Es soportado por todos los motores de base de datos que usan hibernate como Oracle,MS SQL Server, PostgreSQL, MySQL entre otros.
    - Sistemas Operativos con soporte para Java 6 o superior como Linux,Windows y Mac.
    - Servidores de aplicaciones con soporte de Servlets 3.0 o superior como por ejemplo Tomcat, WebSphere, JBoss, etc.
    - Portales con soporte JSR-168 ó JSR-286 incluyendo WebSphere Portal y Liferay
- Código Abierto con licencia LGPL nos permite desarrollar aplicaciones sin tener que pagar.
- Multilingüe, etiquetas y mensajes en varios idiomas incluido Español.
- Gran documentación y una comunidad activa.
    - Miles de descargas
    - Cientos de aplicaciones ya desarrolladas
    - Guías de referencia en varios idioma incluido Español
    - Soporte en los foros de la comunidad.
    - Proyecto activo, con varias versiones al año.
- Basado en estándares Java.
    - Podemos migrar código Java a OpenXava y viceversa
    - OpenXava soporta JSR-317, JSR-303, JSR-330, JSR-220, JSR-153, JSR-168 y JSR-286.

- Herramientas de terceros
    - MinuteProject: Genera App OpenXava a partir de una base de Datos.
    - Mogwai ERDesigner: a partir del modelo entidad relación genera aplicaciones OpenXava.
    - Moskitt: genera aplicaciones OpenXava a partir de modelos UML.
- Interfaz de usuario móvil: En la última versión se dispone de una interfaz de usuario web para dispositivos móviles a partir del mismo código.

Este framework para JAVA provee como mayor ventaja la generación automática de los ABM únicamente desde la escritura de la clase, esta funcionalidad es muy valiosa en referencia al tipo de proyecto que debemos afrontar donde la mayor parte del trabajo de desarrollo conlleva la realización de los ABM's.

Las funcionalidades relacionadas con la seguridad (LDAP) y generación para la plataforma mobile están disponibles en la versión paga. Como alternativa al tema de la seguridad y control de acceso se puede utilizar Liferay en conjunto con OpenXava para ejecutar la aplicación y trasladar parte de los temas de control de acceso a la configuración del sitio en Liferay.

Ventajas

- Facil y Rapido de configurar en 3 pasos ya está pronto para empezar a implementar
- Curva de Aprendizaje rápida/media
- Rápido y sencillo para crear ABMs que estimamos sea un 75% de nuestra aplicación web.
- No es necesario escribir HTML, CSS, SQL, JavaScript
- Aplicaciones con mucha funcionalidad
- El mapeo se puede hacer a través de anotaciones, utiliza Hibernate
- Se puede obtener un prototipo funcional en cuestión de horas
- Integración con Liferay como alternativa al tema de la seguridad y control de acceso

Desventajas

- No es completamente open source hay funcionalidades que vienen en la versión premium como LDAP y generación mobile.
- Un poco más de dificultad para crear objetos que no son ABMs.
- El consumo de recursos puede estar por arriba de la media.

Liferay

Es un portal de gestión de contenidos de código abierto escrito en Java, es una herramienta de desarrollo orientada a crear portales web, intranets y sitios web. Liferay[(9)] permite crear de manera online un portal web eficiente y productivo, comunidades enteras, blogs, foros, etc son sencillos de crear con solo arrastrar los portlets. Se puede elegir entre qué tipos de solución de software nos interese más como: Intranets, Mobile, Portales y Web.

Sus principales características son:

- Multiplataforma:
    - Corre en los principales servidores de aplicaciones como Tomcat y JBoss.
    - Funciona con motores bases de datos como MySQL y PostgreSQL.
    - Funciona con todos los navegadores más importantes.
    - Sistema operativos como por ejemplo Windows y Linux.
- Usabilidad inmediatas: existen más de 60 portlets prediseñados listos para ser utilizados, con las funcionales más comunes y más necesarias dentro de un portal web.
- Seguridad: Se considera uno de los portales más eficientes y seguros
- Usabilidad: Ofrece personalización de páginas para todos los gustos y todas las necesidades
- Liferay como CMS, como plataforma de gestión de contenidos nos ofrece, entre otras características que no están relacionadas con el proyecto.
    - Librería de documentos y galeria de imagenes centralizado.
    - Lista de contenidos web
    - Atributos personalizados

Ventajas

- Fácil e intuitivo: es muy sencillo de utilizar, de configurar y de crear contenido.
- Soporte multilingüe.
- Ofrece interfaces con protocolos que son necesarios como LDAP
- Licenciamiento gratuito.
- Soporte de usuarios y roles, con interfaz para vincularnos a nuestra aplicación.
- Manejo de documentos e imágenes.
- Se pueden incorporar otras aplicaciones al portal

Desventajas

La principal desventaja es que estamos agregando una capa más a nuestra aplicación, si bien es necesaria para el funcionamiento de usuarios, roles y LDAP, también puede acarrear otra clase de dificultades y problemas que a priori no están previstas.

Core.Net

.Net Core[13] es una herramienta de desarrollo multiplataforma para uso general, admite Windows, macOS y Linux también admite otros escenarios como Nube, dispositivos e IoT. Del mantenimiento se encargan Microsoft y la comunidad .Net en GitHub.

Las características de .Net Core son las siguientes:

- Multiplataforma: se ejecuta en Windows, macOS y Linux y se puede portar a otros sistemas operativos.
- Código Abierto: la plataforma .Net Core es de código abierto, con licencias de MIT y

Apache 2.
- Otras características: Ejecución desde línea de comandos, Implementación Flexible y Compatible con Microsoft.

Ventajas

- Una gran comunidad y el soporte e impulso de Microsoft
- Multiplataforma se puede generar para Windows como para Linux
- Aplicaciones con mucha funcionalidad y de uso general
- Compatibilidad con LDAP

Desventajas

- Curva de Aprendizaje alta/media.
- Configuración e Instalación previa de distintas Herramientas
- Si bien es multiplataforma, hace relativamente poco tiempo que se abrió a Linux esto puede generar inconvenientes no previstos.
- Hay que desarrollar desde 0 la seguridad

Genexus

Su enfoque de desarrollo es basado en el conocimiento de los negocios, es una herramienta ágil, productiva y multiplataforma para cualquier proyecto de software. Con Genexus se pueden desarrollar aplicaciones de escritorio, aplicaciones web y aplicaciones móviles. Genexus tiene un paradigma más declarativo se escribe en un lenguaje de alto nivel a la hora de programar y se desarrolla de la misma forma sin dependencia de lenguaje, base de datos o plataforma ya que es un generador de códigos y puede generar para distintos lenguajes (Java, .Net, Ruby, etc), también para distintas base de datos (Oracle, MySQL, MS SQL Server, etc) y distintos sistemas (Windows, IoS, Android, etc). Es una herramienta cuyo uso está supeditado a la adquisición de licencias, por un lado la correspondiente al entorno de desarrollo (IDE) y por otro a la del generador del lenguaje que se desee utilizar para implementar la solución de software. Existen versiones de tipo trial que, si bien son completas en lo que refiere las prestaciones que proveen, las mismas incorporan un conjunto de limitaciones en lo que tiene que ver con el tamaño de las KBs a implementar (cantidad de objetos y atributos) que no permiten su uso para desarrollos de porte medio o grande. Estas versiones están enfocadas en el aprendizaje de la herramienta y generar así una comunidad de intercambio de conocimiento.
Las KBs de GeneXus están enfocadas en consolidar el conocimiento y las reglas del negocio al cual aplica la solución a desarrollar más que en aspectos más técnicos de cómo resolver la conectividad con motor de base de datos u otro. Como ventaja importante, una misma KB permite generar soluciones para diferentes plataformas, en diferentes lenguajes y contra diversos motores de bases de datos.

El IDE de GeneXus incorpora de forma nativa un manejador de versiones a fin de poder tener diferentes versiones, valga la redundancia, de un mismo desarrollo lo cual sirve para tener congelada una versión que está en producción en un cliente y por otro lado tener un branch diferente en el cual se están incorporando las nuevas funcionalidades de la futura versión. Como complemento a GeneXus y para casos de equipos de desarrollo colaborativos existe GXServer que da la solución de Versionado de forma concurrente, obviamente a costo de otro licenciamiento y de la adquisición de conocimiento para su gestión.

Ventajas

- Productividad en el desarrollo ya que se pueden obtener soluciones funcionales de forma muy rápida de manera de enfocar la mayor parte del tiempo en las reglas del negocio más que los aspectos de generación de la base o la seguridad de los objetos.
- Una gran comunidad en Uruguay
- Multiplataforma se puede generar para servidores Windows como para Linux
- Compatibilidad con LDAP
- Los miembros del grupo ya tenemos experiencia en el desarrollo de aplicaciones Genexus
- Baja configuración
- Seguridad con GAM, lo cual resuelve los procesos de autorización y autenticación. También asegura que el código generado cumple con el OWASP Top Ten en lo referente a seguridad.
- Permite el desarrollo de User Controls para extender la capacidad de desarrollo.

Desventajas

- Licenciamiento pago
- Si bien es multiplataforma, se debe desarrollar desde equipos windows.
- No estaríamos incorporando nuevos conocimientos sobre la herramienta de desarrollo salvo posibles obstáculos puntuales que pudiesen surgir.

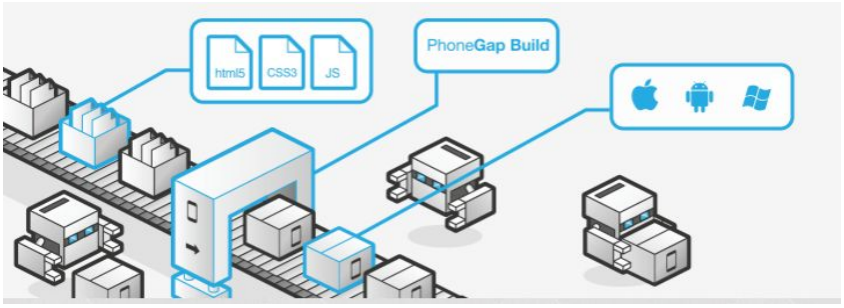
PhoneGap

Es un framework que permite el desarrollo de aplicaciones híbridas para una importante gama de sistemas operativos móviles, como son entre otros: Android, IOS, Windows Phone, BlackBerry, etc.

El desarrollo está basado en HTML5 + CSS + JS y se interconecta con el dispositivo por intermedio de una API que le permite utilizar desde la aplicacion los recursos de este como son la cámara, agenda, teléfono, GPS, etc. En la siguiente imagen se ilustra el funcionamiento de Phonegap.

De esta manera se genera una aplicación híbrida que ejecuta dentro de un webview (navegador a pantalla completa) con la ventaja de poder utilizar el hardware del dispositivo de igual forma que las aplicaciones nativas.

Actualmente PhoneGap permite 2 formas de trabajo en lo que respecta a la compilación y generación de los empaquetados para los distintos S.O. Por un lado se puede trabajar localmente en un windows o mac y compilar localmente (android con el SDK en windows y IOS en una MAC usando XCode). La otra forma, desde que Adobe adquirió PhoneGap, es a través de la nube (Adobe PhoneGap Build) con un costo de por medio. La ventaja de utilizar la nube radica en que Adobe recibe el código fuente y nos devuelve los empaquetados para las distintas plataformas elegidas, .apk en el caso de Android, .ipa en el de IOS, etc. En la imagen siguiente se ilustra el modo de empaquetamiento.



Ventajas

- Open source
- Se basa en lenguaje HTML5 + JS + CSS3
- Genera una aplicación híbrida que puede interactuar con el hardware del equipo por intermedio de una API
- El mismo código puede compilarse para varias plataformas (Android, IOS, BlackBerry, Windows Phone, etc.)

Desventajas

- La curva de aprendizaje en lo que refiere con la interacción con la API puede ser alta.
- Actualizar un proyecto a una nueva versión de PhoneGap puede ser complejo y puede haber incompatibilidades.
-

Jquery Mobile

Este framework es muy utilizado en el desarrollo de soluciones mobile, provee una biblioteca muy amplia de funcionalidades listas para utilizar (botones, tipografía, formularios, etc.) lo cual permite generar aplicaciones de forma rápida utilizando HTML, CSS y JS.

