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
UI	User Interface
USB	Universal Serial Bus

Introducción

Actualmente en Venezuela los historiales médicos de los pacientes son llevados en papel por cada institución o sucursal de manera individual. Esto puede representar inconvenientes para los pacientes al momento de solicitar o requerir un cambio en la institución tratante, sea cual sea la razón. En estos casos los pacientes deben llenar los requisitos para la realización de un nuevo historial médico en la nueva institución y esto puede resultar en pérdidas de información por parte del nuevo médico tratante.

Globinsoft S.A. se dispuso a crear una plataforma web de alcance nacional y proyección internacional para la gestión integral de los historiales médicos que a su vez sirva de vínculo con las empresas farmacéuticas y farmacias en general. Con esta integración se busca facilitar la adquisición de medicamentos de parte de los pacientes y poder llevar un registro integral de los tratamientos, prescripciones y exámenes médicos a los que se haya sometido el paciente.

Según Lopcymat, se considera una enfermedad ocupacional los estados patológicos contraídos o agravados con ocasión del trabajo o exposisción al medio en que el trabajador o trabajadora se encuentra obligado a trabajar. En este orden de ideas, se planteó la creación de "HxPlus Ocupacional" como plataforma web de gestión de historias médicas orientado exclusivamente al área de medicina ocupacional.

Objetivo general: Desarrollar la primera versión de una aplicación que permita la gestión de historias médicas de los empleados de una empresa y automatice la generación de informes médicos a los doctores y departamento de recursos humanos.

Objetivos específicos: Para la primera verisón, la aplicación debe contar con:

- **Módulo Usuarios:** Realizar operaciones de gestión de usuarios.
- **Módulo Historias:** Realizar operaciones de gestión de historias médicas así como generar informes médicos, reposos médicos y planillas de remisión de pacientes.

- **Módulo Reportes:** Generar los documentos requeridos por Inpsasel así como los reportes requeridos por la empresa.

Estructura del informe

El presente informe se compone de 6 capítulos que contienen:

1. Descripción de la organización de Globinsoft S.A. así como información sobre proyectos previos realizados que sirven de precedente para "HxPlus Ocupacional".
2. Definición de los conceptos utilizados y manejados durante el desarrollo del proyecto.
3. Definición y estructura de la metodología utilizada para el desarrollo del proyecto.
4. Descripción de las diferentes herramientas utilizadas, ya sea para el desarrollo como para la gestión del proyecto.
5. Información detallada y puntual de las etapas de desarrollo de la aplicación.
6. Conclusiones generales.

También cuenta con información adicional en el capítulo de apéndices, que puede servir como una visión alternativa de la aplicación o como un soporte adicional a lo descrito en los capítulos precedentes.

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. es una empresa destinada a prestar servicios en el área médica, brindado herramientas y soporte con la finalidad de facilitar, tanto a médicos y asistentes como a pacientes, el acceso a la información sobre las historias médicas, en caso de los pacientes sólo a su historia personal.

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.

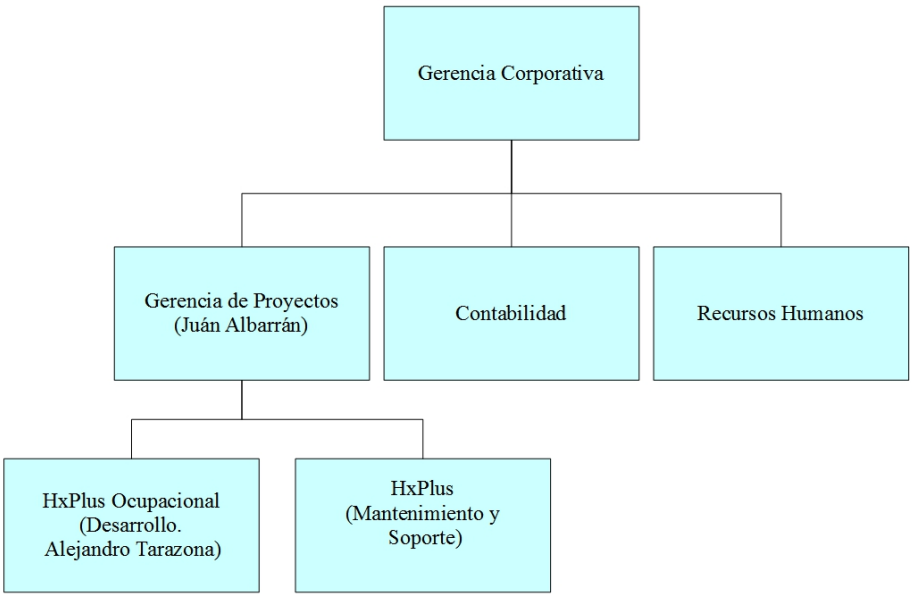

Figura 1.1: Estructura Organizacional de Globinsoft

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."

