Lista de Abreviaturas

API	Application Programming Interface
DI	Demendency Inyection
ECMA	European Computer Manufacturers Asociation
GPL	General Public Licence
HTML	Hypertext Modeling Language
HTTP	Hypertext Transfer Protocol
IOC	Inversion Of Control
JPA	Java Persistence API
JS	JavaScript
JSON	JavaScript Object Notation
LAMP	Linux Apache MySQL PHP (Perl o Python)
MVC	Modelo Vista Controlador
POJO	Plain Old Java Object
POM	Project Object Model
SOA	Service Oriented Architecture
SQL	Structured Query Language
USB	Universal Serial Bus

Introducción

Capítulo 1

Entorno Empresarial

En el presente capítulo se describe el entorno en el cual se desarrolló el proyecto de pasantía HxPlus Ocupacional, el cual fue realizado para la empresa Globinsoft S.A. Se presenta la historia, descripción, estructura organizacional y el cargo ocupado por el pasante dentro de la misma.

1.1 Antecedentes

Globinsoft S.A. posee en la actualidad su producto HxPlus el cual tiene como objetivo el almacenamiento de historias médicas de manera digital, usando tecnología web, para mantener su disponibilidad a cualquier hora del día y desde cualquier dispositivo con acceso a la web. Cuenta también con la opción de generar informes médicos, récipes y reposos médicos para uso de los farmacéutas y comodidad de los pacientes.

1.2 Misión

Brindar apoyo tecnológico al área médica en Venezuela, prestando servicios de calidad dentro del marco de lo estipulado por el Ministerio de Salud.

1.3 Visión

Ofrecer una plataforma integral para la gestión de consultas, historias médicas, récipes y medicamentos con alcance nacional y disponibilidad las 24 horas del día, los 7 días de la semana.

1.4 Estructura organizacional

Globinsoft S.A. mantiene los siguientes departamentos:

1. Gerencia.
2. Recursos Humanos.
3. Finanzas y Contabilidad.
4. Proyectos.

En la gerencia de proyectos se ubica el ingeniero Juán Albarrán, quien a su vez tiene el papel de tutor industrial de la pasantía descrita en el presente informe. La gerencia de proyectos se divide en cada uno de los proyectos realizados por la empresa y hasta el momento de la presente redacción, se cuenta con "HxPlus" como proyecto en producción, al cual se le hace seguimiento, y soporte, y "HxPlus Ocupacional" como proyecto en desarrollo. Figura 1.1.

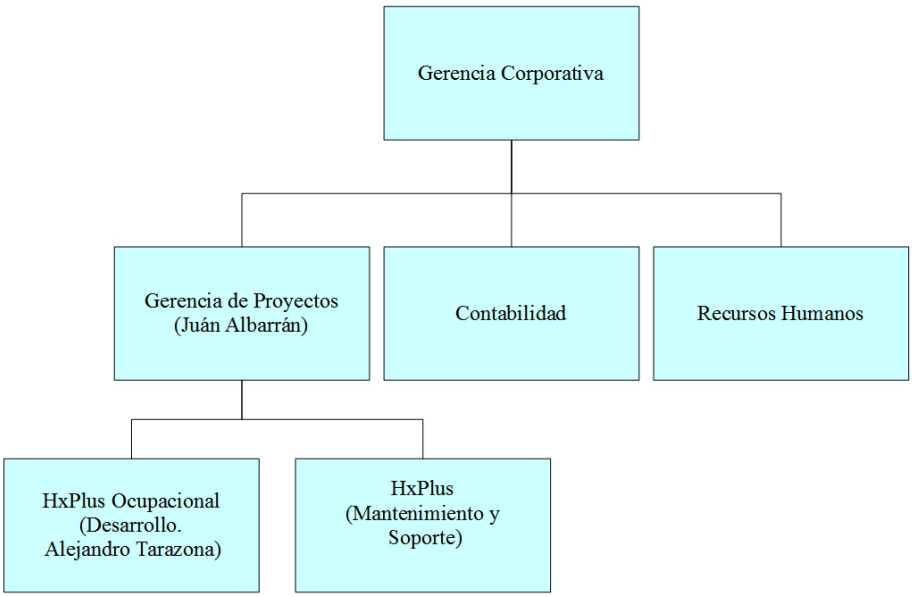

Figura 1.1: Estructura Organizacional de Globinsoft

Capítulo 2

Marco Teórico

En el presente capítulo se describen conceptos y patrones utilizados para el desarrollo del proyecto de pasantía, los cuales fueron seleccionados siguiendo los criterios de usabilidad, mantenibilidad, escalabilidad y portabilidad.

2.1 Modelo Vista Controlador (MVC)

El Modelo-Vista-Controlador es un patrón de arquitectura de *software* que separa el modelo (objetos del negocio), la vista (interfaz con el usuario u otros sistemas) y el controlador (manejo de la información del negocio)[1].

Técnicamente el usuario intereactúa con las vistas llenando formularios, solicitando información o simplemente haciendo algún *click* que genere una petición al sistema. En ese momento es el controlador quien recibe la petición y genera las acciones necesarias sobre el modelo para así acceder a los datos y generar la nueva vista, resultado de la petición realizada y enviarla al usuario. Tal y como muestra la figura 2.1.

Específicamente, cada componente tiene una asignación independiente de los demás componentes. Estas son:

1. **Modelo**
    - Almacenamiento de los datos.
    - Estado del sistema.
    - Recuperación de errores a nivel de datos.

Figura 2.1: Representación gráfica del patrón Modelo Vista Controlador[2]

2. **Vista**

   - Presentación del modelo.
   - Accede al modelo pero no puede cambiar su estado.

3. **Controlador**

   - Reacciona a las peticiones del cliente.
   - Comunica al modelo de las acciones ejecutadas.
   - Direcciona a las vistas requeridas del lado del cliente.

Esto se busca, primordialmente, para hacer del código altamente mantenible en el tiempo, ya que antiguamente se realizaban los sistemas siguiendo lo que se conoce como *"programación de espaguetti"* (programación no estructurada) la cual no llevaba una separación entre lo que era la vista y los procesos internos del sistema. Esto traía como consecuencia, en el momento de realizar algún cambio al sistema ya sea de formato o de interacción, que tuviera que modificarse integralmente vista, interacciones y (potencialmente) el modelo de datos.

2.2 Arquitectura Orientada a Servicios

Es una filosofía de desarrollo la cual actúa como una guía de diseño y facilita el mismo orientado a la escalabilidad del sistema, facilita las actualizaciones de uno o varios de los componentes MVC y minimiza el efecto que dicha actualización tiene sobre los demás.

Según Gartner[3] y Quiroga[4], la incorporación de SOA empieza en las empresas hacia 2003, por las siguiente razones:

- La incesante presión de los negocios para la agilidad. Cuando una empresa quiere modificar sus procesos, productos o servicios, no puede permitirse el lujo de esperar por mucho tiempo. Debe ser posible cambiar la forma de aplicación de los sistemas de trabajo simplemente modificando los componentes que ya están en uso, en lugar de comprar o codificar nuevos componentes o sistemas enteros desde cero.
- La flexibilidad de la arquitectura SOA basada en Servicios Web de apoyo a múltiples aplicaciones.
- La aceptación unánime de proveedores de los estándares de Servicios Web, especialmente de Simple Object Access Protocol (SOAP) y Web Service Description Language (WSDL)[3]

SOA se basa en capas, cada una ofrece servicios a la siguiente y sus procedimientos internos se mantienen ocultos a las demás capas. Con esto se generan APIs de acceso estandarizado que son independientes de las tecnologías utilizadas en el desarrollo.

En *Service Oriented Architecture*[5] definen SOA usando el poema de Saxe sobre seis ciegos y un elefante.

"Seis ciegos de Indostan se encuentran con un elefante, cada uno describe el elefante de forma diferente porque se ve influenciado por sus propias experiencias (Ver figura 2.2)

- Quien le toca la trompa cree que es una serpiente.
- Quien le toca los colmillos cree que son lanzas.
- Quien le toca las orejas cree que son abanicos.
- Quien le toca la panza cree que es una pared.
- Quien le toca la cola cree que es una cuerda.
- Quien le toca las patas cree que son árboles."

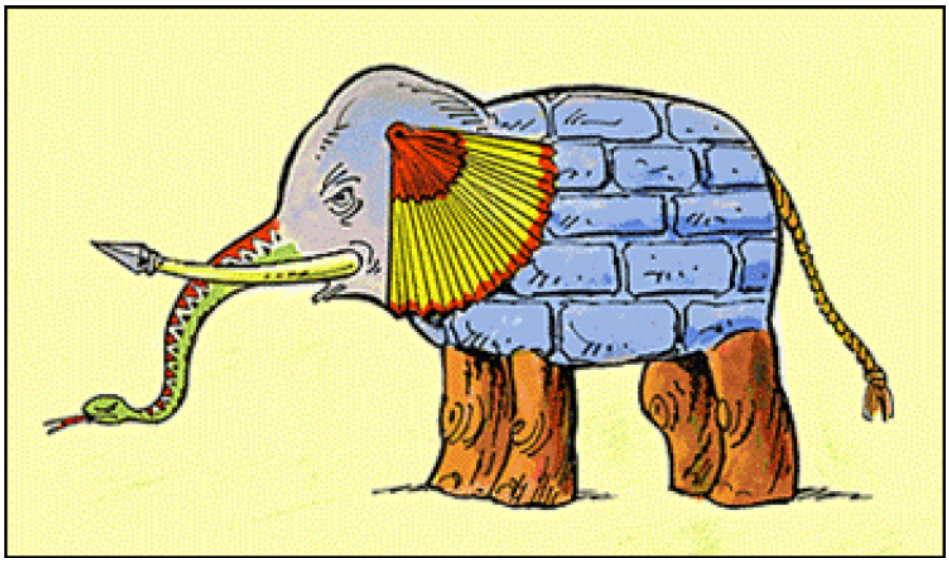

Figura 2.2: El elefante de Saxe

Se usa la analogía para ejemplificar el hecho de que haya varias definiciones diferentes de lo que es SOA en si, porque se le ha definido como patrón de diseño o como una filosofía de desarrollo, siendo esta última la definición adoptada para el trabajo descrito en el presente informe.

También se puede usar dicha analogía para ejemplificar cómo los (potenciamente diversos) dispositivos pueden interactuar con el controlador y este a su vez con el modelo sin que ello represente un cambio fundamental en diseño o estructura de los mismos. Cada dispositivo interactúa con los servicios que necesita y más ningún otro.

2.3 Autenticación Basada en *Tokens*

Autenticación, según se lee en [6], es la "acción y efecto de autentificar". Esto es comprobar ante una autoridad la veracidad o legitimidad de un documento o un hecho[7].