Ventajas

- Se puede desarrollar para todas las plataformas.
- No se restringe a un lenguaje nativo, se usa HTML, CSS y JS
- Soporte para AJAX

Desventajas

- La personalización de la aplicación puede resultar difícil
- El uso de CSS puede resultar complejo
- Es un lenguaje estandarizado por lo que no se adapta a las diferentes plataformas lo cual puede generar una experiencia de usuario pobre.
- No utiliza plantillas prediseñadas al momento de desarrollar la aplicación.

Comparación de Herramientas:

Escala de 1 a 5 : donde 1 es muy malo, 2 Malo, 3 Regular, 4 Bueno, 5 Muy Bueno.

	Spring Boot	open Xava	.Net Core	Genexus	PhoneGap	Jquery Mobile
Configuración e Instalación	★☆☆☆☆	★★★★☆	★★☆☆☆	★★★★☆	★★★★☆	★★★★☆
Prototipado	★☆☆☆☆	★★★★★	★★★★☆	★★★★☆	★☆☆☆☆	★☆☆☆☆
Open Source	★★★★★	★★☆☆☆	★★★★☆	★☆☆☆☆	★★★★☆	★★★★☆
Documentación y Comunidad	★★★★★	★★★★★	★★★★★	★★★★★	★★★★☆	★★★★☆
Integración	★★★★★	★★★☆☆	★★★☆☆	★★★★☆	★★★☆☆	★☆☆☆☆
LDAP y Seguridad	★★★★★	★★★☆☆	★★★☆☆	★★★★★	★★★☆☆	★★☆☆☆
Diseño de ABMs	★★★★☆	★★★★★	★★★★☆	★★★★★	★★★★☆	★★★★☆
Consumo de Recursos	★★★★☆	★★★★☆	★★★★☆	★★★☆☆	★★★★☆	★★★★☆
Curva de Aprendizaje	★★★★☆	★★★★☆	★★☆☆☆	★★★★★	★★★★☆	★★★★☆
Usabilidad, look and feel	★★★★★	★★★★★	★★★★☆	★★★★☆	★★★★★	★★★★☆
Portabilidad	★★★★★	★★★★☆	★★★★☆	★★★☆☆	★★★★☆	★★★★☆
Dispositivos móviles	★★☆☆☆	★★★★☆	★★★★☆	★★★★★	★★★★★	★★★★★

Conclusión para desarrollo

Luego de haber revisado muchas herramientas como mencionamos en el presente documento, algunas de las cuales solo buscamos información y en las que nos pareció más interesantes desarrollamos unos mini prototipos. A partir de toda esta información recabada y de las pruebas realizadas, creemos que lo más conveniente es utilizar OpenXava más Liferay para la realización de la aplicación Web dado el ahorro de tiempo que genera la autogeneración de los ABMs, la curva de aprendizaje rápida, la posibilidades de un desarrollo ágil, el licenciamiento open source y los demás puntos mencionados más arriba. Con el tiempo que creemos ahorraremos en la primera etapa de definición de entidades y diseño de las ABMs, re invertiremos ese tiempo ahorrado en la resolución de problemas que no son resolubles con la versión free de OpenXava.

Para la plataforma móvil existen las alternativas que planteamos y por supuesto otras más, la construcción de un módulo para dispositivos móviles se descartó y quedó fuera del alcance del proyecto. Este es un de los puntos más interesantes para realizar un trabajo a futuro.

Otras Herramientas Utilizadas

A continuación haremos una breve descripción de otras herramientas que utilizamos a lo largo del proyecto y nos fueron de utilidad en distintas etapas del mismo.

Diseño

Para los mockups de diseño y diseño de interacción utilizamos Balsamiq mockups(10), esta herramienta nos pareció muy sencilla e intuitiva, simplemente tuvimos que crear nuestros bosquejos prácticamente como si los estuviéramos dibujando a mano con ésta herramienta creamos nuestro primer prototipo en un breve periodo de tiempo.

Datos de prueba

Para generar datos de prueba se utilizó generatedata(11), es un herramienta libre y de código abierto, que permite generar rápidamente grandes volúmenes de datos personalizados para uso en pruebas y llenado de Base de Datos.

Repositorio GitLab

Utilizamos GitLab como repositorio de código centralizado, esta herramienta nos permitió trabajar en toda la etapa de implementación del Sistema trabajando juntos y en simultáneo gracias a su Integración continua. También GitLab nos da otras funcionalidades como crear tareas y notificaciones, esto nos fue de ayuda para organizarnos el trabajo. Por último GitLab nos ofrece una serie de Estadísticas sobre el trabajo realizado que mencionaremos más adelante en este informe.

Sistema Operativo

Utilizamos Linux/GNU como sistema operativo de preferencia, Ubuntu Mate versión 16.04.4. Si bien también es posible usar un Sistema Operativo Windows, nos dimos cuenta que algunos errores en el Repositorio de Código se daban cuando se hacía push desde equipos con windows, por esta razón decidimos unificar criterios a la hora de programar, entonces por razones de compatibilidad y para mantener criterios de uso de software libre nos decidimos por UBuntu.

Entorno de Desarrollo

El entorno de Desarrollo es la herramienta que nos ayuda y nos facilita la construcción y desarrollo del código de las distintas partes del Sistema.
Utilizamos como IDE de desarrollo a Eclipse Oxygen en su versión 3a 4.7.30, la elección del IDE se corresponde porque es lo recomendado por OpenXava, posee licenciamiento libre y gratuito, porque es compatible con sistemas operativos Linux y Windows, es ideal para desarrollar todo tipo de aplicaciones Java, y por último Elipse posee una gran comunidad de usuarios y una basta documentación a la cual podemos apelar en caso de ser necesario.

JDK

JDK acrónimo de Java Development Kit (Kit de desarrollo de Java), son herramientas de desarrollo para la construcción de programas Java, su licenciamiento es gratuito, en nuestro sistema utilizamos la versión 8.

Requerimientos del Sistema y alcance del Proyecto:

Alcance del Proyecto

El alcance del proyecto es el siguiente:

- Elección de la herramienta de desarrollo, para realizar el sistema.
- Diseño del sistema.
- Construcción del sistema web que cumpla con los requerimientos del cliente.
- Implantación del sistema.

Fuera del Alcance

Queda fuera del alcance la construcción del sistema para dispositivos móviles.

Requerimientos

En este punto se detallan los requerimientos que fueron detectados en las reuniones con el cliente:

Requerimientos funcionales

Alta de consultante

El sistema debe permitir el ingreso de un consultante. El sistema identifica a los consultantes de manera inequívoca a través de su documento de identidad, además solicita los siguientes datos.

- CI
- Nombre completo
- Fecha de Nacimiento
- Estado Civil
- Domicilio
- Teléfono
- Sexo
- Celular
- Correo Electrónico
- Comentario

Alta de consulta

El sistema debe permitir el ingreso de un consulta tanto para un consultante ya creado como para un consultante nuevo, proporcionar un número identificador inequívoco para la consulta.
El sistema permite crear una consulta donde proporciona un identificador y la fecha del día, luego permite asociar un consultante a la consulta, seleccionando si el consultante ya existe o dar la posibilidad de ingresarlo de la forma Alta de consultante.
Para ingresar una consulta el sistema solicita los siguientes valores
Datos de Consulta:

- ID : El Sistema lo proporciona automáticamente
- Temática
- Estado
- El sistema debe permitir asociar un Consultante a la Consulta
- Datos de la Derivación
    - Organismo
    - Referente
    - Contacto
- El sistema debe permitir asociar un Docente a la Consulta
- Observaciones

Procesar una Consulta / Crear un Caso

El sistema debe permitir procesar una consulta, al procesar una consulta se crea un Caso con los datos de la Consulta.

Modificar un Caso

El sistema debe permitir modificar un caso a los distintos involucrados en él, un caso consta de los siguientes datos:

- Datos del Caso
    - Datos del Apoderado Documento y Nombre completo
    - Naturaleza de la Intervención
        Esta puede ser Asesoramiento, Declaración jurada, Certificado, Acta, Otros
    - Estado
    - Si requiere comisión especial
    - Notas
    - Ocupación
    - Ingreso mensual personal en pesos $
    - Grupo Familiar
    - Hijos a cargos y edades
    - Si paga alquiler y su valor
    - Apoderado
        Documento y Nombre

- Comentarios

El sistema emitirá un informe imprimible con dicha información, para que el consultante haga su declaración jurada y la pueda firmar.

Modificar Docente de un Caso

El sistema debe permitir modificar un caso, asociando distintos Docentes al Caso, se debe permitir escoger de los Docentes ya existentes en el sistema, en el momento de asociar el Docente al Caso se debe especificar si el Docente es responsable del Caso o no. El sistema también debe permitir desasociar un Docente de un Caso.

Modificar Estudiante de un Caso

El sistema debe permitir modificar un caso, asociando distintos Estudiantes al Caso, se debe permitir escoger de los Estudiantes ya existentes en el sistema.

Modificar Documentos de un Caso

El sistema debe permitir modificar un caso, asociando distintos Documentos al Caso, se debe permitir escoger archivos desde el sistema operativo.

Modificar Citas de un Caso

El sistema debe permitir modificar un caso, asociando distintas Citas al Caso. Una cita debe contar con los siguientes valores:

- Fecha
- Hora
- Tipo
- Estado
- Entrevistador
    - Este Puede ser un Docente o un Estudiante del Caso

Por otro lado la Cita debe poder modificarse, también Cancelarla o darla por Cumplida.
Al darse de alta un Cita el sistema debe crear una entrada en la Agenda del Estudiante o Docente asociado en la Cita.

Modificar Actividades de un Caso:

El sistema debe permitir modificar un caso, asociando distintas Actividades al Caso. Una actividad debe contar con los siguientes valores:

- Tipo
- Estudiante asignado
- Fecha de asignación

Por otro lado la Cita debe poder modificarse, también Cancelarla o darla por Cumplida
Al darse de alta una Actividad el sistema debe crear una entrada en la Agenda del Estudiante asociado en la Actividad.

Buscar un consultante

El sistema debe facilitar la búsqueda de un consultante al menos por los siguientes criterios: Por su número de cédula de identidad, por su nombre o por su apellido.

Modificar un consultante

El sistema debe permitir la modificación de un consultante, si la modificación incluye datos sensibles el sistema emitirá un informe imprimible con dicha información, para que el consultante haga su declaración jurada y pueda firmarlo.

Alta de Docente

El sistema permite la creación de usuarios con rol de docente, debe sincronizar los usuarios desde una base de datos de LDAP y a partir de allí crear un usuario con el Rol de Docente.

Alta de Funcionario

El sistema permite la creación de usuarios con rol de funcionario, debe sincronizar los usuarios desde una base de datos de LDAP y a partir de allí crear un usuario con el Rol de Funcionario.

Alta de Estudiante

El sistema permite la creación de usuarios con rol de Estudiante.
El sistema debe sincronizar los usuarios desde una base de datos de LDAP y a partir de allí crear un usuario con el Rol de Estudiante.

Buscar un consulta

El sistema debe facilitar la búsqueda de un consulta al menos por los siguientes criterios: Por identificador y por fecha de consulta, por cédula de identidad, por nombre y apellido de consultante.

Aceptar/Rechazar Consulta

El sistema permite a los usuarios con el rol necesario, aceptar o rechazar una consulta.
En el caso que se rechace una consulta el sistema debe permitir ingresar un comentario con los motivos del rechazo, con esto el sistema pasa un estado rechazada a la consulta y allí finaliza su ciclo.
En el caso que se acepte la consulta el sistema pasa a un estado de Aceptada la consulta y se crea un caso a partir de ella.

Un caso está compuesto por la siguiente información:

- Consultante asociado
- Un Docente responsable asignado al caso
- Un grupo de Estudiantes asignados al caso
- Una colección de citas
- Una colección de Tareas y Actividades
- Una colección de Documentos asociados

Login de Usuario

Para ingresar al sistema debe hacerlo con credenciales correctas.
El sistema debe validar las credenciales de un usuario que quiere loguearse en el, y debe hacerlo contra dos base de datos LDAP, el usuario tiene que tener credenciales válidas en una de la base de datos, en estas se guarda la información de funcionarios y de estudiantes respectivamente.

Seguridad

El sistema debe permitir o no, el ingreso de información de acuerdo a distintos Roles.
Los Roles son los siguientes, Técnico Administrador, Funcionario Administrativo, Docente, Estudiante, cada uno de estos roles debe tener potestades y restricciones para las distintas funcionalidades del sistema.

Requerimientos no funcionales

Aquí se detallan los requerimientos, condiciones o restricciones que especifica el cliente.

Funcionamiento en Linux

El servidor de la aplicación debe poder instalarse y ejecutarse en sistemas operativos Linux.
El desarrollo de la aplicación debe poder diseñarse, implementarse y configurarse desde un sistema operativo Linux, aunque por razones de comodidad, conocimiento o rendimiento puede hacerse de un sistema operativo distinto.
La aplicación debe poder accederse desde equipos que tengan sistemas operativos Linux, por lo tanto como mínimo debe funcionar sin inconvenientes desde el navegador Mozilla Firefox.

Login contra LDAP

El sistema debe contar con control de acceso, este control de acceso debe autentificarse contra uno o dos servidores LDAP. Estos servidores contarán con la Base de Datos de Estudiantes y de Docentes/Funcionarios, el sistema debe debe detectar si el usuario que se está haciendo el login es un Estudiante o un Funcionario.

Almacenamiento de Documentos y Archivos

En el sistema debe de poder configurarse donde se guardaran los archivos que se adjunten, los archivos a adjuntar puede ser al menos de los siguientes tipos: documento de texto, hoja de cálculo, documentos pdf y documentos de imagen. Esto puede ser en un directorio específico o bien dentro de la Base de Datos.

Sincronización con sistema de Consultorio Jurídico

El sistema debe tener interfaces con el sistema de Consultorio Jurídico, estas interfaces a pueden ser de sincronización de Consultantes y de Casos. Tambíen cualquier otra interfaz que se detecte como necesaria en las próximas etapas.

Control de acceso por perfiles

El sistema debe contar con distintos perfiles de usuarios, estos perfiles deben de poder administrarse por un usuario con rol de administrador o similar. Cada usuario tendrá distintos permisos y potestades de acuerdo a su Perfil.

Robustez

El sistema debe ser robusto y confiable, es decir debe permanecer en un estado estable y seguir funcionando incluso si sufre perturbaciones o interacciones inesperadas.

Usabilidad

El sistema es fácil de usar, rápido e intuitivo incluso para usuarios que no han sido entrenados en el uso del mismo.

Concurrencia

El sistema debe seguir funcionando y soportar la carga de trabajo cuando varios usuarios se conecten al mismo tiempo.

Confiabilidad

El sistema garantiza la confiabilidad y la seguridad de los datos almacenados, estos pueden ser consultados y modificados en cualquier momento y por todos los usuarios del sistema.

Solución Propuesta

Aquí se muestran los aspectos más relevantes de la solución propuesta en relación con el alcance y los objetivos planteados en el presente documento.
También se exponen los lineamientos de análisis, diseño, construcción e implantación. Esto se detalla en distintos puntos como diagrama de Base de Datos, requerimientos, diseño de casos de prueba, arquitectura del sistema y todo lo que necesitamos para construir el Sistema Clínica Notarial.

Selección de las Herramientas

Para la realización del sistema nos inclinamos por la opción de OpenXava más Liferay, estas herramientas nos permiten un desarrollo rápido y ágil, con ellas podemos desarrollar todos los requerimientos que surgieron en las distintas instancias de reunión con el cliente.
OpenXava nos provee toda las funcionalidades de ABMs que nos ahorra todo el desarrollo de la su interfaz gráfica y toda la construcción de las tablas y la base de datos, esto hace que nos enfoquemos en un buen diseño de las clases. A la hora de buscar una solución para el manejo de Usuarios y Roles, nos encontramos que OpenXava no nos resuelve este problema en la versión Open Source. El módulo de Usuarios y Roles está disponible sólo en la versión paga XavaPro. Para resolver este problema investigamos por varias opciones, una opción era desarrollar el módulo de usuarios nosotros, la otra era tratar de integrar Spring Security en OpenXava y la opción restante era utilizar Liferay como portal y delegar el manejo de Usuarios y Roles. Esta última opción nos pareció la más adecuada ya que contaba con documentación en el sitio web de OpenXava de cómo realizar dicha integración, ademas de que la integración con LDAP es muy sencilla. Liferay nos complementa las funcionalidades de manejo de usuarios y roles con estas funcionalidades ya disponibles no tenemos que reinventar la rueda, y también nos brinda la interfaz necesaria para el Login contra los servidores LDAP de funcionarios y estudiantes.
Para la persistencia, manejo y procesamiento de los datos usamos el motor de base de datos MySQL sabemos que su eficiencia y seguridad está altamente probada.
Para las documentos que almacenamos en nuestra aplicaciones utilizamos el File System del sistema operativo Linux. Es responsabilidad del administrador de red, crear carpetas con los permisos necesarios y también de proveer un método de respaldo y recuperación.
Para el Servidor de aplicaciones utilizamos Apache Tomcat ya que es cien por ciento compatible con todas las herramientas anteriormente mencionadas y está más que probado su correcto desempeño.
Por último como navegador web sugerimos utilizar Mozilla Firefox por ser compatible con cualquier distribución Linux que se instale en los equipos de escritorio de las oficinas del Consultorio Jurídico.
Con esta elección de herramientas garantizamos uno de los principales requerimientos no funcionales, el software es libre y gratuito en todas las etapas del proceso de realización del Sistema y en todas las etapas de su uso.

OpenXava

Se desarrollara la aplicación web usando OpenXava (Ver capítulo Uso de OpenXava en este documento) ,con su arquitectura la basada en Componentes de Negocio, cuando esté finalizada se generarán los portlets para poder añadirlos al Portal.

Portal

Liferay[12] es un portal de gestión de contenidos desarrollado en Java y de código abierto, Liferay nos ofrece un alojamiento web dinámico para alojar nuestra aplicación generada con OpenXava.
Decidimos utilizar Liferay con OpenXava por los siguientes motivos:

- Control de Usuarios y Roles : Utilizando las herramientas que ya nos ofrece Liferay no tenemos que rediseñar un sistema con control de Usuarios y Roles, simplemente definimos los Roles y sus permisos dentro del portal por un lado, luego creamos y asignamos los usuarios a los roles previamente creados. Con estos simples pasos tenemos una aplicación con seguridad por control de Usuarios.
- Integración con LDAP : Aprovechamos la compatibilidad de Liferay con LDAP para tener una aplicación segura y privada. Simplemente se debe asignar los usuarios logueados en la aplicación a un rol dentro del Portal, esto se debe hacer una sola vez para cada Usuarios. Con esta funcionalidad no tenemos que diseñar una política de contraseñas sino que utilizamos la de LDAP, tampoco los administradores deberán crear un nuevo conjunto de cuentas de usuario, sino que usaran las ya existentes. De esta manera los administradores de la aplicación, los desarrolladores y los usuarios finales están simplificando sus tareas.
- Generación de Portlets : Para componente de Negocio creado en OpenXava, generamos un portlet, al conjunto de portlets desarrollados le hacemos un Deploy en Liferay, luego en Liferay asignamos roles a dichos Portlets y rápidamente tenemos nuestra clases Java funcionando dentro de la aplicación. Con relativa simpleza y velocidad podemos añadir nuevas componentes de negocio en nuestra aplicación o corregir alguno que no esté funcionando de la manera esperada.

Servidor Web

El servidor web es un Apache Tomcat, todo el Sistema se aloja en el servidor.
Elegimos utilizar un servidor Apache por que es fácil de instalar, porque es libre y gratuito, por que es el ideal para aplicaciones java, es compatible con sistemas operativos Linux y porque tiene gran cantidad de documentación y forma parte de una gran comunidad.

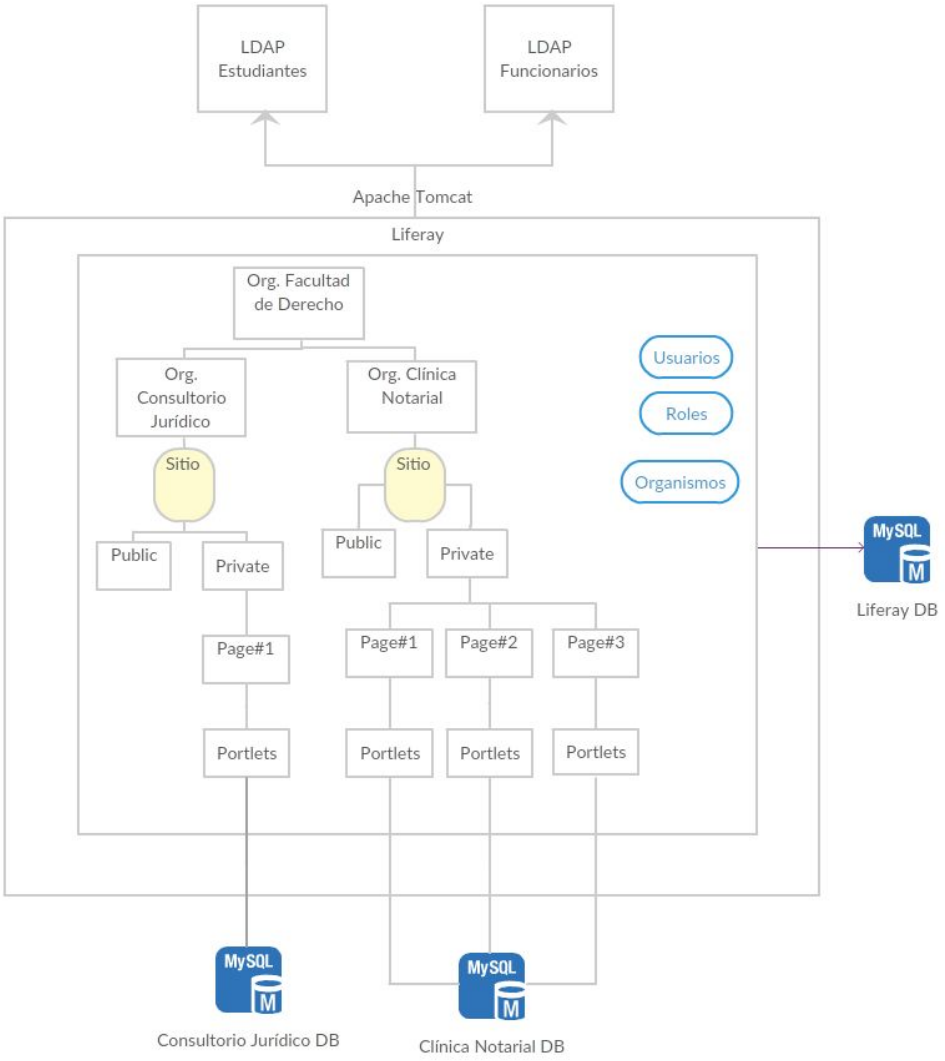
Base de Datos

El motor de base de datos es MySQL, dentro del servidor de base de datos tenemos las siguientes bases lportal (base de datos de Liferay) y clinica_notarial (base de datos de la Clínica Notarial).
Dentro de cada organización existe un sitio y para cada sitio una base de datos, por otro lado también existe un Base de Datos para la administración de Liferay.
En un principio tenemos una solo sitio, el de la Clínica Notarial, pero este se puede ampliar al Consultorio Jurídico u otro que se necesite.
Para conectarnos a la base de datos de la Clínica Notarial utilizamos un conector JDBC, la de Liferay ya viene configurada y lista para usar.

Arquitectura

Se plantea una aplicación web, embebida en un portal Liferay también tiene procesos de sincronización para obtener los usuarios de tipo Estudiantes y de tipo Funcionarios, cada uno de ellos sincroniza contra distintas bases LDAP. A continuación detallamos los distintos componentes de la arquitectura.