En sistemas, el término es utilizado para definir el proceso de verificación de las credenciales de usuarios dentro de un sistema. Comunmente usando el par "nombre de usuario" y "contraseña" para ello.

Existen diversos métodos para llevar a cabo la autenticación[8]:

1. Sistemas basados en algo conocido, ya sea *password* o *passphrase*.

2. Sistemas basados en algo poseído que puede ser un tarjeta de identidad o dispositivos USB.

3. Sistemas basados en características físicas como voz, huellas dactilares o patrones oculares.

La autenticación basada en *tokens* es una forma de autenticación ligera que va de la mano con SOA, usa *tokens* (o fichas) cifrados para la verificación de usuarios. Estas fichas son almacenadas en el cliente y enviadas en cada una de los *requests* que realiza el navegador, la ficha es descifrada y se verifican las credenciales del usuario en cuestión[9].

Entre los datos almacenados en las fichas suelen estar:

- Nombre de usuario.
- Fecha de autenticación.
- Fecha de caducidad de la ficha.

Estas son cifradas en el servidor bajo una clave secreta elegida por el mismo servidor y que se usa para el posterior descifrado de las fichas.

Aunque no es un determinante, es deseable que la ficha contenga suficiente información del usuario para poder indentificarlo en cada una de las solicitudes y además sea lo suficientemente ligera como para no afectar la carga de datos. Todo esto permite que el servidor no se sobrecargue con variables de estado o de sesión por cada usuario autenticado en un momento dado y además permite la portabilidad desesada para el sistema.

En la figura 2.3 se especifican los pasos básicos de la autenticación basada en *tokens*:

1. El cliente hace una solicitud de autenticación enviando en la solicitud el nombre de usuario (*username*) y su contraseña (*password*).
2. El servidor, una vez procesadas y confirmadas las credenciales del usuario, envía el *token* cifrado de regreso para que sea almacenado en el cliente.
3. El cliente envía una solicitud de lo que sería una página de "inicio" dentro del sistema, adjuntando entre los encabezados (*headers*) de dicha solicitud, el *token* almacenado.
4. El servidor, en caso de confirmar el *token*, responde con la página y acciones solicitadas.

En adelante, y hasta que se complete el cierre de sesión, todas las solicitudes realizadas al servidor deben llevar adjunto el *token* asignado para ser revisado en el servidor.

En caso de fallos de autenticación, ya sea por credenciales incorrectas o un *token* inválido, el servidor responderá con un mensaje de error y redireccionará la navegación a la vista de "inicio de sesión" u otra página por defecto.

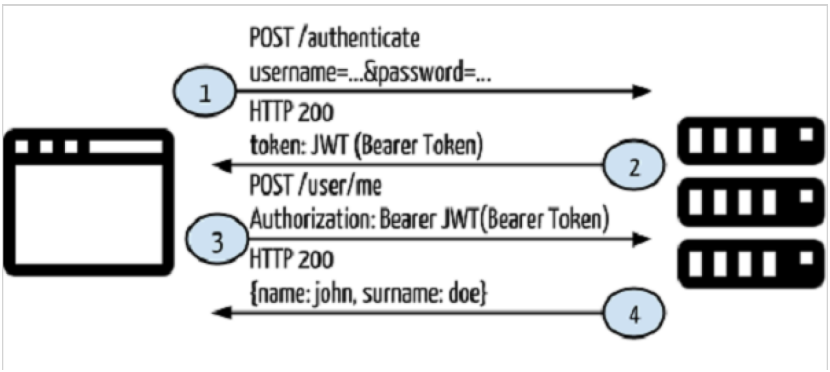

Figura 2.3: Autenticacion basada en fichas.

2.4 Inversión de Control

Concepto utilizado por primera vez por Martin Fowler[10] que define una diferenciación principal entre un *framework* y una biblioteca, siendo la biblioteca un conjunto de clases y funciones que pueden ser llamadas por el programa desarrollado y el *framework* un diseño que controla el flujo y eventualmente llamará al código desarrollado para cumplir con las funciones específicas. De esta forma, el control de flujo no lo tiene el código si no el *framework*, que responde a las solicitudes de los usuarios y solicita al programa las acciones requeridas.

En este sentido, Spring cumple la cunción de manejar las solicitudes que surgen del *front end* e invoca las secciones de código designadas como "controladores" las cuales dependen (ver 2.5) de "servicios".

2.5 Inyección de Dependencias

Es una forma de IOC, se basa en que una clase A "inyecte" objetos en otra clase B, siendo B dependiente de A, en lugar de permitir que la clase B se encargue de crear los objetos en sí misma[11, 10].

De esta forma se pueden crear objetos de una clase B y pasar el objeto de la clase A como argumento en el constructor de B o en algún *setter* asignado al atributo. En adelante, cualquier función o procedimiento que precise información de A puede ser invocado a través de B aunque la responsabilidad de ejecución recaen en el objeto A.

De forma más abstracta, se puede utilizar lo que en Java se conoce como *interface* para dejar mayor margen a las modificaciones que puedan surgir con el tiempo en el código o en los requerimientos.

En el caso de Spring, se usa inyección de dependencias en varios niveles:

1. Los “controladores” utilizan a los “servicios”, que son interfaces creadas para realizar el procesamiento interno de los datos que llegan al servidor. Estas interfaces tienen procedimientos predeterminados, como toda “interface”, sus implementaciones deben implementar individualmente todos estos procedimientos y son inyectados (los “servicios”) en sus respectivos “controladores”.

2. Los “servicios” a su vez hace uso de los diferentes “repositorios”. También son interfaces que están estipuladas por JPA e implementadas por Hibernate (ver puntos 4.1.6 y 4.1.12) que son inyectados en los “servicios” que así lo requieran.

Capítulo 3

Marco Metodológico

En el presente capítulo se describe la metodología usada para el desarrollo del proyecto así como sus componentes, los actores que participaron y los roles que cumplieron en el desarrollo del mismo.

Para *HxPlus Ocupacional* fue seleccionado Scrum como metodología de desarrollo a seguir. Y aunque no está estipulado por la metodología, se llevaron a cabo diagramas de casos de uso y de clases para una documentación completa del sistema.

Scrum es una metodología de gestión de proyectos ágil que utiliza uno o más equipos de trabajo, de a lo sumo 7 personas, en iteraciones de tiempo fijo, llamados *Sprints*, para la entrega de tareas o avances en el proyecto que sean funcionales y probados [1].

Los equipos apuntan siempre a conseguir avances limpios, probados y aceptados de manera que puedan ser puestos en producción inmediatamente.

Scrum se divide en Roles, Eventos y Artefactos tal como se describe a continuación:

3.1 Roles

3.1.1 Product Owner

Este rol representa la voz del cliente en la gestión del proyecto. Debe velar por la realización del proyecto desde la perspectiva del negocio. Se encarga de escribir las historias de usuario, las prioriza y las anexa al "Product Balcklog" (o también Lista del Producto).

[1] Con información de Jeff Sutherland[12], Pete Deemer[13] y proyectosagiles.org[14]

El Dueño de Producto es la única persona responsable de gestionar la Lista del Producto. La gestión de dicha lista, según expresa Jeff Sutherland[12], consiste en:

- Expresar claramente los elementos de la Lista del Producto.
- Ordenar los elementos en la Lista del Producto para alcanzar los objetivos y misiones de la mejor manera posible.
- Optimizar el valor del trabajo desempeñado por el Equipo de Desarrollo.
- Asegurar que la Lista del Producto es visible, transparente y clara para todos, y que muestra aquello en lo que el equipo trabajará a continuación.
- Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario.

Para el presente proyecto se designó al ingeniero Juan Albarrán como Product Owner siendo el mejor representante de los intereses de Soluciones Globinsoft S.A. y estando él familiarizado tanto con los porcesos de negocio como con el equipo de desarrollo y el proceso de desarrollo mismo.

3.1.2 Scrum Master

Es el responsable de llevar el procedimiento de gestión del proyecto según los lineamientos de Scrum. Sirve de asesoría entre invloucrados y comprometidos en materia de organización y distribución de la información. Él dirige las reuniones y vela por su cumplimiento según las reglas de procedimiento de Scrum.

Jeff Sutherland[12] clasifica los servicios del Scrum Master en tres categorías:

1. Servicio al dueño del producto
    - Asesoramiento para la gestión eficiente de la Lista del Producto.
    - Asesoramiento para la planificación del producto.
2. Servicio al equipo de desarrollo
    - Apoyar en la organización del equipo.
    - Eliminar los impedimientos y dificultades que limiten el progreso del equipo.
    - Facilitar la organización de los eventos de Scrum según sea requerido.
    - Guiar al equipo en entornos donde Scrum no haya sido del todo adoptado.

- Apoyar al equipo en dudas que surjan respecto a los eventos y artefactos de Scrum.

3. Servicio a la organización

   - Liderar la adopción de Scrum como metodología de desarrollo.
   - Asesorar a los empleados en el uso de Scrum y sus procedimientos.

Por su experiencia en previos proyectos, entre ellos *HxPlus* (antecedente de *HxPlus Ocupacional*) y su amplio conocimiento en el negocio, se le delegó también el puesto de "Scrum Master" en el presente proyecto al ingeniero Juan Albarrán.

3.1.3 Development Team

Es el conjunto de personas dedicadas a construir el producto indicado por el dueño del producto. Se requieren que sean grupos lo suficientemente pequeños como para fomentar un alto nivel de independencia y autoorganización pero, a su vez, lo suficientemente grandes como para poder presentar avances significativos en cada Sprint y que potencialmente, estos avances, puedan ser puestos en producción.

Equipos pequeños, de menos de tres personas, reduce la productividad global y reduce los posibles avances en el producto. También podrían presentar limitaciones en cuando a las habilidades de los integrantes, lo cual resultaría convirtiéndose en impedimentos en el desarrollo.

Por otro lado, equipos muy grandes, de más de nueve miembros requiere demasiada coordinación entre los miembros lo cual podría significar una menor eficiciencia y mayor trabajo para la organización de los avances. En otras palabras, se perdería agilidad tratando de organizar un equipo semejante.

Idealmente un equipo de desarrollo varía entre tres y siete personas. Esto mantiene un equilibrio saludable entre la cantidad de trabajo que pueden manejar y la capacidad de autoorganización del equipo.

El equipo debe ser multifuncional, debe tener las capacidades necesarias para desarrollar el producto y poder apoyarse entre sí para compensar los puntos débiles de cada individuo.

No existen roles dentro del equipo de desarrollo. A pesar que uno podría desempeñarse mejor en un área de desarrollo, no existen como tal roles. Esto es debido a que la responsabilidad del desarrollo recae sobre el equipo como un todo y sobre ningún miembro en particular, por ello tampoco existe un rol de Líder o Jefe de Proyecto.

Para HxPlus Ocupacional el equipo de desarrollo consta de un solo miembro, Alejandro Tarazona, pasante y autor del presente libro.

3.2 Eventos

Esta sección describe los eventos determinados por la metodología seleccionada y cómo fueron establecidos para la gestión del proyecto. Estos son:

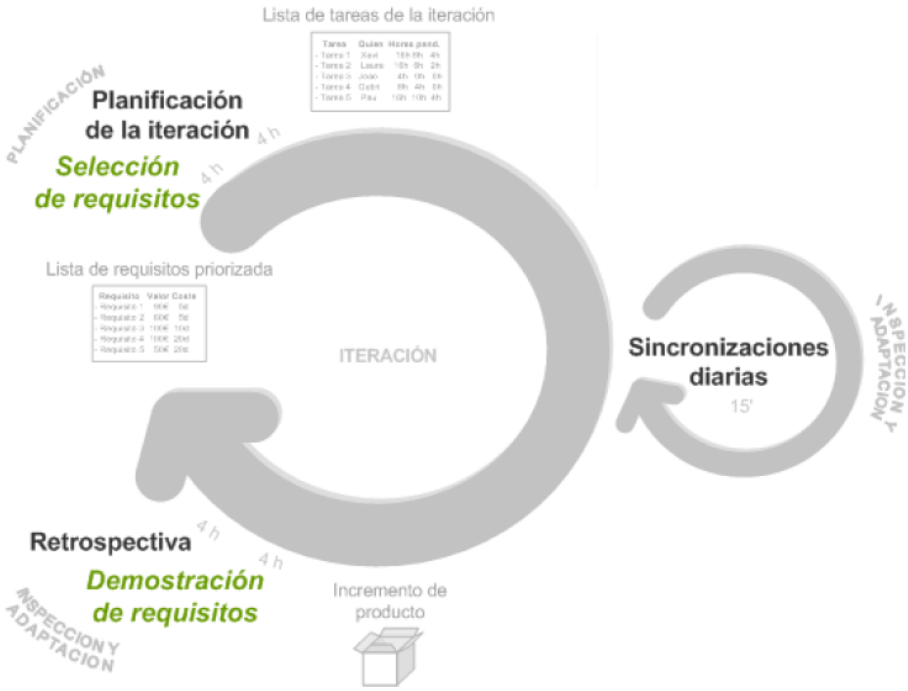

Figura 3.1: Esquema de trabajo de SCRUM

3.2.1 Sprint

Unidad mínima de desarrollo, usualmente determinada por una tarea corta o un período de tiempo pequeño, usualmente 1 o 2 semanas, nunca más de 30 días; durante el cual el equipo de desarrollo trabaja según las metas estipuladas al principio del mismo. Normalmente éstas metas no cambian durante el desarrollo del Sprint sino al final del mismo, cuando se planifica el siguiente Sprint.

Para HxPlus Ocupacional fue determinado en una semana para las primeras tareas y dos para las últimas, debido a las pruebas subyacentes y el trabajo de integración que representan.

Se llevaron a cabo 3 fases:

1. Preparación (1 Sprint)
2. Desarrollo (8 Sprints)

3. Cierre (2 Sprint)

Tal y como se describe en el capítulo 5.

3.2.2 Sprint Planning

Planificación del siguiente Sprint a realizar. El equipo de desarrollo ("Development Team", punto 3.1.3) hace los pronósticos e indica qué puede llevar a cabo en el siguiente Sprint de lo que propone el dueño del producto ("Product Owner", punto 3.1.1) que debe hacerse durante el Sprint. El "Scrum Master" (punto 3.1.2) está encargado de revisar cuidadosamente con el equipo las posibilidades, negociar con el dueño del producto las posibilidades de realización y establecer las metas.

Semanalmente se realizó una reunión del *Development Team* con el *Scrum Master* y *Product Owner* para evaluar el resultado del Sprint de esa semana y realizar la planificación adecuada a los logros y el desenvolvimiento en el proyecto.

3.2.3 Daily Sprint Meeting

Reuniones diarias realizadas entre el "Scrum Master" y el equipo de desarrollo para revisar los avances diarios, aclarar dudas y difundir información acerca del progreso alcanzado hasta el momento. Tiempo fijo en 15 minutos y usualmente se realizan en la mañana.

En este punto, Scrum, hace una diferenciación clave entre los elementos que están compromentidos (*"commited"*) y los que sólo están involucrados (*"involved"*) en el desarrollo del proyecto. Siendo comprometidos los equipos de desarrollo, el "Scrum Master" y el "Product Owner" e involucrados los demás departamentos de la empresa que puedan tener interés en el estado del proyecto (dpto. de ventas, clientes, etc).

En este órden de ideas, durante una reunión diaria de Sprint, sólo los comprometidos tienen potestad de hablar o comentar las cosas que han sucedido. Esto se hace para lograr que, en los 15 minutos de duración de la reunión, se discutan temas que sean de suma necesidad para el desarrollo del proyecto, se ponen de manifiesto dificultades técnicas o impedimentos dentro de los equipos de desarollo y, también, para difundir información sobre el estado del proyecto a las partes involucradas.

La responsabilidad de resolver todo impedimento manifestado en dichas reuniones recae sobre el "Scrum Master".

Durante el proyecto se realizaron las reuniones con la presencia del Ing. Juan Jesús Albarrán para actualizar el estado del desarrollo del proyecto y aclaración de dudas por parte del pasante en cuanto

a lo acaecido durante el día previo.

3.2.4 Sprint Review

Son reuniones que se realizan, como su nombre lo indica, para hacer una revisión del trabajo realizado durante el Sprint y presentar el trabajo completado a las partes involucradas. Estas reuniones no deben pasar de 4 horas de duración y todo trabajo incompleto no debe ser presentado.

Una vez mostrados los resultados del Sprint, el "Product Owner" debe realizar una evaluación de las metas cumplidas (o no) y si ha habido cambios en el contexto, deberá también realizar las adaptaciones necesarias a la planificación del proyecto.

En *HxPlus Ocupacional* se realizaoron en conjunto las reuniones de *Sprint Planning* y *Sprint Review* del último Sprint finalizado con la finalidad de minimizar el tiempo de reuniones y aprovechar las disponibilidades de los comprometidos y los involucrados.

3.2.5 Sprint Retrospective

Al finalizar cada Sprint el equipo se reúne con un tiempo fijo de 4 horas para revisar sus técnicas y la forma en que han abordado el desarrollo del proyecto, discutir las impresiones referentes al Sprint superado y revisar los inconvenientes presentados.

Es deber del "Scrum Manager" revisar los inconvenientes y buscarles solución rápida para mejorar la productividad del equipo.

Debido al que el grupo de trabajo sólo consta de 1 desarrollador, se consideró inconveniente realizar reuniones de 4 horas exclusivamente para hacer la restrospectiva. En su lugar se atendieron los inconvenientes, dudas y revisones durante las reuniones diarias y se apartó un espacio de 15 minutos de las reuniones de planificación y revisión para realizar actividades de ésta reunión.

3.3 Artefactos

Documentos realizados para llevar registro de las etapas de desarrollo del proyecto y a su vez realizar la evaluación de las mismas.

3.3.1 Product Backlog

O también “Lista de Producto”. Es una lista con todas las consideraciones necesarias de parte del dueño del producto, quien la organiza y la gestiona. Cualquier cambio a realizarse dentro de la planificación debe pasar por esta lista.

En esta lista se enumeran los deseos del cliente, se priorizan y se estima el esfuerzo requerido. Esta lista debe ser seguida por el equipo de desarrollo para dirigir sus avances.

Es una lista que hace el Dueño del Pruducto iniciando el proceso de gestión de requerimientos, sin embargo, esta lista tiene la característica de ser mutable, como los requerimientos del Dueño del Producto, y es modificada conforme sea necesario o requerido. Por eso es que Jeff Sutherland[12] dice: “Una Lista de Producto nunca está completa”.

Para ello existe el “refinamiento” de la lista del producto, el cual es el proceso de añadir detalles, granularidad y prioridad a cada uno de los requerimientos, según Jeff Sutherland[12]. Usualmente, las tareas o requerimientos que pasan a ser parte de la siguiente planificación de Sprint son las de mayor prioridad y granularidad y que, además, suele ser el caso que las actividades prioritarias son refinadas primero para así llevarlas a desarrollo lo antes posible.

En el caso de *HxPlus Ocupacional* el Dueño del producto realizó un levantamiento de requerimientos y utilizó “Trello” para la gestión de los requerimientos.

3.3.2 Sprint Backlog

O “Lista de Pendientes del Sprint”. Es una lista de objetivos del Sprint tomada de la lista de producto durante la planificación del Sprint. Puede ser uno o varios objetivos, lo suficientemente refinados como para que el equipo de desarrollo pueda entenderlos en la reunión diaria y puedan ser llevados a cabo durante el Sprint.

Según se requiera nuevo trabajo, el equipo de desarrollo lo irá añadiendo a la lista de pendientes del Sprint, ya sea por inconvenientes surgidos o problemas no tomados en cuenta o por refinamiento de los objetivos. Además, conforme el trabajo vaya siendo completado, se debe actualizar la estimación del trabajo restante. Sólo el equipo de desarrollo tiene potestad sobre la lista de pendientes del Sprint y es su forma de ver, transparentemente y en tiempo real, el estado del dearrollo de un Sprint.

Usando también las facilidades de “Trello”, el equipo de desarrollo gestionó cada Sprint a través de las listas creadas dentro de una “Pizarra” del sistema.

Capítulo 4

Marco Tecnológico

En este capítulo se presentarán las herramientas y protocolos utilizados durante el desarrollo del proyecto, ya sea para el desarrollo en sí mismo o para el apoyo en cuanto a control de versiones y gestión de tareas.

4.1 Herramientas para el desarrollo de la aplicación

En esta sección se describen las herramientas usadas en el desarrollo de la aplicación y las características que hicieron que fueran seleccionadas para tal fin.

4.1.1 Eclipse

Es un entorno integrado de desarrollo (IDE) basado en Java. Provee las librerías necesarias para el desarrollo, facilita la configuración del proyecto y hace uso de herramientas como Maven para la gestión de librerías del proyecto.

Combina un compilador junto con facilidades para la configuración de diferentes servidores, tanto de bases de datos como servidores web para atender los servicios del *back end*.

Usando esta herramienta se procedió a la configuración de los repositorios de Maven (ver 4.1.11), el servidor tomcat (ver 4.1.3), el entorno de desarrollo de Java y su respectivo entorno de ejecución (ver 4.1.4).

En el desarrollo de "HxPlus Ocupacional" se utilizó *Eclipse Kepler* en su versión de 64 bits para Linux - Debian 9.