Arquitectura del sistema



Diseño

Mockups

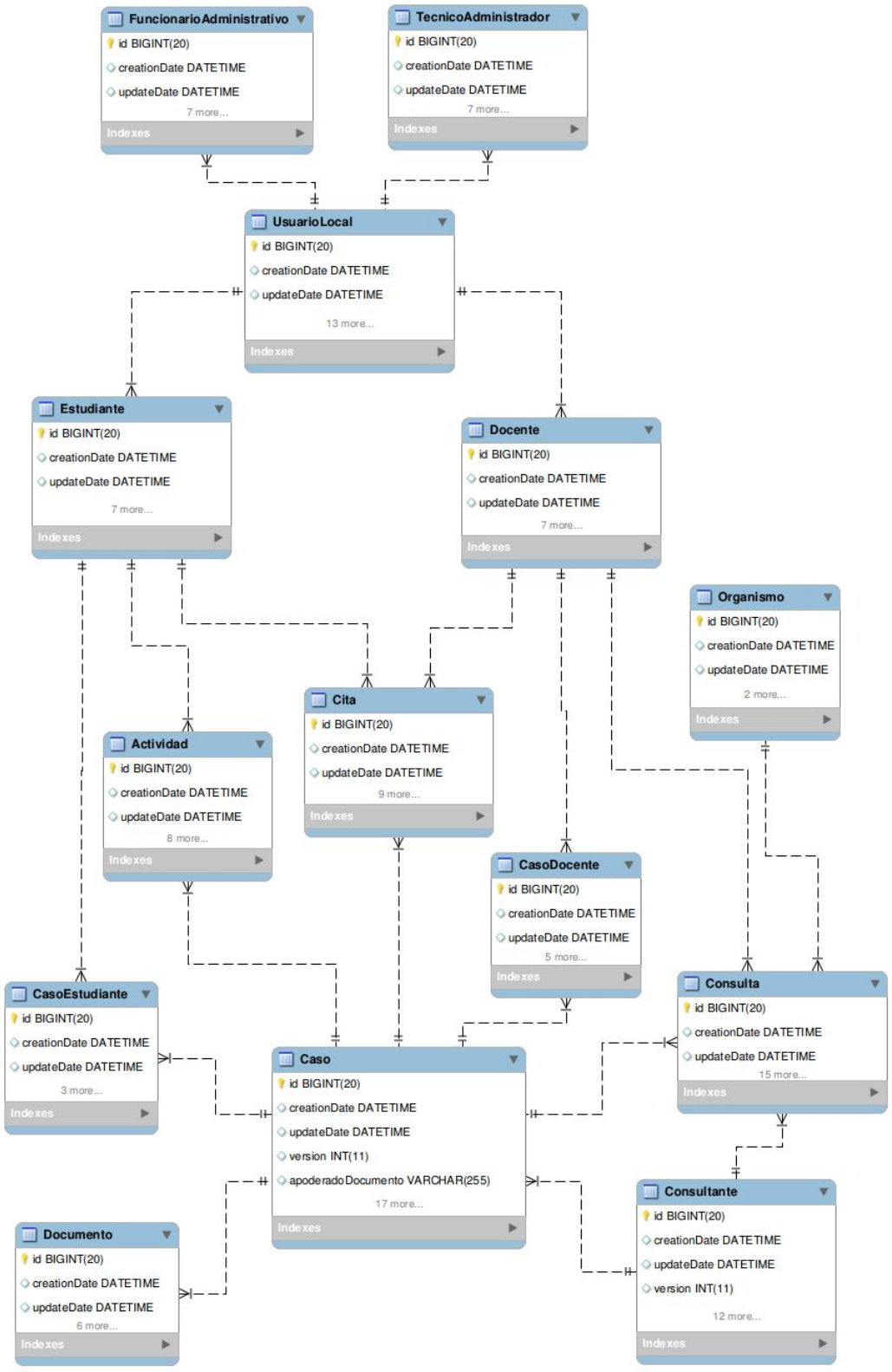
En una primera etapa del diseño realizamos mockups interactivos para representar los casos de uso más relevantes. Estos mockups fueron presentados ante el cliente y este nos dio su aval para continuar con la construcción del Sistema.
Si bien estos Mockups distan un poco de la solución final, ya que cuando los realizamos todavia no teniamos elegida la herramienta de desarrollo a utilizar, consideramos que esta etapa fue muy importante ya que se presentó una primera aproximación del sistema al cliente, este acercamiento al Sistema fue visual e interactivo. Le mostramos al usuario el diseño del sistema y el diseño de interacción, esta interacción fue pensada para que el usuario final pueda entender y realizar las acciones de manera sencilla y eficiente.
Ver mockups en Anexo 1

Diagrama de Entidad Relación

A continuación explicamos el diagrama de Relación, si bien siempre es importante tener un diseño claro desde un comienzo, en el sistema que estamos construyendo cobra una vital importancia ya que por utilizar OpenXava el mapeo de estas entidades con objetos Java y por subsiguiente con las tablas de la base es casi un espejo.

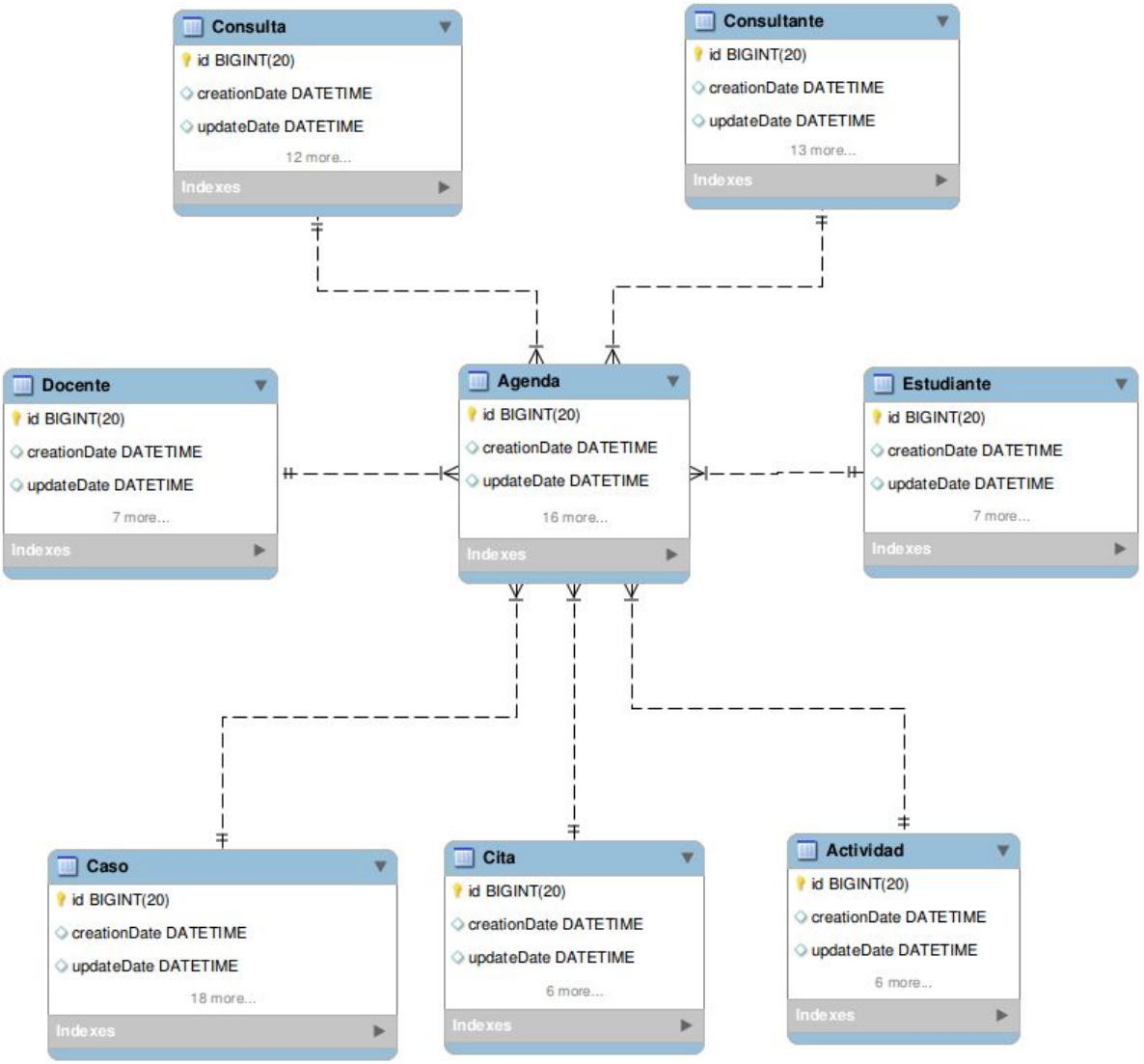
En la siguiente figura se muestra la relación entre la entidad , el diseño y los datos

Como se muestra en la figura siguiente figura a nuestro entender 3 entidades principales UsuarioLocal, Caso y Cita.
UsuarioLocal maneja toda la información de los usuarios y heredan de él, las clases Funcionario, Administrador, Docente y Estudiante , cada uno de estos tipos de usuario contiene distintos atributos y relaciones.
Cita: Es la entidad que relaciona los estudiantes y docentes con sus diferentes casos.
Caso: Es una entidad más importante, en el se maneja toda la información Actividades, Consultas, Citas, Documentos y que crea un nexo entre el Consultante (usuario de la Clínica Notarial) y los Docentes y Estudiantes (servidores de la Clínica Notarial) en este caso cuando decimos Clínica Notarial hablamos de la Oficina y no del Sistema y con usuario nos referimos a las personas que necesitan de la Clínica Notarial y con servidores a las personas que brindan el servicio.


FuncionarioAdministrativo
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
7 more...
Indexes
TecnicoAdministrador
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
7 more...
Indexes
UsuarioLocal
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
13 more...
Indexes
Estudiante
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
7 more...
Indexes
Docente
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
7 more...
Indexes
Organismo
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
2 more...
Indexes
Cita
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
9 more...
Indexes
Actividad
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
8 more...
Indexes
CasoDocente
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
5 more...
Indexes
CasoEstudiante
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
3 more...
Indexes
Consulta
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
15 more...
Indexes
Caso
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
version INT(11)
apoderadoDocumento VARCHAR(255)
17 more...
Indexes
Consultante
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
version INT(11)
12 more...
Indexes
Documento
id BIGINT(20)
creationDate DATETIME
updateDate DATETIME
6 more...
Indexes

Relación entre Agenda y el resto de las entidades

El sistema ya provee las funcionalidades de gestionar consultas, casos, citas y actividades de estos últimos pero no encontramos con la dificultad funcional que era muy poco práctico para el usuario final poder tener visibilidad del volumen de trabajo ya planificado para los siguientes días y menos aún si esta información la necesitaba más en detalle (por docente, por tipo de tarea, etc.). Para solucionar esta problemática se incorporó la idea de la Agenda que básicamente es una entidad que aplana los ingresos de consultas, casos, citas y actividades de formar de tener una visión directa de todo lo planificado en ese sentido y pudiendo acceder rápidamente a la información de que docente y estudiante o consultante es el afectado. La agenda no sólo se utiliza como un visor de tareas planificadas sino también como un punto de entrada para ejecutar procesos de forma indirecta a través de la misma como pueden ser: *Cumplir una cita, Cancelar una cita, Cumplir una actividad, procesar una consulta, etc.* Lo anterior implica que se aplica una acción sobre la agenda que afecta su estado actual pero también de forma indirecta se afecta el estado de la entidad directamente relacionada a la misma (Actividad, Consulta, Caso, Cita).