4.1.2 apache

Es un servidor web multiplataforma. Utiliza el protocolo http para la transferencia de información con los clientes y posee soporte de seguridad para SSLy TLS[15]. Posee licencia GPL y una comunidad de desarrolladores que mantiene el servidor actualizado continuamente.

Dadas sus características *open source* se cuenta con equipos de desarrolladores alrededor del mundo que además funjen como soporte del mismo, lo cual lo hace un servidor altamente utilizado y con muy buena capacidad de respuesta en caso de inconvenientes con el mismo.

Está alojado dentro del servidor de "Apache Foundation"[16] en donde además se alojan otras herramientas utilizadas en el proyecto.

Para "HxPlus Ocupacional" se designó Apache como servidor web exclusivo de *front end*. Se hace incapié en ello ya que dada la orientación a servicios del sistema, se pueden crear nuevos servidores, para dispositivos móbiles primordialmente, que están dedicados exclusivamente a su función y no cambiarían la forma de interacción de este servidor.

4.1.3 Tomcat

Apache Tomcat, Jakarta Tomcat o comunmente llamado Tomcat es un servidor web especializado en el almacenamiento de sistemas web bajo las especificaciones JSP (JavaServer Pages) de Oracle Corporation, aunque fue creado, originalmente, por Sun Microsystems.

Siendo parte de Apache Foundation, también posee licencia GPL y es de código abierto, con una comunidad que desarrolla, bajo "Java Comunity Process", las especificaciones de Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket[17].

Este servidor fue instalado y configurado para hospedar el *back end* de "HxPlus Ocupacional" y los servicios que ofrece. Tomando la filosofía de SOA (Ver 2.2), en este servidor estarán alojados los servicios que proporcionará el sistema, para ser usados por las distintas implementaciones del *front en.*

4.1.4 Java

Lenguaje de programación utilizado para el desarrollo del *backend*. Java es un lenguaje de programación imperativo orientado a objetos que facilita el desarrollo independiente de los servicios y el fácil acceso al modelo de datos permitiendo así la baja cohesión entre los componentes del sistema.

El entorno de desarrollo (JDK) y de ejecución (JRE) fue Java 8. Ya que porvee las últimas actualiza-

ciones de las librerías de Java y previene problemas de seguridad por puertas traseras presentados en la versión 7. Además que ofrece facilidades adicionales en lo que se refiere a la utilización de herramientas como Hibernate, JPA e iText.

4.1.5 JSON

Por *JavaScript Object Notation*, es un formato de intercambio de datos, ligero[18] que permite una sencilla comunicación entre la vista y el controlador. También permite la fácil depuración y la visualización de los datos enviados, dentro del entorno de desarrollo, para la verificación y corrección de errores.

JSON es un formato de texto que es independiente del lenguaje. Utiliza convenciones que son conocidos por los programadores de Java, JavaScript, Perl, Python, y otros. Estas propiedades hacen que JSON sea el lenguaje ideal para el intercambio de datos[19].

JSON se contruye de la siguiente forma:

- Todo objeto atómico comienza y termina con "{}" (llaves).
- Los objetos llevan nombre del objeto seguido de ":" y el valor del objeto.
- Si el valor del objeto es compuesto (varios atributos), los componentes se enlistan entre llaves y separados por "," (coma).
- Los atributos de un objeto son siempre un par *nombre:valor*. Siendo "nombre" el nombre del atributo. Es similar a la sintaxis usada para el nombre del objeto.
- Listas o arreglos de objetos son nombrados como un objeto atómico mas una "s" al final del nombre.
- Los arreglos van enmarcados de "[]" (corchetes).

Fue elegido por su compatibilidad con Java y porque es independiente de la tecnología usada en la vista, lo cual permite a su vez realizar cambios en la vista sin afectar las funcionalidades del controlador.

4.1.6 JPA

Por "Java Persistence API", proporciona un modelo de persistencia basado en objetos de Java planos (POJO, por sus siglas en inglés *Plain Old Java Object*) para hacer la correspondencia con las

```
{"employees":[
    {"firstName":"John", "lastName":"Doe"},
    {"firstName":"Anna", "lastName":"Smith"},
    {"firstName":"Peter", "lastName":"Jones"}
]}
```

Figura 4.1: Ejemplo de lista de empleados genérica en *JSON*

entidades en la base de datos[20]. En la práctica, hace transparentes las consultas y acciones realizadas sobre la base de datos.

Como tal, JPA, no implementa los modelos de persistencia que usará la base de datos si no que proporciona un estándar para que se puedan mantener las caractarísticas de la orientación a objetos de Java y se puedan enlazar, uno a uno, con las entidades, atributos y relaciones de la base de datos.

JPA se puede configurar vía anotaciones o usando un documento XML que debe ser distribuido junto con el sistema. En el caso de "HxPlus Ocupacional" se eligió la configuración por anotaciones debido que está presente directamente en los objetos (clases) de Java y permite una mejor mantinibilidad del código.

Entre las implementaciones conocidas de JPA tenemos:

- Hibernate
- ObjectDB
- EclipseLink
- OpenJPA

Siendo la implementación de Hibernate la seleccionada por lo descrito en el punto 4.1.12.

4.1.7 Angular JS

Es un *framework*orientada a facilitar el desarrollo web de aplicaciones dinámicas del lado del cilente. "AngularJS le permite extender el vocabulario HTML para su aplicación"[21]. Utiliza lo que llama "directivas" que son bloques de código en javascript que ayudan a estructurar las acciones del *front-end*. También maneja "atributos" y "elementos" que pueden ser programados separadamente del código HTML y luego insertados en dicho archivo para su utilización.

Provee asociación birireccional de variables del DOM lo cual simplifica drásticamente las pruebas del lado del cliente y mantiene, como se mencionó anteriormente, la estructura organizada del código. La asociación bidireccional a través de "expresiones" se utiliza para mantener actualizado al cliente en cuanto a cambios que se realizen en las variables internas y mejora la respuesta visual sin realizar una recarga de la página.

El código de Javascript (punto 4.1.8) debe ser importado en el archivo HTML en que se quieren utilizar. Existen dos formas de importarlas, desde el servidor de google[1] o descargando los archivos al servidor local y agregando la dirección local.

Por todo esto, se eligió AngularJS para su utilización como *framework* del *front end* del sistema.

4.1.8 JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico[22, 23]. También posee soporte en casi todos los navegadores utilizados actualmente.

Es un lenguaje de programación orientado a crear contenidos dinámicos para páginas web del lado del cliente. AngularJS, previamente mencionado, usa JavaScript como lenguaje de programación para su implementación.

4.1.9 MySQL

Manejador de bases de datos, posee licencia GPL y licencia comercial de Oracle[24]. Ofrece alto rendimiento, eficiencia y seguridad en el almacenamiento y recuperación de datos[25]. Comúnmente utilizado en entornos de desarrollo LAMP para desarrollo web.

Es un manejador multi-hilo, multi-usuario y robusto, está diseñado para soportar altos niveles de carga y ser utilizado en entornos de producción con fuerte afluencia de datos. Además de poseer librerías ampliamente usadas para Java las cuales ayudan al desarrollo debilmente acoplado del *back-end*.

Para "HxPlus Ocupacional" se eliegió este manejador, haciendo uso de su licencia GPL, para el desarrollo del sistema.

[1] http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js

4.1.10 SPRING

Framework para el desarrollo de aplicaciones que provee inversión de control; es de código abierto y está diseñado sobre Java. Permite integración con Hibernate, JPA y JSON[26].

SPRING fue diseñado para facilitar el desacoplamiento de los compomentes del sistema utilizanco IOC. Esto permite que los componentes sean desarrollados una y sólo una vez y que puedan ser reutilizados en diferentes contextos[27].

Su fácil integración con Hibernate y, por consecuencia, con JPA permite que la interacción con la base de datos sea transparente al desarrollador y evita que tenga que reescribirse el código en caso de cambios en el manejador (de bases de datos) utilizados.

4.1.11 Maven

Es una herramienta de gestión y manejo de librerías, parecida a "Apache Ant". Utiliza el concepto del "Modelo del Objeto de Proyecto" (del inglés *Project Object Model*, o POM) para gestiónar la construcción del proyecto dónde se utilice. Esto es, gestiona las librerías, dependencias y versiones (de las librerías) de forma centralizada y limpia.

El POM es un archivo en formato XML dónde se registran las librerías que serán usadas por el proyecto para que el manejador de Maven se encargue de la descarga de las mismas. Se gestiona a través de artefactos que registran la información de una librería de acuerdo con la figura 4.2.

Figura 4.2: Artefacto de Maven y descripción de su contenido.

La estructura del POM puede llegar a ser tan compleja como el proyecto que gestiona, llegando incluso a depender de otros POM. En "HxPlus Ocupacional" se manejó usando un sólo POM de la

manera más sencilla posible.

En *The Apache Software Foundation*[16] mantienen repositorios de librerías actualizados y correctamente asociados a las dependencias de dichas librerías.

4.1.12 Hibernate

Es un framework de persistencia que provee una implementación de JPA[28]. Gestiona las comunicaciones a nivel de nombres de entidades y atributos con su respectiva contraparte de Java, clases con sus atributos. A esto se le conoce como "Correlación Objeto-Relacional" (ORM, por su siglas en inglés *Object Relational Mapping*).

Esta correlación que crea Hibernate permite ciertas facilidades al momento de manejar las distintas interacciones con la base de datos que devienen en un código mantenible en el tiempo.

El framework es integrado al desarrollo a través de Maven y tiene soporte para los siguientes manejadores de bases de datos[29]:

- MySQL
- PostgreSQL
- Oracle
- DB2/NT
- HSQL Database Engine
- entre otros...

Tradicionalmente se utilizan dos archivos de configuración (llamados "hibernate.properties" y "hibernate.cgf.xml") que se modifican con cada nueva tabla o "clase de persistencia" que se requiera, sin embargo este método tiende a no ser mantenible en el tiempo ya que debe buscarse dentro de estos archivos las configuraciones de las clases y no existe un orden estipulado para la creación y modificación de estos archivos. Para evitar esto, y dado que se tiene una versión de Java opsterior a JDK 5, se procedió a la utilización de la versión 3 de hibernate que incorpora librerías para el uso de anotaciones, quedando así la configuración de las clases de persistencia dentro del código de las clases de Java, lo cual permite y permitió, durante el desarrollo, fácil depuración de errores y un sencillo mantenimiento del código.

4.1.13 iText

Herramienta de generación de PDF dinámicos[30]. Proporciona un API que permite la generación de archivos PDF usando la información enviada.

Actualmente existen módulos de gestión, interpretación y conversión de archivos PDF, sin embargo para efectos de "HxPlus Ocupacional" sólo se utilizó el módulo de generación de los mismos.

Su integración con Maven permitió la fácil descarga de la librería y su configuración dentro del proyecto. Para su uso sólo se necesitó la creación de una clase dentro del *back end*, que contiene las importaciones requeridas para así poder generar los PDF necesarios.

4.2 Herramientas para el control de versiones y planificación

En la presente sección se presentan las herramientas de planificación y control de versiones usadas durante el desarrollo de la aplicación. Si bien no están vinculadas directamente al código, las mismas sirvieron de soporte para la organización del proyecto.

4.2.1 Git

Es un sistema de control de versiones distribuido, diseñado por Linus Torvalds, enfocado en eficiencia, integridad de datos, velocidad de transferencia y soporte para flujos de trabajo no lineales.

Siguiendo este enfoque, se logró un *software* que almacena los archivos relativos a un proyecto tomando como base un directorio "raiz" y los archivos y directorios que lo conformen, incluye recursivamente los archivos y directorios dentro del directorio raiz, y a partir de ellos almacena los cambios realizados a la última versión, sin modificar los archivos originales ni los archivos de modificaciones; con ello se logra que los cambios puedan ser reversibles y que sean almacenados con poco uso de memoria, reduce la carga de datos al momento de crear repositorios remotos ya que no se está enviando los archivos completos sino los archivos de cambios.

Para el caso de repositorios remotos y el manejo de la carga, también incluye la restricción de versiones. Un usuario debe tener en el repositorio local la última versión disponible, en caso de que la última versión esté en el repositorio local, se cargan los cambios normalmente al repositorio remoto; en caso contrario, el usuario debe descargar la última versión del repositorio remoto, (potencialmente) resolver conflictos que puedan surgir entre los archivos modificados de manera local y los que hayan sido modificados en el servidor remoto y luego hacer la carga al servidor remoto.

También ofrece la posibilidad de crear "ramas" de desarrollo. Esto es, cambios y modificaciones de un proyecto que parten de una raiz pero que puede tener una meta diferente. Esto facilita el proceso cuando existen varios desarrolladores trabajando en paralelo sobre el mismo proyecto, aunque en el caso de surgir conflictos en los cambios puede llegar a ser engorrosa la integración (*merge*) del código.

Para HxPlus se crearon dos repositorios raices, dadas implicaciones de permisos de ejecución dentro del sistema operativo elegido. Estos repositorios fueron llamasdos "occupational" y "proyectoAngular" refiriendose al *back-end* y *front-end* respectivamente.

4.2.2 GitHub

Servidores online de repositorios remotos para Git. Puede ser usado de manera gratuita y pública. Permite el acceso a los repositorios de manera ininterrumpida y global.

Los repositorios fueron almacenados en la cuenta personal de Alejandro Tarazona, en el url: *http://www.github.com/atara* con los nombres descritos con anterioridad, para su almacenamiento en la web.

4.2.3 Trello

Herramienta diseñada con la misión de facilitar la gestión de tareas usando listas o tablas. Cuenta con una interfaz intuitiva y de fácil aprendizaje para llevar a cabo dicha misión. Una vez creadas las tablas que se desean, se pueden crear tareas dentro de ellas, las cuales a su vez pueden ser etiquetadas, organizadas o comentadas por los participantes. Las tareas pueden ser arrastradas entre las listas emulando así la transición entre los estados de desarrollo del proyecto.

Capítulo 5

Desarrollo de la Aplicación

En el presente capítulo se presenta, de forma detallada, cómo fue el desarrollo del proyecto, los objetivos y actividades específicas de cada *Sprint* y finaliza con una revisión de dificultades técnicas y los resultados del proyecto.

5.1 Fase de Preparación

5.1.1 Primer Sprint

En este *sprint* se realizó las descargas de herramientas y la creación de los diagramas que serán la guía de desarrollo del sistema.

1. Objetivos
   (a) Levantar los requerimientos del producto.
   (b) Documentación de la aplicación.
   (c) Introducción al ambiente de trabajo.
2. Actividades
   - Creación del *Product Backlog*
   - Documentar la aplicación.
   - Creación del diagrama ER-E para la base de datos de la aplicación.
   - Creación de un modelo de casos de uso de la aplicación.
   - Descarga de las herramientas ya mencionadas en el marco tecnológico.

5.2 Fase de Desarrollo

5.2.1 Primer Sprint: Configuración de la aplicación

En este *sprint* se configuraron Eclipse, Apache y MySQL para crear un ambiente de desarrollo adecuado para el sistema a desarrollar.

1. Objetivos
   (a) Crear y configurar el ambiente de desarrollo
   (b) Descargar y configurar las herramientas necesarias para el desarrollo
   (c) Descargar y configurar las herramientas necesarias para la gestión del proyecto
2. Actividades
   - Creación de cuenta y tablas en Trello para la gestión del proyecto

     Las tablas de notas usadas fueron "Pendiente", "Impedido", "En desarrollo" y "Tarminado"; para referirse al estado actual de las tareas en cada una de las listas.
   - Creación de repositorios

     Se crearon los repositorios Git tanto local como remoto, para el almacenamiento de la aplicación usando Git y Github como sitio de almacenamiento para los repositorios remotos. Los repositorios están bajo los nombres de "ocupacional" y "proyectoAngular" en el url: *www.github.com/atarazona89*.
   - Diseño de la Base de Datos

     Creación del diagrama ER-E y el glosario de términos de la aplicación. Ver apéndice **??**.
   - Descarga del *IDE* Eclipse

     Para el desarrollo se utilizo Eclipse Kepler, descargado de la página oficial de Eclipse[31]
   - Descargar y configurar *plugins* de Eclipse

     Se realizaron los cambios necesarios en la configuración del "pom.xml" para así acceder a los repositorios de Maven que contienen las distintas librerías tanto de SPRING como de las distintas herramientas ya mencionadas. Se realizó la depuración de los archivos y la respectiva configuración adecuada al ambiente de trabajo. Esto se refiere a la configurar Eclipse para usar *SPRING* como *framework*, Hibernate y Liquibase como gestores de comunicación con la base de datos y configurar la base de datos con las credenciales de MySQL asignadas para el desarrollo del proyecto.

     Aunque, en lo referente a liquibase, se utilizó un plugin, no una librería. Ver cuadro 5.1.

Grupo	Artefacto	Versión
org.springframework	spring-core	4.1.2.RELEASE
org.springframework	spring-orm	4.1.2.RELEASE
org.springframework	spring-webmvc	4.1.2.RELEASE
mysql	mysql-connector-java	5.1.9
org.liquibase	liquibase-plugin	1.6.1.0

Cuadro 5.1: Artefactos de Maven: Spring

- Configuración de las características de *SPRING* para trabajar con anotaciones

  Se realizaron los cambios dentro del archivo "occupational-servlet.xml" que permiten el uso de anotaciones para el direccio amiento interno de los procesos.

- Descargar y configurar la librería de JSON para la comunicación con el *front-end*

  Usando los repositorios de Maven se descargaron las librerías necesarias de JSON para la comunicación. Las dependencias descargadas fueron, del grupo *com.fasterxml.jackson.core*, las enumeradas en el cuadro 5.2.

Artefacto	Versión
jackson-core	2.2.2
jackson-annotations	2.2.2
jackson-databind	2.2.2

Cuadro 5.2: Artefactos de Maven: JSON

- Descargar y configurar las librerías de Hibernate para la comunicación con la base de datos

  Usando Maven fueron descargadas las librerías de Hibernate que usan JPA como API para comunicarse con la base de datos.

Grupo	Artefacto	Versión
org.hibernate.javax.persistence	hibernate-jpa-2.0-api	1.0.1.Final
org.hibernate.common	hibernate-commons-annotations	4.0.4.Final
javax.persistence	persistence-api	1.0.2
org.hibernate	hibernate-entitymanager	4.1.9.Final
org.springframework.data	spring-data-jpa	1.8.1.RELEASE

Cuadro 5.3: Artefactos de Maven: Hibernate

5.2.2 Segundo Sprint: Configuración de la aplicación

En este *sprint* se relizaron las configuraciones necesarias de las herramientas y la creación de funcionalidades básicas del lado del servidor para empezar el desarrollo del sistema.

1. Objetivos

   (a) Creación de clases, repositorios y servicios básicos.

   (b) Creación de controladores básicos.

2. Actividades

   - Crear las siguientes clases para manejo de la información:
     - Usuario
     - Paciente
     - Doctor
     - Empresa
     - Centro de Costos
     - Sede
     - Departamento
     - Consulta
     - Nota de revisión (*SoapNote*, por *Subjective, Objective, Assessment, Plan*)
     - Diagnóstico
     - Examen
     - Consulta
     - Récipe
     - Medicamento
     - Laboratorio
   - Crear los repositorios

     En este entorno 'repositorios' se refiere a las interfaces que usa *Hibernate* para manejar los accesos y consultas con la base de datos. Estos repositorios vienen con métodos y procedimientos implemantados por Hibernate y estipulados por JPA.

     Se creó un paquete de clases, llamado "repositories", con un repositorio para cada una de las clases mencionadas anteriormente.
   - Crear los servicios para las clases

Los servicios están compuestos por una interfaz y su respectiva implementación por cada una de las clases mencionadas. Estos servicios se encargan de procesar la información haciendo uso de los repositorios, en caso de ser necesario, para brindar respuestas encapsuladas a los respectivos controladores.

- Crear los controladores de la aplicación

  En este *Sprint* se crearon los controladores con operaciones básicas de gestión (crear, listar, consultar, modificar y eliminar) de cada una de las clases, dejando para futuros *Sprint* la tarea de modificar los mismos las tareas específicas, en caso de ser necesario.

5.2.3 Tercer Sprint: Autenticación y gestión de Usuarios

En este *sprint* se implementó el módulo de autenticación y las funcionalidades básicas de gestión de usuarios, dejando para el siguiente *sprint* la parte estética de dicha gestión.

1. Objetivos

   (a) Implementación de la autenticación basada en tokens.

   (b) Creación, consulta, edición y eliminación de usuarios.

2. Actividades

   - Se implementó el módulo de autenticación siguiedo los parámetros de autenticación basada en *tokens* La clave usada por el servidor fue una clave generada en tiempo de ejecución para que la misma fuera cambiante y mejorar la seguridad. Sin embargo, la clave, una vez generada se mantiene igual mientras el servidor esté en funcionamiento. Ver figura 5.1.