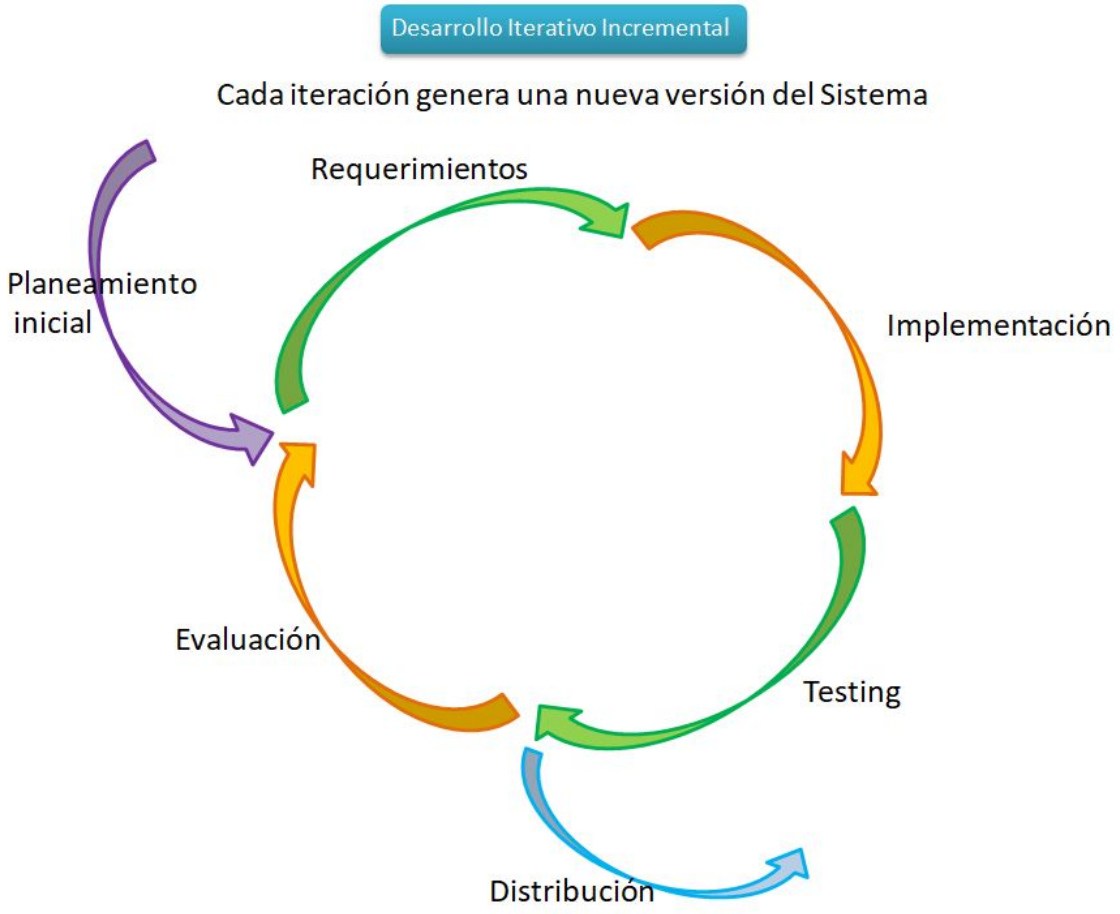
Implementación

Para la implementación utilizamos una metodología de desarrollo iterativa incremental[14], esta metodología nos permite en cada iteración avanzar paulatinamente con la resolución de requerimientos. En cada iteración se desarrollan un conjunto de requerimientos nuevos y mejorando los ya existentes, esto hace que el producto final evolucione.

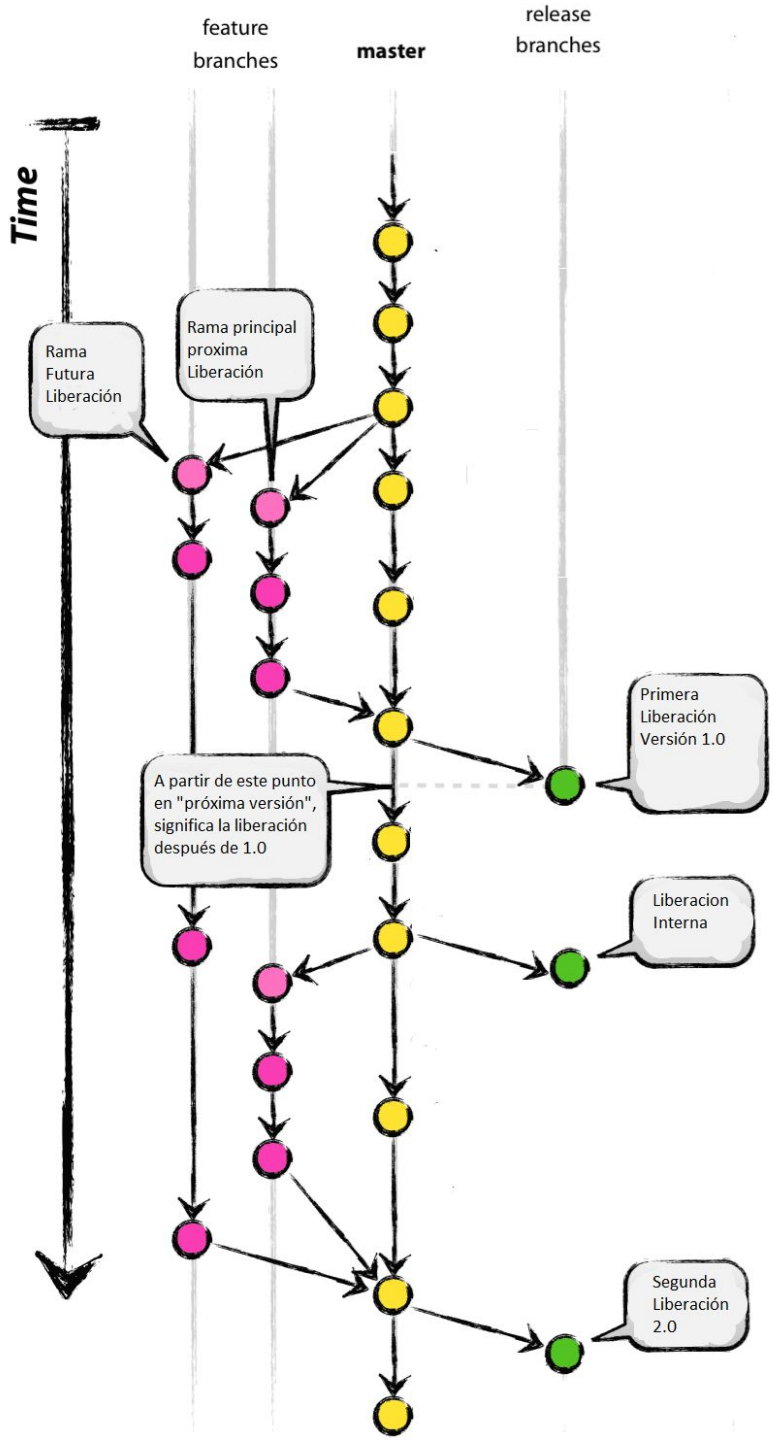
Ventajas de este Proceso

- En cada iteración los desarrolladores ya tienen un camino aprendido y ese conocimiento es utilizado para hacer más eficiente la iteración siguiente.
- El cliente obtiene resultados utilizables desde las primeras iteraciones.
- Conocer el progreso del proyecto.
- Mitigar los riesgos desde un comienzo.
- Como en cada iteración se entregan requerimientos terminados, se minimizan los errores haciendo que aumente la calidad del producto.

En la siguiente figura vemos un esquema de como fue nuestro desarrollo iterativo.



La recomendación que seguimos para modelar el flujo de trabajo fue la siguiente:
Por una lado la Rama principal (Master), por otro lado se van abriendo distintas ramas de trabajo incorporando nuevas características y funcionalidades, cuando esas funcionalidades están prontas, los cambios se combinan en el Master. Por último cuando se obtienen los requerimientos suficientes como para liberar una versión, se abre una rama de liberación para cada versión, con esto nos aseguramos de saber cual es el código correspondiente a cada liberación. A modo ilustrativos nos basamos en la siguiente figura, este modelo es una simplificación de este otro que es explicado con mayor detalle[15]



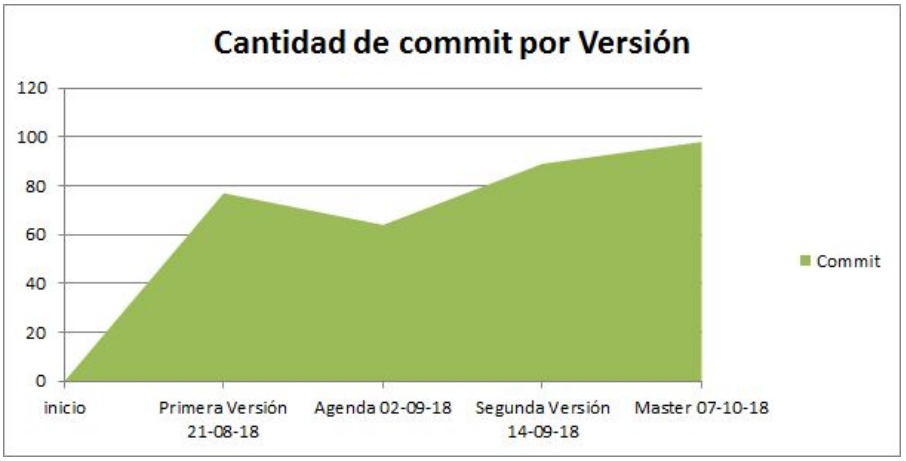
Cuando cualquiera de los integrantes del grupo comenzaba a trabajar en la implementación debía crearse un nuevo branch, luego realizar sus cambios y luego de que estuviesen testeados recién en ese momento se podía hacer un push en el repositorio, si se detectaban conflictos debían resolverse si no había conflictos el cambio quedaba actualizado en el repositorio.
Para la organización de las distintas liberaciones del sistema si utilizaron distintas ramas de código.
Las ramas o branch fueron:

- Master: Rama principal de código, sobre ella se realizaron las modificaciones a lo largo de la implementación aquí se encuentra la versión final del código correspondiente a la versión Productiva del Sistema con fecha 07-10-18
- release/version-2.0: En ella se realizó la segunda liberación de Prototipo con las funcionalidades de agenda ya incorporadas,versión con fecha 14-09-18.
- Agenda: En esta rama se incorporó todas las funcionalidades correspondientes a la agenda, se realizó en una rama separada debido a la complejidad y la reestructura que significaba dentro de la base de datos. Se utilizó como versión de prototipo interno, luego de probado se mergeo con la rama principal. Versión con fecha 02-09-18
- release/version-1.0: En ella se realizó la primera liberación de Prototipo con fecha 21-08-18

El trabajo realizado fue parejo y uniforme a lo largo del tiempo de la etapa de implementación, en la siguiente tabla se muestran el periodo, la cantidad de commit y su promedio que se realizaron en cada una de las versiones.

	Commit	Inicio	Fin	Cant. Días	Commit por día
Primera Versión 21-08-18	77	09-06-18	21-08-18	73	1.05
Agenda 02-09-18	64	09-06-18	02-09-18	85	0.75
Segunda Versión 14-09-18	89	09-06-18	14-09-18	97	0.92
Master 07-10-18	98	09-06-18	07-10-18	120	0.82

Gráfico con la cantidad de commit por versión.



A continuación se muestra una gráfica sobre los commits realizados a lo largo del tiempo en la rama master.

Primera Iteración Implementación del Prototipo

El objetivo de esta primera iteración fue crear un prototipo del sistema y validar que el proyecto fuera viable desde el punto de vista tecnológico por estos motivos ésta iteración fue realizada mucho antes en el tiempo y no está contemplada en las tablas y gráficas anteriores.

Los puntos fuertes a validar fueron:

- Crear un ABM de una entidad simple con OpenXava.
- La integración de OpenXava con Liferay, generar un portlet con OpenXava y deployarlo en Liferay.
- El manejo de Seguridad con Liferay (Autenticación y Autorización).
- Integración de Liferay con LDAP.

Los resultados de esta primera versión fueron ampliamente satisfactorios, pudimos resolver todos los puntos que nos planteamos a priori, con eficiencia y agilidad.
Cumplir con todos estos puntos fue determinante a la hora de decidirnos para utilizar OpenXava y Liferay como herramientas de desarrollo.

Segunda Iteración Implementación del versión Prototipo 1.0

En esta segunda iteración los objetivos principales fueron los siguientes:

- Crear una versión utilizable del producto cumpliendo con los Casos de Uso más representativos.
- Implantarlo en un ambiente lo más parecido al real que nos fuera posible.
- Obtener una primera devolución del usuario final, para retroalimentarnos y seguir avanzando

Los Casos de uso que seleccionamos para esta primera versión fueron los siguientes:

- Seguridad

    - Definición de Roles y Permisos
    - Integración con LDAP de la Facultad de Derecho
- Mesa de entrada - Funcionarios Administrativos
    - ABM de Consultantes
    - ABM de Consultas
- Casos para el usuario Técnica / Administrativa
    - ABM de Casos,
    - ABM de Documentos para un Caso
    - Comentarios para un Casos
    - Asignación de Roles a Usuarios Generales
    - ABM de Codigueras (Tablas básicas del sistema)

Implantación y hosteo del sistema

En un primer momento hosteamos el sistema en el sitio Digital Ocean, y simulamos un LDAP para usuarios y funcionarios. Esto nos permitió a los desarrolladores y al tutor y su grupo de trabajo acceder a un sitio donde los datos estaban unificados, con esto fuimos resolviendo los errores que íbamos detectando.

En segunda instancia hosteamos el sistema directamente en un servidor de Test de la facultad de derecho. Esto fue un gran paso ya que después de algunos idas y vueltas con el tutor y la oficina de sistemas de la facultad de derecho, logramos implantar el sistema en un host de la facultad con todo lo que eso concierne, seguridad, velocidad y también acceso a las bases de datos LDAP reales de funcionarios y estudiantes.

Devolución del usuario.

Con el sistema implantado el usuario final pudo ver y acceder a una versión estable del sistema. Ellos se vieron muy conformes con lo mostrado, entre todos detectamos algunos errores u oportunidades de mejora para agregar a la siguiente versión.

Tercera Iteración, Implementación de Agenda

Los objetivos de esta versión fueron incorporar las funcionalidades de Agenda.
En esta iteración se incorporaron todas las funcionalidades de agenda, esto requirió de una reestructura importante a nivel de Base de Datos y una gran incorporación de objetos y clases, esta versión no fue liberada al cliente sino que fue testeada internamente por nosotros, luego de resolverse todas las funcionalidades se incorporaron estos cambios a la rama master del repositorio para ser liberados en la siguiente iteración. Con la incorporación de la agenda en el sistema se buscó consolidar en una única vista toda la información referente a consultas, casos, actividades entre otras entidades. Una vez en funcionamiento, la agenda le permitirá al usuario realizar varios de los procesos más importantes desde un único punto del sistema, pudiendo tomarse esta pantalla como el escritorio de trabajo de cada usuario.

Cuarta Iteración Implementación del Prototipo versión 2.0

Los objetivos de esta cuarta iteración fueron:

- Corregir errores de la segunda iteración.
- Agregar funcionalidades faltantes para cumplir con el alcance.
- Incorporar funcionalidades de la agenda, vista en la tercera iteración.
- Recibir una devolución final de los usuarios que van a utilizar el sistema.

Se realizó la corrección de Errores de la versión 0.1 y se agregaron las siguientes funcionalidades:

- ABM de Citas para un Caso
- ABM de Actividades para un Caso
- Funcionalidad de Agenda

Devolución del usuario.

Con el sistema implantado se corrigieron los errores de la versión anterior. Detectaron algunos oportunidades de mejora para agregar a la siguiente versión. Por ejemplo imprimir declaración de ingresos e imprimir información de cita.

Quinta Iteración Implementación del Sistema real versión 1.0

Los objetivos de esta iteración son:

- Ampliar los nuevos requerimientos solicitados
- Implantar la versión del sistema que será utilizada por los usuarios

Se realizó la corrección de Errores de la versión segunda de prototipo y se agregaron las siguientes funcionalidades:

- Imprimir la declaración de ingresos
- imprimir la cita donde conste día y hora y nombre del Docente que se ocupará de su caso

Toda esta versión está implantada en los servidores de la facultad de derecho, con seguridad y control de acceso mediante los servidores LDAP de facultad de derecho.
De esta manera se da una cobertura del 100% de los casos de usos planteados en el alcance del proyecto.

Test

Nuestra primera idea fue realizar Test automatizados con Junit y hacer pruebas de regresión con cada liberación de una nueva versión. En la primera versión logramos preparar alguno de eso test utilizando las capacidades extendidas de OpenXava con JUnit esto no fue posible en las siguientes versiones por la integración entre OpenXava y Liferay, este inconveniente lo explicamos en el capítulo “Uso de OpenXava” del presente documento.

Como alternativa para testear nuestro sistema utilizamos una técnica de testing unitarios, esto nos permite probar los distintos casos de uso por separados simplificando la integración del sistema y facilitando la detección de errores. A continuación se enumeran los casos de uso testeados con la solución final de la herramienta. En el anexo 2 “Documento de casos de uso” se documentan y detallan cada uno de estos casos. En el anexo 3 “Documento de Testing” se detallan las pruebas realizadas para cada uno de estos casos de uso.

Casos de uso

En la siguiente tabla se muestra la relación de código de Caso de uso con su nombre en los Anexos, Documentos de casos de uso y Documento de Testing.

- RF-01 Alta de consultante
- RF-02 Modificar un consultante
- RF-03 Alta de consulta
- RF-04 Procesar consulta
- RF-05 Alta de caso
- RF-06 Modificación de caso
- RF-07 Alta de comentario en un caso
- RF-08 Alta de documento en un caso
- RF-09 Alta de cita en un caso
- RF-10 Modificación de cita en un caso
- RF-11 Cancelación de citas en un caso
- RF-12 Cumplir citas en un caso
- RF-13 Alta de actividad en un caso
- RF-14 Modificación de actividad en un caso
- RF-15 Cumplir actividad en un caso
- RF-16 Cancelación de actividades en un caso
- RF-17 Alta de docente
- RF-18 Modificación de un docente
- RF-19 Alta de estudiante
- RF-20 Modificación de un estudiante
- RF-21 Alta de funcionario administrativo
- RF-22 Modificación de un funcionario administrativo

- RF-21 Alta de técnico administrativo
- RF-22 Modificación de un técnico administrativo

Instalación e Implantación

Para facilitar la tarea de instalación e implantación del Sistema, a nosotros como creadores del mismo, al personal de la oficina técnica de la Facultad de Derecho y por sobre todas las cosas a futuros desarrolladores y administradores, creamos un pequeño manual en donde se mencionan punto por punto las tareas a realizar para Instalar, Desarrollar, Implantar y respaldar el Sistema. Toda esa información se encuentra disponible en el Anexo 5 "Manual para desarrolladores y administrador".

Manual de Usuario

Para facilitar y familiarizar a los usuarios más frecuentes del Sistema como lo son los funcionarios administrativos de la Clínica, y a futuros usuarios como Docentes y Estudiantes, contamos con un Manual de Usuario, en él se muestran las principales características propias del Sistema y de OpenXava en Genera. Toda esa información se encuentra disponible en el Anexo 4 "Manual de usuario".

Uso de OpenXava

En este capítulo mencionaremos los aspectos más relevantes del uso de OpenXava desde el punto de vista de un desarrollador. Como ya vimos anteriormente en este informe el paradigma de OpenXava es completamente distinto a los Framework MVC o Genexus que son los paradigmas a los que estamos acostumbrado como desarrolladores. Entonces uno de los primeros cambios que tuvimos que hacer fue el de nuestras cabezas y pensar distinto.
OpenXava se muestra como un software agil y rápido, creando simplemente las entidades como clases java ya podemos prácticamente crear una aplicación completa, en este capítulo veremos según nuestra experiencia que tan real es eso, que ventajas y desventajas de uso fuimos encontrando, las lecciones aprendidas y las expectativas que teníamos en comparativa con la realidad.

Filosofía

Recordemos que el paradigma de OpenXava es una aplicación orientada al modelo, las clases java son el centro de la aplicación y contienen la lógica del negocio como es la estructura de datos, los cálculos , las validaciones y los procesos orientados a los datos . El modelo es lo importante y lo demás depende de él, por ejemplo la Interfaz Gráfica.

OpenXava utiliza un desarrollo dirigido por el modelo (MDD)

Arquitectura

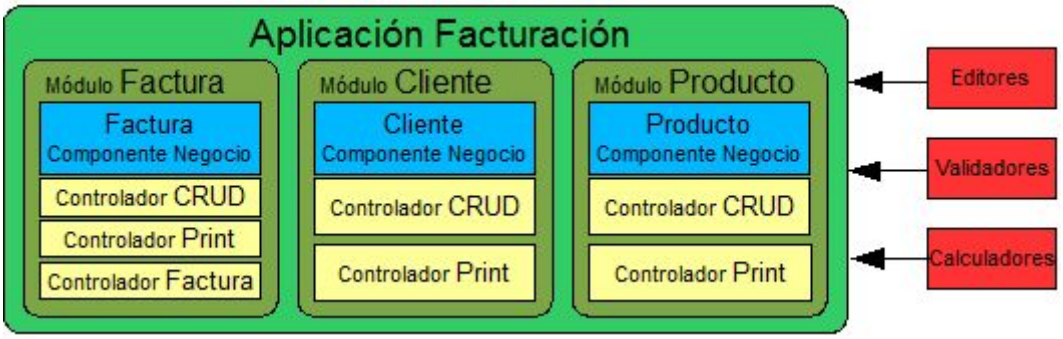
El modelo MDD determina que solo se debe desarrollar el modelo de una aplicación y el resto se generará a partir de dicho modelo.

El modelo se puede representar con una notación gráfica como es UML o bien mediante una notación textual, el uso de MDD es muy complejo, por lo tanto OpenXava toma esa idea pero la utiliza de manera muy simplificada, usa clases java para definir el modelo y no usa generacion de codigo. Se dice que OpenXava es un Marco de trabajo ligero dirigido por el modelo.

En la arquitectura openxava todos los artefactos de software se organizan a través del componente de Negocio como muestra la siguiente figura

Solo para contrastar vemos como sería la arquitectura de una aplicación MVC.

El componente de Negocio son lo central de las aplicaciones OpenXava, pero es necesario agregar elementos adicionales para ajustarse a los requerimientos.
En la siguiente figura se muestra como se vería una aplicación desde el punto de vista del desarrollador

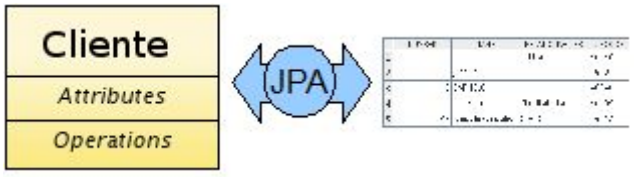


De esta manera la interfaz de usuario, el acceso a la base de datos y la lógica de negocio se encuentra toda en un mismo lugar, todo se reduce a una clase Java con Anotaciones. Esto tiene la siguiente ventaja, si quieres modificar algo de un componente de negocio ya sea agregar un campo a la base, un comportamiento de determinado atributo o simplemente la interfaz, alcanza con solo ir ir modificar la clase java correspondiente. Y no hay necesidad de modificar una multiplicidad de clases y archivos.
Aparte de componentes de negocio dentro de OpenXava contamos con módulos, controladores, editores, validadores y calculadores.
Controladores: Un controlador es una colección de acciones. Para el usuario, las acciones son botones o vínculos que él puede clickear, para el desarrollador son clases con lógica a hacer cuando el usuario pulsa en esos botones. Los controladores definen el comportamiento de la aplicación y normalmente son reutilizables. OpenXava incluye un conjunto de controladores predefinidos pero también se puede definir nuestros propios controladores.
Editores: Componentes de la interfaz de usuario para definir la forma en que los miembros de un componente de negocio son visualizados y editados. Existen editores predefinidos o también pueden personalizarse. Por ejemplo un campo fecha puede editarse con un calendario.
Validadores: Lógica de validación reutilizable que podemos usar en cualquier componente de negocio. Por ejemplo si un campo debe ser Fecha o Numérico o cualquier otra validación que se requiera.
Calculadores: Lógica de negocio reutilizable que puedes usar en algunos puntos de los componentes de negocio. Por ejemplo un campo edad Calculado a través de la fecha de nacimiento.

Las Clases Java con Anotaciones son claves para desarrollar una aplicación OpenXava y se podría desarrollar una aplicación completa de esta forma, de todas maneras Openxava ofrece la posibilidad de generar extensiones por ejemplo para personalizar la interfaz de usuario pudiendo usar JSP, JavaScript, HTML o cualquier otra tecnología de presentación para crear un Interfaz de usuario personalizada.