     Para ello se agregó al "pom.xml" las dependencias requeridas para la autenticación. Ver cuadro 5.4

Grupo	Artefacto	Versión
io.jsonwebtoken	jjwt	0.5.1

Cuadro 5.4: Artefactos de Maven: Autenticación

   - Se implementó el módulo de gestión de usuarios, en el mismo, un usuario, con las credenciales adecuadas, puede crear, usuarios nuevos y se despliega una lista de los usuarios en la cual se puede elegir uno para luego poder editarlo o eliminarlo definitivamente de la base de datos. Esta última acción no es reversible por lo cual queda a consideración del *Product Owner* si quedará finalmente la funcionalidad al alcance de los usuarios. Ver figuras 5.2 y 5.3.

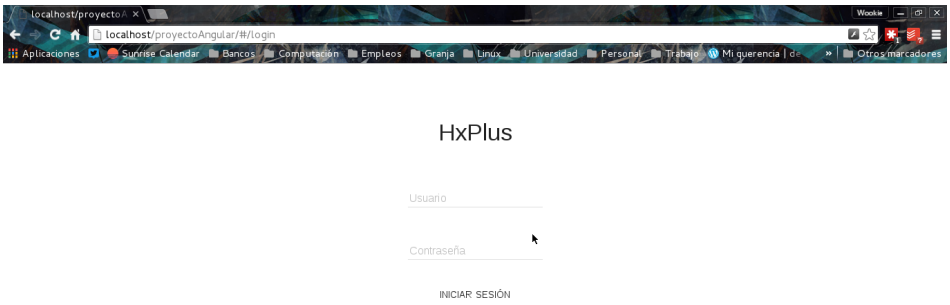

Figura 5.1: Pantalla de Autenticación de usuarios

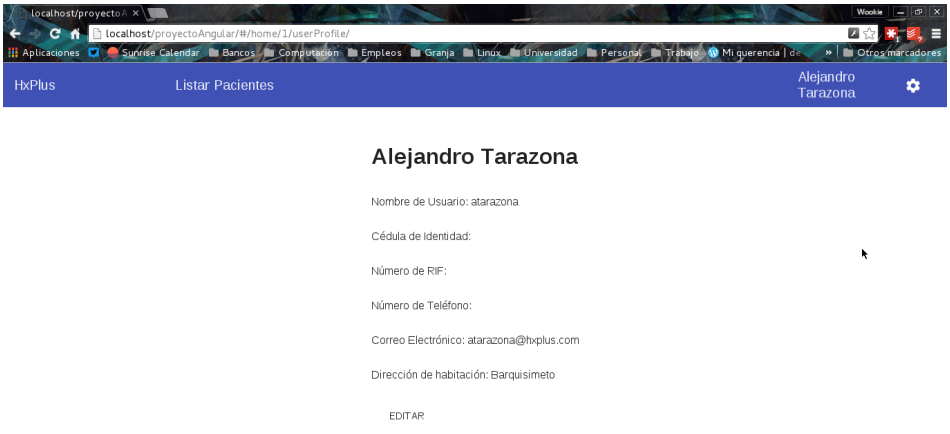

Figura 5.2: Pantalla de Revisión de Usuario.

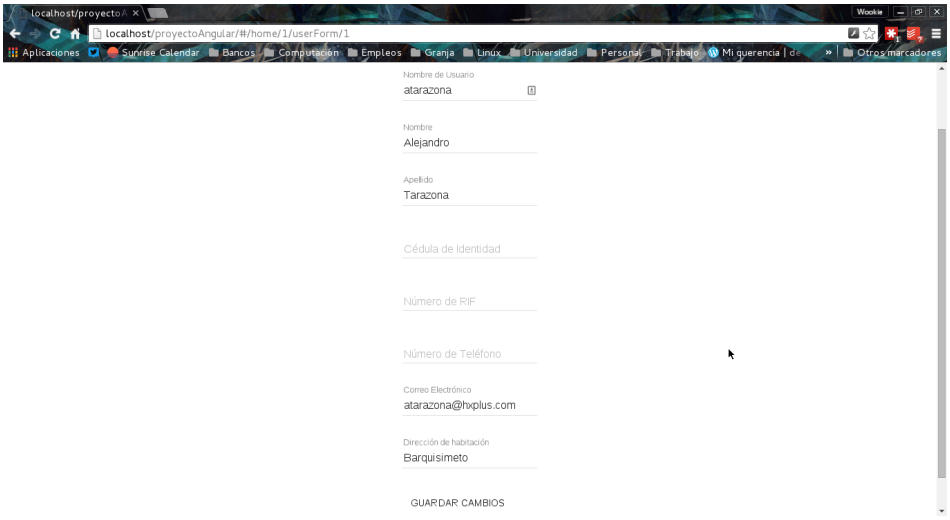

Figura 5.3: Pantalla de Edición de Usuario.

5.2.4 Cuarto Sprint: Consultas de Usuarios y Vista del doctor

En este *sprint* se mejoró la estética de la lista de usuarios y se implementó la visualización de parte de los doctores.

1. Objetivos

   - Consultas del módulo de usuarios.
   - Desarrollo de la vista del doctor.

2. Actividades

   - Ordenamiento de las listas de usuarios

     La lista de usuarios implementada en el *sprint* anterior fue reordenada para aparecer alfabéticamente en pantalla. Las funcionalidades previas no fueron modificadas.

   - Visualización de usuarios

     La lista también fue modificada para que fuese visualizada con apariencia de lista de usuarios siguiendo los lineamientos de Material Design. Ver figura 5.4.

   - Creación de la vista del doctor

     Esta incluye la lista de pacientes a atender en el día actual y la lista de pacientes que ha atendido el doctor. Junto con las funcionalidades de:

     – Agregar Paciente: Agrega un paciente nuevo a la lista de doctor, ya sea tomado de la lista de pacientes (Tal como se muestra en la figura 5.5) o creando un nuevo historial y agregándolo inmediatamente a la lista de pacientes del doctor (Figura 5.6).

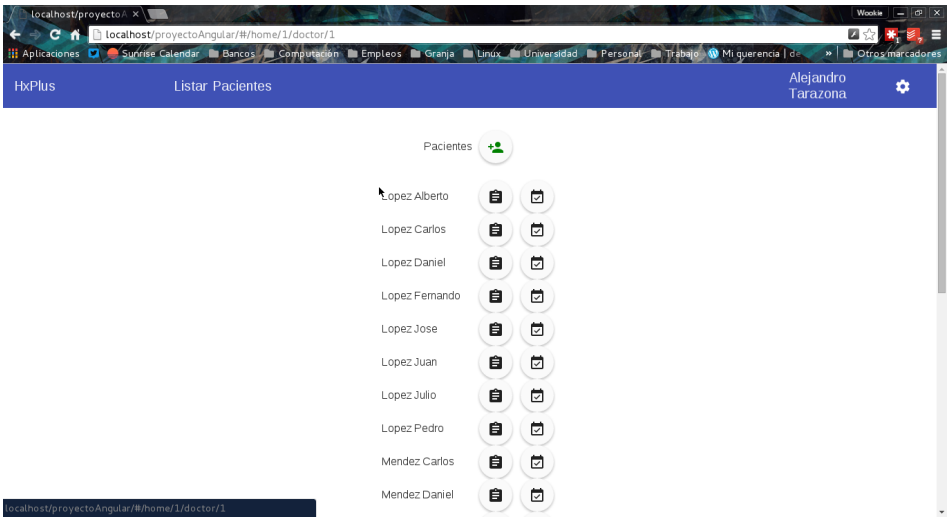

Figura 5.4: Lista de Pacientes del Médico

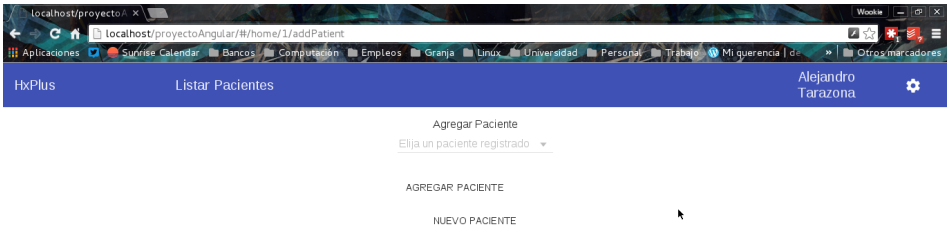

Figura 5.5: Vista de Agregar pacientes que ya han sido atendidos por algún médico.

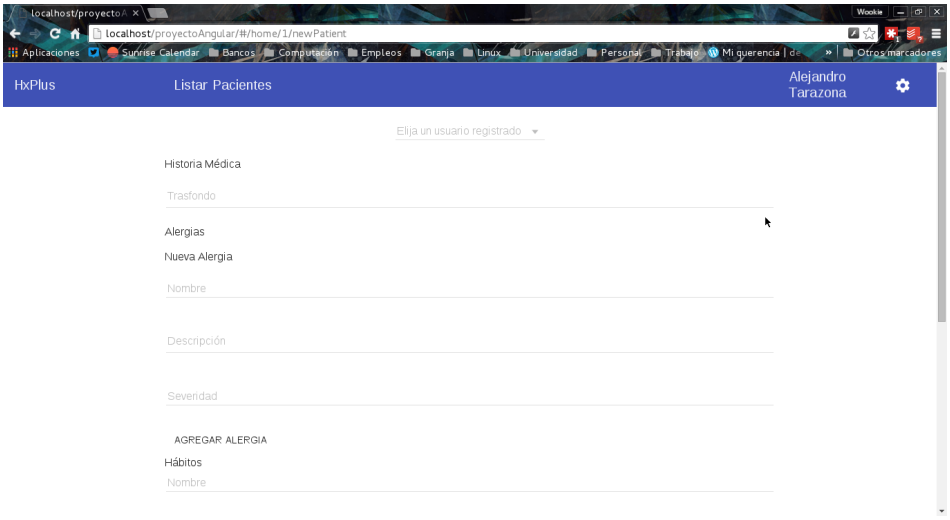

Figura 5.6: Vista de creación de nueva historia médica para un paciente nuevo.

– Crear Consulta: Crea una nueva consulta al historial de paciente. En este *sprint* se realizó la funcionalidad de "programar" una confulta futura, que permitirá la planificación de consultas. Se dejó para un futuro *sprint* la ceración de la consulta en sí misma ya que amerita el manejo de otros tipos de datos adicionales.

No se implementó la funcionalidad de "Eliminar Paciente" ni la de "Modificar Paciente" dado que el sistema busca mantener un registro histórico de las consultas y un médico no debería poder modificar ni eliminar dicho registro, sólo agregar nueva información al mismo.

5.2.5 Quinto Sprint: Consulta del Paciente

En este *sprint* se implementó el módulo de consultas del paciente, en este módulo el doctor puede crear y almacenar los datos pertinentes a una consulta dada y los mismos serán agregados al historial médico del paciente.

1. Objetivos

   - Generación de consultas.
   - Gestión de la linea de tiempo de la consulta del paciente.

2. Actividades

   - Crear consulta nueva

     Se crea la nueva consulta, permitiendo a médico almacenar los datos pertinentes. La consulta creada se agrega inmediatamente al historial del paciente, anexando los diagnósticos, en caso

de haberlos, a su apartado exclusivo dentro del historial.

Se almacenan los signos vitales o parámetros fisiológicos pertinentes, peso, estatura, tensión arterial entre otros. Se deja libertad al médico para añadir nuevos parámetros a los que ya estén almacenados en la base de datos (Figura 5.7).

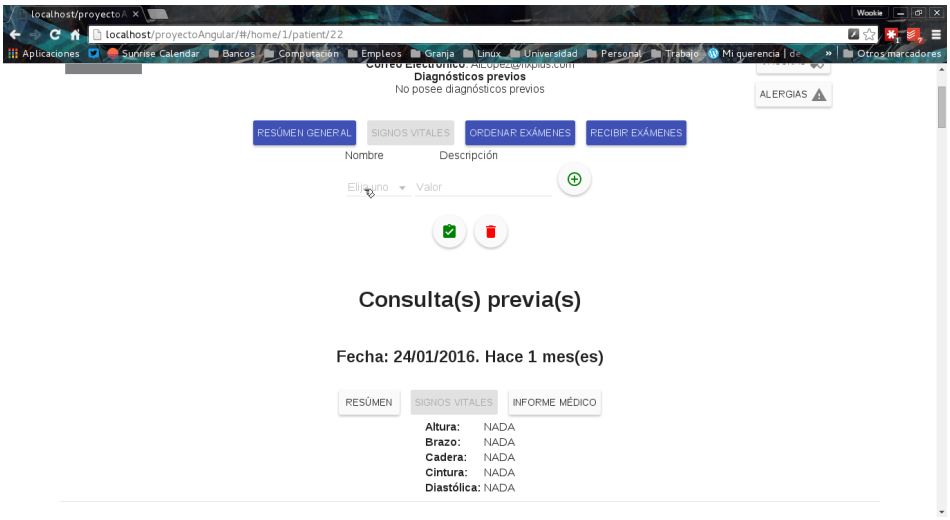

Figura 5.7: *Paarámetros Fisiológicos*

- Creación de *Soap Note*

  Se crea y almacena, para cada consulta, los datos recogidos en la consulta (Figura 5.8). Tales son:

  – Subjetivos (*Sbjective*): Lo que el paciente atestigua, sintomas y comentarios hechos por el paciente en el lenguaje que el mismo paciente los exprese para así mantener la información sin cambios en caso de que sea necesario revisarlos.

  – Objetivos (*Objective*): Lo que el médico puede observar en la consulta. Se usa en esta sección el lenguaje técnico propio de la medicina.

  – Comentarios (*Assessment*): Algún comentario u obsevación adicional que pueda hacer el doctor.

  – Plan: Se acordó que el *Plan* estaría en un apartado especial por cuestiones de simplicidad y acceso a los datos.

- Solicitud y recepción de exámenes médicos

  El médico tratante tiene uns sección de solicitud de exámenes médicos. En ella indica el nombre del exámen a solicitarle al paciente (Figura 5.9) Estas solicitudes quedan almacenadas y en futuras consultas, el médico tiene la opción de indicar la recepción de algún exámen previamente solicitado, ya sea uno o varios de ellos (Figura 5.10).

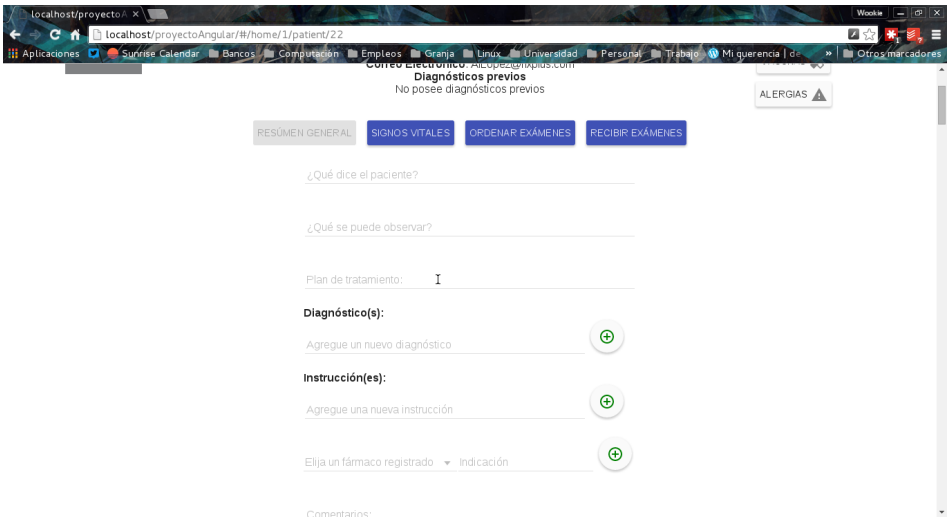

Figura 5.8: *Soap Note*

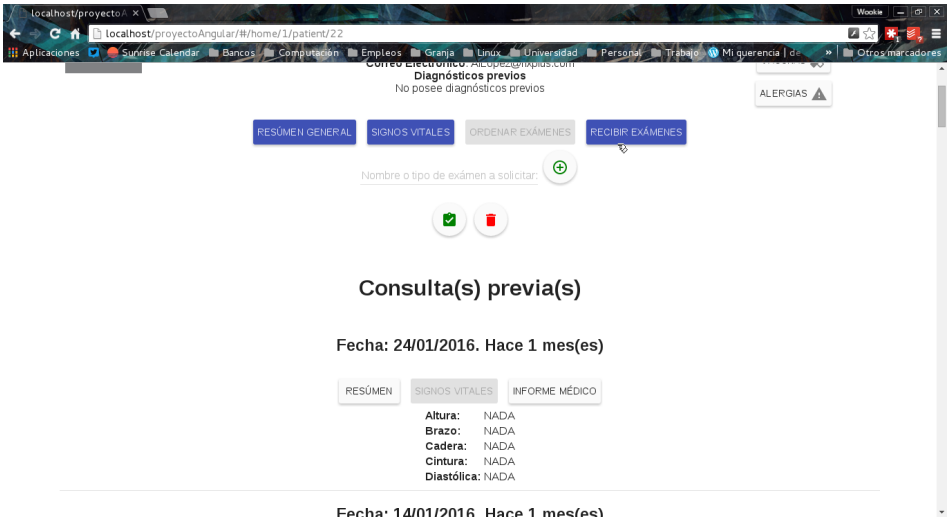

Figura 5.9: *Solicitud de Exámenes*

Figura 5.10: *Recepción de Exámenes*

En la sección de recepción de exámenes, el médico también tiene la opción de adjuntar algún archivo, ya sea resultados (en el caso de exámens sanguíneos) o imágenes (ecografías, rayos X, etc) los cuales serán anexados a la respectiva consulta y con ella al historial del paciente.

- Creación de diagnósticos

  El médico también puede almacenar uno o varios diagnósticos por consulta, estos constan de un nombre de enfermedad o dolencia y una gravedad. Ello lo elige de listas desplegables que estarán a su disposición tomando la base de datos del sistema. También se le permite agregar un nuevo diagnóstico en caso de carecer de ello el sistema, y el mismo será almacenado con posibilidad de reutilización (sólo del nombre) en futuras consultas.

- Creación del plan de tratamiento

  El médico puede crear un plan de tratamiento, el cual consiste en:

  - Recomendaciones: Cambios o hábitos de los que el paciente deba cuidarse, así como recomendaciones alimenticias y reposos. Es una sección informal.
  - Comentarios: Algun comentario extra, preguntas o anotaciones que el médico quiera hacer del conocimiento del paciente.
  - Tratamiento: Este incluye al menos un medicamento y la posología elegida para cada uno de los medicamentoa añadidos. Los medicamentos son elegidos de la base de datos del sistema. El administrador del sistema se encararía de añadir o eliminar medicamentos a la base de datos por ahora.
  - Récipe Médico: En este se indica un medicamento y su respectiva dosis para permitir al paciente su compra en la farmacia de su preferencia.

Esta información estará disponible, en un futuro *Sprint* para ser impresa en hojas, según el formato lo establezca, y ser entregada al paciente.

5.2.6 Sexto Sprint: Generar informes

1. Objetivos
   - Generación de informes médicos y reposos.
2. Actividades
   - Investigar las herramientas disponibles para generar archivos PDF.
   - Evaluar la compatibilidad con las herramientas y el *framework* que se está usando en el dessarrollo de *HxPlus Ocupacional*.
   - Elegir una de las herramientas y realizar los cambios adecuados para su uso en el sistema. Para *HxPlus Ocupacional* se eligión iText por las razones mencionadas anteriormente.

5.2.7 Septimo Sprint: Generar informes

1. Objetivos
   - Vista de Solicitud de informes.
2. Actividades
   - Se implantó la vista de solicitud de informes médicos con posibilidad de selección múltiple. Los parámetros son:
     - Examen físico
     - Diagnóstico(s)
     - Tratamiento
     - Comentario(s)

     Tal como se muestra en la figura 5.11. Esto permite que en un solo formulario, el médico tiene la posibilidad de generar tanto informes como reposos médicos.

5.2.8 Octavo Sprint: Generar informes

1. Objetivos
   - Generacion de informes.

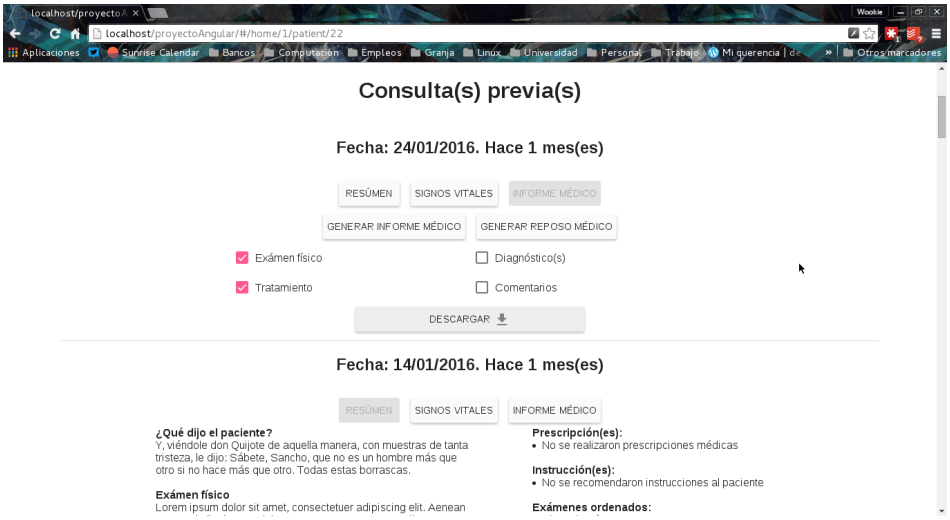

Figura 5.11: Sección de solicitud de informes médicos.