JPA

OpenXava utiliza JPA para el manejo de la persistencia de la aplicación, el desarrollador solo escribe los objetos, estos objetos se declaran como persistentes y es responsabilidad del motor JPA leer y grabar los datos desde la aplicación a la base de datos, JPA ahorra de escribir mucho código SQL.
JPA tiene dos aspectos diferentes por un lado las anotaciones para marcar las clases como persistentes y en segundo lugar la API para leer y escribir en la base de datos.



Anotaciones
Más adelante hablaremos de las anotaciones, pero como adelanto podemos decir que la anotación @Entity es la marca un clase como persistente, luego existen un conjunto muy grande de anotaciones para definir nombres de tablas y columnas, claves, tipos de datos, relaciones y colecciones entre otros.

API JPA
La clase de JPA más importante es javax.persistence.EntityManager. Un EntityManager te permite grabar, modificar y buscar entidades.
Con Openxava es bastante simple realizar estas acciones aquí mostramos algunos ejemplos
Código para grabar:

```
Docente docente = new Docente();
docente.setNumero(1);
docente.setNombre("JUAN");
XPersistence.getManager().persist(docente);
```

Código para modificar, primer se debe buscar el registro:

```
Docente docente = XPersistence.getManager()
   .find(Docente .class, 1);
cliente.setNombre("JUAN PEREZ");
```

JPA también permite el uso de consultas SQL, acá mostramos un caso simple:

```
Docente docente = (Cliente) XPersistence.getManager()
   .createQuery(
      "from Docente d where d.nombre = 'JUAN')")  // Consulta JPQL (1)
   .getSingleResult();  // Para obtener una única entidad (2)

List docentes= XPersistence.getManager()
   .createQuery(
      "from Docente d where d.nombre like 'JUAN%')")  // Consulta JPQL
   .getResultList();  // Para obtener una colección de entidades (3)
```

Instalación

La instalación del IDE Eclipse Oxygen nos resultó muy sencilla tal como se menciona en el capítulo Manual de Instalación, y en unos simples pasos y configuraciones ya estamos en condiciones de ejecutar las aplicaciones de prueba que trae el IDE o de comenzar a trabajar para diseñar nuestras propias aplicaciones. Para ello debemos Crear el Proyecto, configurar el Tomcat dentro de Elipse y preparar la base de datos el IDE ya viene con una embebidas que se ejecuta en memoria y es útil para los primeros pasos.

Uso

En primer lugar veremos cómo se trabaja desde OpenXava y lo explicaremos de la manera más detallada posible.
En esta sección mostraremos y haremos una breve descripción de cómo se modela en Openxava,obviamente solo mostraremos algunos puntos que nos parecieron importantes y los más utilizados dentro de nuestra aplicación. Al principio nos costó adaptarnos a su filosofía. Pero a medida que avanzamos en la implementación fuimos adquiriendo el conocimiento necesario. Como veremos casi todo está basado en “anotaciones”[16] esto es tan importante que hay capitulo entero mostrando y explicando las distintas anotaciones.

Componente de Negocio

La unidad básica de una aplicación Java es el Componente de Negocio y se define creando una Entidad, para crear una entidad simplemente se debe crear una clase Java, y añadirle la anotación @Entity, agregar las propiedades, los getter y setter. Luego de esto podemos hacer “Build all” y ya tenemos creado todo lo que necesitamos para usar esta clase en nuestra aplicación, todo la interfaz de usuario y el manejo de los datos. En la figuras siguiente se muestra el código escrito y la interfaz generada.

```java
package org.openxava.facturacion.modelo;

import javax.persistence.*;
import org.openxava.annotations.*;

@Entity  // Esto marca la clase Customer como una entidad
public class Cliente {

    @Id  // La propiedad numero es la clave.  Las claves son obligatorias (required) por defecto
    @Column(length=6)  // La longitud de columna se usa a nivel UI y a nivel DB
    private int numero;

    @Column(length=50) // La longitud de columna se usa a nivel UI y a nivel DB
    @Required  // Se mostrará un error de validación si la propiedad nombre se deja en blanco
    private String nombre;

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

}
```

Modelar

Vimos que crear un clase sencilla no tiene mayores complicaciones, pero qué pasa cuando queremos tener modelo más complejos para representar distintas realidades.
Si bien los manuales aparecen como que es muy fácil de representar las distintas relaciones como Na1 , 1aN simplemente se debe poner las anotaciones @ManyToOne , @OnetoMany respectivamente o si queremos representar que una entidad tiene una colección de otra se utiliza la anotación @ElementCollection .

```java
@ElementCollection
private Collection<Detalle> detalles;
```

Cuando queremos complejizar el modelo con múltiples relaciones de todo tipo como (Na1, 1aN, NaM, 1a1, subconjuntos, colecciones) entre todas las entidades empezaron a surgir distintos inconvenientes de compilación, o de ejecución o de datos. Somos conscientes que muchos de estos problemas se debieron a nuestra inexperiencia en el uso de la herramienta

y esto fue parte de nuestro aprendizaje, nuestras costumbres y el cambio de paradigma tal vez aquí fue donde más se notó, por ejemplo las tablas generadas no eran como las esperábamos. Pero por otro lado aparecen algunos errores propios del generador, las tareas de encontrarlos y resolverlos empezó a consumirnos tiempo que a priori no pensábamos invertir y algunos problemas al parecer pequeños llevaban mucho tiempo de resolución.

Estereotipos

Un estereotipo especifica y define un uso de un tipo dado. Por ejemplo almacenar string, porcentajes, dinero e imágenes entre mucho más, existen una gran variedad de Estereotipos para los usos más comunes y se pueden definir otros que necesitemos.
Básicamente se define como una “anotación” en la propiedad que queremos manejar el comportamiento y OpenXava hace el resto en la interfaz de usuario donde se utilice esa propiedad. Por ejemplo si queremos que una propiedad sea una imagen marcando la propiedad con @Stereotype(“IMAGES_GALLERY”) tenemos disponible toda una galería de imágenes, otra ventaja que se puede escribir la anotación en el idioma de preferencia a continuación mostramos el código para que una propiedad sea una Galería de Imagen.

```
@Stereotype("GALERIA_IMAGENES") // Una galería de fotos completa está disponible
@Column(length=32) // La cadena de 32 de longitud es para almacenar la clave de la galería
private String masFotos;
```

Como vemos esto es muy sencillo, creemos que es una gran virtud de OpenXava.

Vistas

En determinadas ocasiones queremos mostrar solo un subconjunto de las propiedades de una entidad, para ello existen las vistas simplemente se enumeran las propiedades a mostrar con la anotación @View y luego se debe hacer referencia con @ReferenceView

```
@View(name="Simple", // Esta vista solo se usará cuando se especifique “Simple”
    members="numero, nombre" // Muestra únicamente numero y nombre en la misma línea
)
public class Cliente {
```

```
@ManyToOne(fetch=FetchType.LAZY, optional=false)
@ReferenceView("Simple") // La vista llamada 'Simple' se usará para visualizar esta referencia
private Cliente cliente;
```

En nuestro sistema usamos una gran variedad de vistas, según la información que debíamos mostrar en las diferentes pantallas.

Validaciones

Las validaciones de datos se asemejan a funciones en otros lenguajes o procedimientos en Genexus. Primero definimos la validación con la “anotación” @EntityValidator junto con su nombre y sus propiedades y por lado definimos la clase con la lógica de validación deseada.

, entonces se crea una clase cada que vez que se cree o modifique la entidad, vemos con un ejemplo para ser más ilustrativos

```
@EntityValidator(
    value=ValidadorEntregadoParaEstarEnFactura.class, // Clase con la lógica de validación
    properties= {
        @PropertyValue(name="anyo"), // El contenido de estas propiedades
        @PropertyValue(name="numero"), // se mueve desde la entidad 'Pedido'
        @PropertyValue(name="factura"), // al validador antes de
        @PropertyValue(name="entregado") // ejecutar la validación
})
public class Pedido extends DocumentoComercial {
```

```
public class ValidadorEntregadoParaEstarEnFactura
    implements IValidator { // ha de implementar 'IValidator'
```

Es en este tipo de clases que se define la lógica de la aplicación, por ende es muy importante entenderlo y saber aplicarlo. Este es un mero ejemplo pero existen otras anotaciones para que se ejecuten en distintos momentos según las acciones que se están realizando por ejemplo: antes de modificar,al eliminar, también podemos extender el funcionamiento y crear nuestros propios validadores.

Problemas encontrados

Uno de los principales con que nos encontramos en el uso de OpenXava fue el no poder realizar testing automatizados, aquí explicamos las razones de ello.
OpenXava extiende las capacidades de JUnit para permitir probar un módulo exactamente de la misma forma que lo haría un usuario final. De hecho, OpenXava usa HtmlUnit, un software que simula un navegador real (incluyendo JavaScript) desde Java. Todo está disponible desde la clase de OpenXava ModuleTestBase, que permite automatizar las pruebas que un usuario haría a mano usando un navegador de verdad de una forma simple.
Al principio cuando todavía no estábamos utilizando el portal de Liferay creamos tests con esta herramienta y funciono correctamente. Al hacer la integración con Liferay, se generó una dependencia con la api de Liferay, la cual hizo que los tests no funcionaran correctamente. Intentamos corregirlos pero vimos que no era sencillo y rápido, también no encontramos documentación al respecto, por esta razón decidimos hacer tests funcionales del sistema en el portal de Liferay.

OpenXava, elección correcta

Luego de haber logrado cerrar una solución funcional con esta herramienta podemos hacer una autocrítica respecto a nuestra visión y expectativas iniciales de la misma versus la percepción actual. Si bien la herramienta resuelve de forma rápida y sencilla todo lo referente a los ABMs de las entidades simples (Docentes, Estudiantes, etc.) no es tan simple como se muestra en la documentación cuando las entidades incorporan cierta complejidad como puede ser relaciones 1:1, 1:N o M:N. Para estos casos hubo que realizar muchas pruebas de ensayo y error para lograr, por un lado la estructura de base de datos esperada y por otro que a nivel funcional el sistema se comportará según lo esperado. A esto también hubo que sumarle que la herramienta no es totalmente Open Source ya que hay prestaciones que sólo están incluidas en la versión paga de la misma (por ejemplo el manejo de seguridad, la generación para smart devices o por ejemplo la conectividad nativa contra un LDAP) lo que llevó a buscar soluciones alternativas para remediar estos faltantes en lo que refiere a funcionalidades ya que el pago de la licencia no era una opción. Como punto favorable la herramienta cuenta con mucha información a nivel de la web (foros, tutoriales, videos, etc.) que aportan mucho al momento de resolver los problemas pero como desventaja la mayoría están basados en el ejemplo clásico del sistema de facturación que no incorpora ejemplos de entidades con todos los tipos de relaciones (1:1, 1:N o M:N) sino más bien de las que funcionan sin ningún problema. Esto último generó que hubiera un mayor tiempo de dedicación en el aprendizaje del lenguaje para encontrarle la vuelta a problemas no documentados.

Fuera de lo que es el desarrollo clásico de los ABMs, tuvimos dificultades en generar una transición fluida entre las diferentes pantallas del sistema más precisamente en aquellos procesos que tenían una complejidad mayor como por ejemplo la acción de cumplir una agenda la cual disparaba procesos en paralelo dependiendo el tipo de agenda.

Lo importante de la elección es que, independientemente de las ventajas y desventajas, se pudo desarrollar la solución pedida con esta herramienta y la misma creemos cumple con las expectativas del cliente tanto en lo funcional como en la calidad de la misma.

Resumen de trabajo

La Clínica Notarial de la facultad de Derecho de la UdelaR cumplen una tarea importantísima brindado un servicio de asesoramiento Notarial a sectores vulnerables de nuestra sociedad que de otra manera no podrían acceder a él. Es por eso que construir una solución de software para facilitar las tareas de la Clínica fue una desafío y una responsabilidad, un desafío desde el punto de vista académico y una responsabilidad de aportar un trabajo para mejorar el funcionamiento de otra facultad y de ayudar con un granito de arena a nuestra sociedad.
Nosotros como aspirantes a Ingenieros construimos una solución utilizando muchísimas herramientas aprendidas a lo largo de la carrera. Durante el proyecto transitamos todos los caminos y cumplimos las distintas etapas para llevar a cabo el Desarrollo de software y así brindar una solución al problema planteado .
Esas etapas son:

- **Pre Análisis:** Reunirse con el cliente, estudio de viabilidad y alcance del sistema.
- **Análisis:** Definición de Requerimientos.
- **Diseño:** Arquitectura de software.
- **Implementación:** Codificación del software, resolución de problemas y documentación.
- **Test:** Pruebas sobre el sistema.
- **Implantación:** Instalación y entrenamiento.