2. Actividades

- Se implantó el módulo de generación de informes médicos, tal y como fue descrito anteriormente, usando los parámetros selecciónados se generó el informe respectivo usando el formato indicado. Ver figura 5.12.

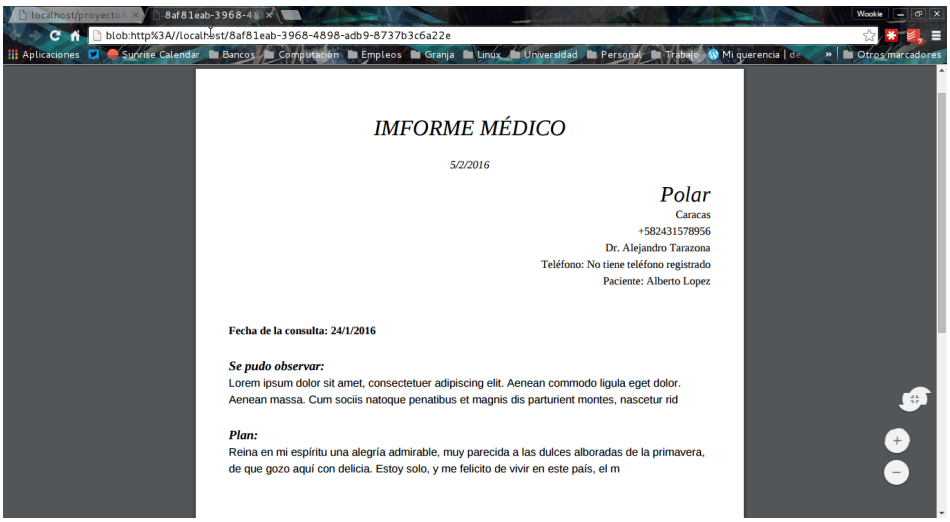

Figura 5.12: Informe modelo

5.3 Fase de Cierre

5.3.1 Primer Sprint: Pruebas finales y puesta en producción

1. Objetivos
    - Resolución de incidencias.
    - Diseño y ejecución de pruebas.
    - Puesta en producción.
2. Actividades
    - Se revisaron las incidencias y posibles errores de consistencia de la información y su acceso en el sistema.
    - Se realizaron pruebas de aceptación de parte del *Scrum Master* en el uso del sistema.

5.3.2 Segundo Sprint: Redacción del libro de pasantías

1. Objetivos
    - Redacción del libro de pasantías.
    - Revisión y culminación del libro de pasantías.
2. Actividades
    - Se recopiló la información del proyecto
    - Se realizó el filtrado de dicha información y se llevó al presente trabajo en limpio para su presentación.

5.3.3 Dificultades generales encontradas

Las dificultades encontradas durante el desarrollo del proyecto fueron:

- **Restricción en el uso de herramientas:** el *Product Owner* solicitó explícitamente el uso de Java, SPRING, Hbernate, JavaScript, MySQL, AngularJS y JSON, tal y como fue descrito, para el desarrollo. Lo cual, si bien facilitó la creación y desarrollo del proyecto, dejó imposibilitada la posibilidad de, con tecnologías actuales, dar soporte a una base de datos orientada a eventos, tal como se pretendía hacer en el principio, y no se logró este enfoque.

- **Falta de informción de parte de Inpsasel:** originalmente se buscaba generar informes médicos especializados para el reporte de enfermedades ocupacionales al Inpsasel, órgano encargado de la gestión de enfermedades ocupacionales en el émbito nacional. Sin embargo, la falta de información, formatos y, más adelante, el lanzamiento de su portal (de Inpsasel) con automatización de dicha gestión, dejó a HxPlus Ocupacional sin la necesidad de realizar dichos informes.

5.3.4 Resultados Generales

El proyecto de pasantía se completó satisfactoriamente. A pesar de no poseer la funcioncionalidad de generación de informes de Inpsasel, la empresa de consideró satisfecha con los módulos implementados. La línea de tiempo de consultas se muestra adecuadamente y en orden cronológico, los diagnósticos y planes son almacenados y mostrados a gusto de la empresa y de conformidad con los lineamientos.

Queda de parte de la empresa gestionar el desarrollo de futuros módulos o funcionalidades para la aplicación.

Capítulo 6

Conclusiones y Recomendaciones

Habiendo terminado el proyecto, se logró la implementación del sistema "HxPlus Ocupacional" para Globisoft S.A. apoyando de esta manera su misión de brindar apoyo tecnológico al área médica en Venezuela. El mercado actual de herramientas en este ámbito está en su fase inicial y, aunque hay resistencia al cambio por parte de los médicos nacionales, la familiaridad y una interfaz planificada para la usabilidad, ayudará a que las nuevas generaciones hagan uso de este sistema.

SCRUM permitió una organización rápida y la debida evaluación oportuna de los objetivos logrados y la planificación de la siguiente serie de objetivos, adaptándose así a los inconvenientes evidenciados.

El proyecto, si bien orientado a medicina ocupacional, se recomienda en un futuro, incluir también el área farmacéutica y su integración con "HxPlus", el cual se encuentra en funcionamiento. Con esto se crearía un sistema distribuido integral de gestión de consultas, remisión de informes médicos y generación de constancias, informes y récipes médicos y que a su vez le permita a las empresas farmacéuticas publicidad y registros en tiempo real, a los pacientes, tener siempre a su disposición los planes de tratamiento, récipes médicos, los diagnósticos realizados y su historial médico en caso de alguna eventualidad como pérdida del mismo u olvidos ocasionales.

También, y en el marco de las tecnologías usadas, se sugiere una evaluación del patrón de almacenamiento de la base de datos ya que el uso de un patrón orientado a eventos podría ser provechoso al momento de recuperación de fallos dentro de la base de datos.

Bibliografía

[1] Ejemplos TIW. *Modelo-Vista-Contolador*. URL: http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html (visitado 05-12-2015).

[2] Universidad de Alicante. *Modelo vista controlador (MVC)*. URL: http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html.

[3] Gartner. *Predicts 2003: SOA is changing Software*. 2002.

[4] María González Quiroga. "ESTUDIO DE ARQUITECTURAS DE REDES ORIENTADAS A SERVICIO". PROYECTO FINAL DE CARRERA. Universidad Politécnica de Calalunya, 2011.

[5] Microsoft. *Service Oriented Architecture*. URL: https://msdn.microsoft.com/en-us/library/bb833022.aspx (visitado 04-12-2015).

[6] *DLE*. URL: http://dle.rae.es/?w=autenticad&origen=REDLE (visitado 19-04-2016).

[7] URL: http://www.wordreference.com/definicion/autenticar (visitado 19-04-2016).

[8] Héctor Mitre B. *Aplicaciones de Autenticación*. Universidad Latina de Panamá.

[9] Hüseyin Babal. *Token-Based Authentication With AngularJS & NodeJS*. URL: http://code.tutsplus.com/tutorials/token-based-authentication-with-angularjs-nodejs--cms-22543 (visitado 04-08-2015).

[10] *Inversión de Control e Inyección de Dependencias*. 15 de ene. de 2014. URL: https://danielggarcia.wordpress.com/2014/01/15/inversio-de-control-e-inyeccion-de-dependecias/ (visitado 19-04-2016).

[11] *Inversión de Control e Inyección de Dependencias en Java/Spring*. 22 de jul. de 2014. URL: http://www.solvetic.com/tutoriales/article/987-inversion-de-control-e-inyeccion-de-dependencias-en-javaspring (visitado 19-04-2016).

[12] Ken Schwaber y Jeff Sutherland. *La Guía Definitiva de Scrum: Las Reglas del Juego*.

[13] Craig Larman y Bas Vodde Pete Deemer Gabrielle Benefield. *SCRUM PRIMER. Una introducción básica a la teoría y práctica de Scrum. Versión 2.0*. Trad. por Ángel Medinilla. 2012.

[14] proyectosagiles.org. *Qué es SCRUM*. URL: http://proyectosagiles.org/que-es-scrum/.

[15] *¿Qué es Apache?* URL: http://culturacion.com/que-es-apache/ (visitado 30-06-2015).

[16] *The Apache Software Foundation*. URL: http://www.apache.org/.

[17] *Apache Tomcat*. URL: http://tomcat.apache.org (visitado 28-06-2015).

[18] Yahoo Developer Network. *Using JSON (JavaScript Object Notation) with Yahoo! Web Services*. URL: http://web.archive.org/web/20100106010113/http://developer.yahoo.com/common/json.html (visitado 25-11-2015).

[19] Json Org. *Introducción a JSON*. URL: http://www.json.org/json-es.html (visitado 25-11-2015).

[20] ejemplosTIW. *Interfaz de Persistencia Java (JPA) - Entidades y Managers*. URL: http://www.lab.inf.uc3m.es/~a0080802/RAI/jpa.html (visitado 10-08-2015).

[21] Google. *Superheroic JavaScript MVW Framework*. URL: https://angularjs.org/ (visitado 01-08-2015).

[22] La enciclopedia libre Wikipedia. *JavaScript*. URL: https://es.wikipedia.org/wiki/JavaScript (visitado 01-08-2015).

[23] Jose Antonio Rodríguez. *Manual de JavaScript*. URL: http://www.internetmania.net.

[24] *Reference Manual*. URL: http://dev.mysql.com/doc/refman/5.7/en/introduction.html (visitado 04-01-2016).

[25] ORACLE. *MySQL, The World's Most Popular Open Source Database*. URL: http://www.oracle.com/us/products/mysql/overview/index.html (visitado 04-08-2015).

[26] Rod Johnson et al. *Spring Framework Reference Documentation*. 2004. URL: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/.

[27] Rod Johnson et co. *Spring Framework Reference Documentation*. 2004. URL: http://docs.spring.io/spring/docs/4.2.0.RC1/spring-framework-reference/htmlsingle/#extensible-xml-resources (visitado 19-07-2015).

[28] Hibernate comunity. *Hibernate ORM*. URL: http://hibernate.org/orm/ (visitado 03-07-2015).

[29] Tutorials Point (I) Pvt. Ltd. *Hibernate Java persistence framework*. URL: http://www.tutorialspoint.com/hibernate/hibernate_tutorial.pdf.

[30] ITEXT corporation. *The future belongs to automated PDF*. URL: http://itextpdf.com/ (visitado 10-09-2015).

[31] Eclipse Foundation. *Eclipse - The Eclipse Foundation*. URL: https://eclipse.org/.