Creemos que nuestra solución de software cumple con informatizar las gestión de la Clínica Notarial, y esperamos que este software sea un punto de partida para que la Clínica Notarial mejore su servicio y que un futuro se mejore y actualice nuestro sistema y se puedan añadir otras funcionalidades que hagan aún mejor a la Clínica Notarial

Conclusiones y trabajo a futuro

En este punto se presentan las conclusiones del trabajo realizado en nuestro proyecto y se mencionan algunos ítems que a nuestra consideración se deberán tener para ampliar el desarrollo de esta solución.

Conclusiones

Hicimos un relevamiento de las herramientas de software disponibles para el desarrollo de aplicaciones web y mobile, como resultado de esto hicimos algunas comparativas en el capítulo Herramientas para el Desarrollo , sección Comparación de Herramientas.
Este relevamiento nos dio un panorama de algunas de las varias herramientas existentes en el mercado, tanto para el desarrollo del sistema como para la gestión de todos los puntos del proyecto. Nos hizo ver la importancia de hacer un trabajo concienzudo nos da un respaldo a lo hora del desarrollo del proyecto, ya que con este análisis se mitigan riesgos tecnológicos y de recursos humanos en una etapa temprana del proyecto.
A partir del estudio de las herramientas y tecnologías disponibles elegimos OpenXava más Liferay, ya que a nuestro entender fue el conjunto más apropiado para cubrir cada objetivo del desarrollo que teníamos por delante.
La construcción del prototipo inicial nos ayudó a adquirir experiencia en el uso de las herramientas y a validar nuestro relevamiento tecnológico, esta experiencia nos fue de gran utilidad para las siguientes iteraciones del proyecto.
La generación de los documentos de usuario y de instalación le dan un soporte extra al sistema, ya que usuario nuevos, desarrolladores o administradores futuros podrán entender y utilizar rápidamente el sistema.
Como conclusión final tenemos un Sistema Clínica Notarial implantada y totalmente funcional en un servidor de la Facultad de Derecho[17] listo para utilizarse y avalada por el Cliente llámese Consultorio Jurídico de la Facultad de Derecho de la Universidad de la República.
La idea es el que el Sistema comience a funcionar fuertemente en Marzo de 2019, cuando comienza el año lectivo y finaliza la feria judicial.
Actualmente existen dos versiones:

- Una de test con datos simulados y acceso a LDAP ficticios que ayudan a los usuarios a realizar las prácticas que crean necesarias para llegar a la fecha señalada con al menos una mínima experiencia de uso, más allá de las pruebas que ya realizaron en los distintos prototipos que les fueron entregados.
- Una de Producción con datos reales y acceso a los LDAP tanto de Estudiantes como de Funcionarios Docentes y no Docentes (administrativos) .

Estamos muy contentos con haber logrado el objetivo principal y creemos firmemente que la solución implementada e implantada será de muchísima utilidad para la Clínica Notarial, para la Facultad de Derecho y por supuesto será un beneficio enorme para todo el conjunto de usuarios, Docentes, Funcionarios, Estudiantes, de todos ellos los más beneficiados sin lugar a dudas serán los usuarios de la Clínica Notarial que dispondrán de un sistema mucho más eficiente y de mejor calidad adaptado a los tiempos que corren para atender sus

distintas problemáticas notariales, recordemos que estos usuarios finales son personas de bajo recursos y de una clase socioeconómica Media/Baja, por todo esto que planteamos entendemos que el Sistema Clínica Notarial es un beneficio para toda la sociedad.

Trabajo a futuro

Requerimientos que quedaron fuera de la alcance

Creemos que lo primero que se debe hacer a futuro es incorporar los requerimientos que quedaron fuera del alcance. Principalmente se destacan dos, tener un sistema de notificaciones y alertas que den aviso a los usuarios del sistema cuales son los eventos y tareas que se deben realizar en un determinado día, o cuales actividades deben realizarse presencialmente por algún estudiante como por ejemplo asistir a un caso o presentar determinado documento en algún lugar.
También avisar a los usuarios de la Clínica cuando tienen sus citas, estos avisos podrían darse por sms, mail, whastapp o lo que se considere más apropiado.
Por otra lado sería de gran utilidad contar con paneles de gestión, informando distintas métricas de atención, como cantidad de casos atendidos en un día, mes o año. Cantidad de usuarios con casos abiertos o información de la cantidad de intervenciones de los estudiantes en los diferentes casos.
Es importante destacar que la información mencionada en los anteriores párrafos existe en el sistema y la misma es accesible por parte del usuario, lo que se busca es dar un paso hacia adelante en lo que refiere a la experiencia de usuario de manera que los actores involucrados estén al tanto de las actualizaciones en la información del sistema sin la necesidad que para esto se vean obligados a ingresar al sistema y buscarla por sus medios.

Integración con Consultorio Jurídico

Sería de gran utilidad tener integrado los dos sistemas, el de Clínica Notarial y el Consultorio Jurídico esta integración debería darse en varios planos, Por un lado desde la tecnología y por otro lado desde el lado de los datos.
Desde la tecnologia podrian compartir el mismo servidor, mismo sitio y las mismas herramientas de desarrollo.
Desde los datos es una opción más factible, sería de gran interés que los usuarios, los roles, los permisos estuvieran unificados ahorrando así problemas de incompatibilidad de datos. Tambien se podrian derivar casos de un sistema a otro, mejorando así la eficiencia de los sistemas y de la Clínica Notarial y el Consultorio Jurídico en general. En este sentido el sistema desarrollado incorpora, pero de modo pasivo, la entidad Derivación para que sirva a futuro como punto de entrada de las derivaciones del Consultorio Jurídico hacia la Clínica Notarial. Si bien resolver esta interacción requiere de un análisis más profundo y detallado, en su momento se trabajó en el mismo y se sostuvo que la forma más práctica para la comunicación sería el intercambio de información por intermedio de la publicación de Web Services, sean estos de tipo SOAP o REST.

Extender aplicación para dispositivos móviles

Tener una aplicación para dispositivos móviles para que estudiantes, docentes y funcionarios puedan realizar acciones, tareas en línea desde cualquier lugar, también recibir notificaciones y todas las funcionalidades que fueran necesarias.
Esto seria un gran avance, ya que se pasaría a un sistema totalmente en línea, en donde por ejemplo un Estudiante puede modificar un caso directamente desde el lugar donde se encuentre, un ente público o un juzgado. Esto daría una usabilidad mayor y una gran eficiencia. En su momento se planteó la posibilidad de realizar este agregado pero desde la Clínica Notarial se prefirió en esta primera etapa no realizar la apertura de ese canal. De todos modos estamos convencidos que, no sólo va a ser de gran beneficio sino que también muy necesario.

Digitalización y Reconocimiento de documentos

Actualmente todo el registro del Consultorio Jurídico se lleva a papel, sería ideal tener un sistema que pueda digitalizar esos documentos y cargarlos al sistema pero no solo como un archivo digital si no que ese nuevo sistema tenga la suficiente inteligencia como para incorporarlos y convertirlos en la entidades del Sistema Clínica Notarial, sería un especie de software de reconocimiento de imágenes que convierta archivos en papel a entidades de un Sistema.

Glosario

SCN : Sistema Clínica Notarial, software a desarrollar en el presente proyecto
LDAP: El Protocolo de Acceso Ligero a Directorio, mejor conocido como LDAP (por sus siglas en inglés) [18]
MySQL: es un sistema de gestión de base de datos relacional, multihilo y multiusuario.
Prototipo: Modelo del sistema que posee casi todas las características del sistema final y es representativo de éste, sirve para entender aspectos del sistema y clarificar los requerimientos
JDBC: [19] Java Database Connectivity (en español: Conectividad a bases de datos de Java)
FileSystem: Componente del sistema operativo encargado del manejo de archivos
Open source: es un modelo de desarrollo de código de software, basada en la colaboración abierta.
SVN: Es una herramienta de software open source que se utiliza para el control de versiones de archivos.
ABM: Abreviatura de Altas, bajas y modificaciones, se refiere a las funciones básicas en base de datos.
Portlet: Los portlets son componentes modulares de las interfaces de usuario gestionadas y visualizadas en un portal web.
ACID: En bases de datos se denomina ACID a las características de los parámetros que permiten clasificar las transacciones de los sistemas de gestión de bases de datos. Es un acrónimo de Atomicidad, Consistencia, Aislamiento y Durabilidad en español
SQL: (por sus siglas en inglés Structured Query Language; en español lenguaje de consulta estructurada) es un lenguaje específico del dominio utilizado en programación, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales
Mockups: son bosquejos del sistema, permiten con un diseño ágil mostrarle al cliente un prototipo en etapas tempranas de la construcción del sistema.
IoC de Spring: Inversión de Control por sus siglas en inglés, es un proceso en el que los objetos definen sus dependencias a través de los argumentos del constructor, o métodos setter que son invocados luego de la construcción del objeto.
MVC: Modelo vista controlador, es un patrón de arquitectura de software que separa la representación de una aplicación de los datos y lógica de negocio.
IDE: Acrónimo de Integrated Development Environment en inglés, Entorno de desarrollo integrado, es una herramienta de software que facilita al programador el desarrollado de software.
JDK: Acrónimo de Java Development Kit en inglés, Kit de desarrollo Java es un software que provee herramientas de desarrollo para la creación de programas en Java.
GAM: GeneXus Access Manager, es un módulo de seguridad que resuelve las funcionalidades de autenticación y autorización, tanto para aplicaciones Web como para aplicaciones para Smart Devices.
Environment: Entorno de desarrollo en GeneXus que consta de un lenguaje de generación, un motor de base de datos y una plataforma objetivo dados.

OWASP: Acrónimo de Open Web Application Security Project, en inglés. Es un proyecto abierto de seguridad de aplicaciones web dedicado a determinar y combatir las causas que hacen que el software sea inseguro.
OWASP Top 10: es un documento que lista los diez riesgos de seguridad más importantes en aplicaciones web según la organización OWASP.
AJAX: acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones.
JPA: es la API de persistencia desarrollada para la plataforma Java
API: Interfaz de programación para aplicaciones, API por sus siglas en inglés, es un conjunto de funciones, procedimientos y subrutinas que ofrece una biblioteca para ser utilizado por otro software.
Servlets: El servlet es un clase Java, comúnmente utilizada para ampliar capacidades de un servidor web.
MDD: El desarrollo impulsado por modelos, MDD por sus siglas en inglés, aprovecha los modelos gráficos y los componentes de aplicaciones pre-construidos para que los usuarios puedan construir visualmente aplicaciones complejas.
Junit: es un conjunto de bibliotecas que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java.

Anexos

1. Mockups - Proyecto de grado
2. Documento de casos de uso
3. Documento de Testing
4. Manual de usuario
5. Manual para Desarrolladores y Administradores

Bibliografía

1. https://www.postgresql.org
2. https://www.mysql.com
3. http://projects.spring.io/spring-ldap/
4. https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-testing.html
5. http://www.mybatis.org/spring-boot-starter/mybatis-spring-boot-autoconfigure/
6. http://projects.spring.io/spring-data/
7. http://projects.spring.io/spring-security/
8. http://openxava.org/es
9. https://www.liferay.com/es/home
10. https://balsamiq.com/products/
11. https://www.generatedata.com/
12. https://webtematica.com/plataforma-digital-liferay-diseno-y-desarrollo-web-con-este-cms
13. https://docs.microsoft.com/es-es/dotnet/core/index
14. https://proyectosagiles.org/desarrollo-iterativo-incremental/
15. https://nvie.com/posts/a-successful-git-branching-model/ por Vincent Driessen.
16. http://openxava.org/OpenXavaDoc/docs/annotations_es.html
17. https://itn.fder.edu.uy/
18. http://ldapman.org/articles/sp_intro.html
19. https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html