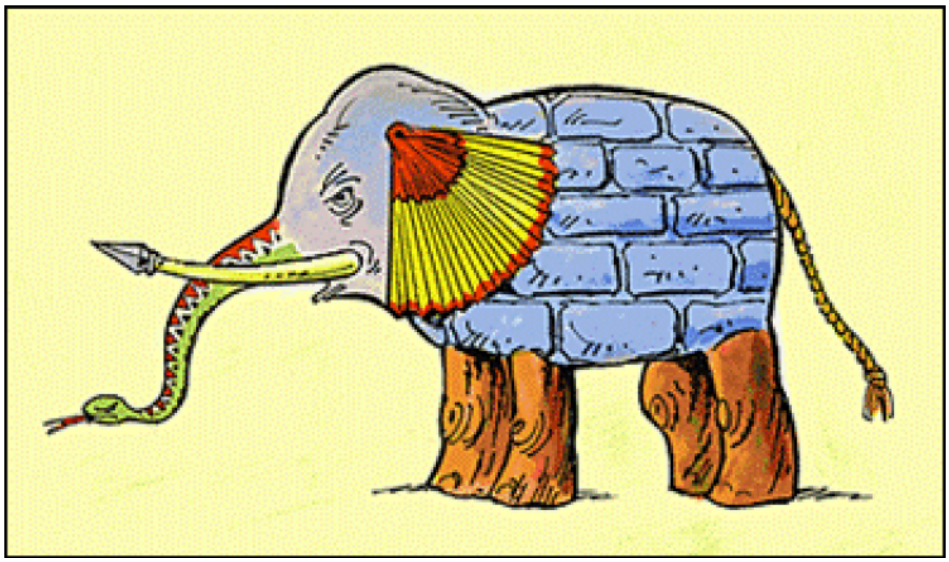

Figura 2.2: El elefante de Saxe

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.

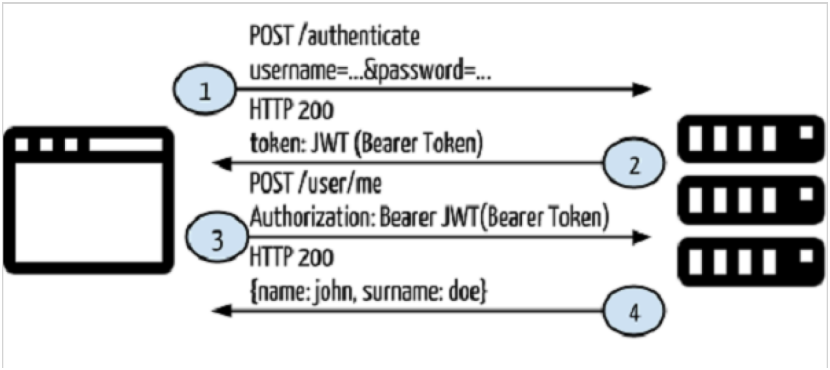

Figura 2.3: Autenticacion basada en fichas.

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” (figura 2.4).

```
@Controller
@RequestMapping(value = "/user")
public class UserController {

    @Autowired
    UserService userService;

    @RequestMapping(value = "/{id}", method = RequestMethod.GET)
    public @ResponseBody
    User getUser(@PathVariable("id") Long id) {

        return userService.findById(id);
    }
```

Figura 2.4: Ejemplo de inyección de dependencias en 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 (figura 2.5).

```
@Service
public class UserServiceImpl implements UserService{

    @Autowired UserRepository userRepository;

    public User findById(Long id) {
        return userRepository.findOne(id);
    }
```

Figura 2.5: Ejemplo de inyección de dependencias en servicios.

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:

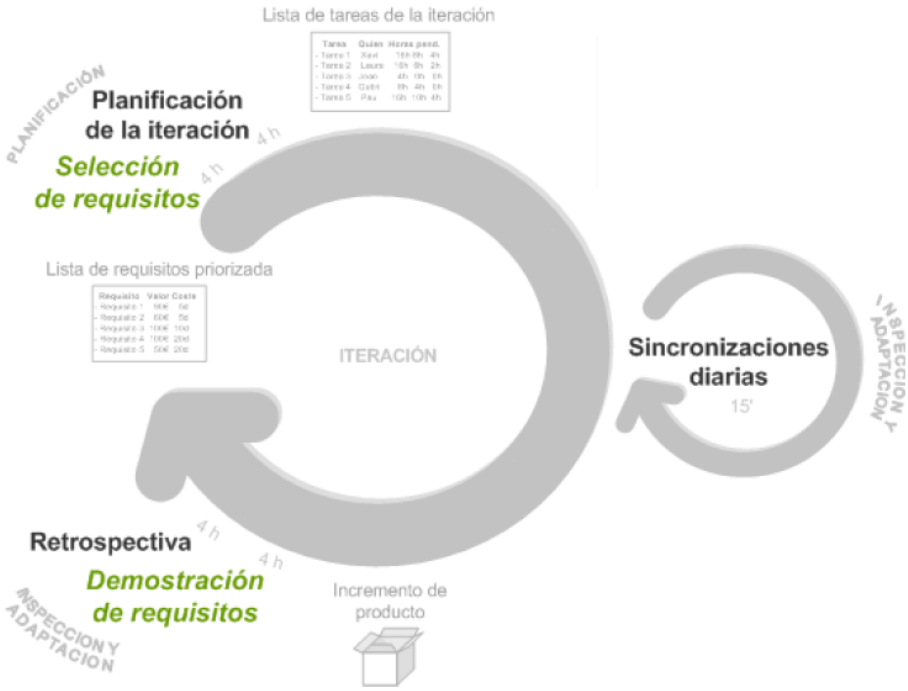

Figura 3.1: Esquema de trabajo de SCRUM

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 llevó a cabo 3 fases:

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

3. Cierre (2 Sprint)

Tal y como se describe en el capítulo **??**.

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 bireccional 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 y finaliza con una revisión de dificultades técnicas y los resultados del proyecto.

El desarrollo se dividió en tres fases:

1. Fase de Preparación.
2. Fase de Implementación.
3. Fase de Cierre.

Las cuales poseen sus propios objetivos generales y específicos, la fase de implementación contempla también los *Sprint* realizados siguiendo la metodología seleccionada.

5.1 Fase de Preparación

Objetivo General: Llevar a cabo el levantamiento de requerimientos la descarga de las herramientas a utilizar durante el desarrollo y la configuración inicial de las mismas.

Objetivos Específicos:

- Levantamiento de los requerimientos del sistema.
- Descarga y configuración de Eclipse, JSON, Hibernate, Liquibase, Maven, AngularJS, Apache y Tomcat.

- Creación de cuentas para el manejo de los repositorios del proyecto y *Product Backlog*.
- Creación del *Product Backlog*.
- Elaboración del diagrama inicial de base de datos.
- Evaluación de restricciones y riesgos por parte del equipo de desarrollo.

En esta fase se realizó, en conjunto con el ing. Juán Albarrán, el levantamiento de los requerimientos funcionales y no funcionales del sistema, dichos requerimientos fueron:

- **Requerimientos Generales:**

  El sistema debe proveer una plataforma de almacenamiento de historiales médicos con asociaciones entre la empresa en la que trabaja el paciente, la zona o "centro de costos" en el cual labora el paciente al momento de la consulta, mantener los historiales a través del tiempo, sin importar los contratos contraidos por el paciente ni el médico que lo atienda.

  En casos de cambio de centro de costos de trabajo, se deberán especificar como terminación o renovación de contratos de parte de los usuarios y la empresa.

  Todo esto con la finalidad de poder realizar cruces de información e investigación de causas de enfermedades tomando en cuenta los lugares de trabajo y las tareas desempeñadas por los trabajadores.

  El sistema debe poseer mecanismos de autenticación, se sugirió el uso de autenticación basada en *tokens* para este requerimiento.

- **Usuarios:**

  El sistema debe ser capaz de agregar nuevos usuarios que serán almacenados en la base de datos con los atributos especificados en A que a su vez puedan ser utilizados tanto en historiales médicos como en pacientes de algún doctor de la empresa. También los doctores de cada empresa tienen que ser registrados como usuarios del sistema y ser asignados al cargo de "Doctor" de algún departamento de dicha empresa.

  El cargo de "Doctor" en una empresa se refiere al personal médico calificado para realizar consultas y diagnósticos médicos. Por consecuencia será representado como un contrato especial y, sin discriminación por la especialidad o nivel de instrucción del mismo, será almacenado en el sistema con el nombre de "Doctor". En la sección de almacenamiento de los datos del doctor, será almacenada la información específica del mismo.

- **Historial Médico:**

  El historial médico de un paciente se realiza al momento de la primera consulta que sea registrada en el sistema, en ese momento de registran:

1. Antecedentes o Trasfondos: Enfermedades diagnosticadas previamente al paciente así como predisposiciones a enfermedades hereditarias.
2. Hábitos: Son actividades regulares del paciente. Pueden variar en tipo y frecuencia. Éstos pueden ser habitos recomendados o perniciosos para el paciente. La finalidad es que las investigaciones de enfermedades ocupacionales puedan descartar los hábitos de los pacientes como factores.
3. Alergias: También se registran alergias diagnosticadas al paciente. Se realiza en una sección a parte de los trasfondos debido a que las alergias poseen características específicas y se manifiestan sólo con la presencia del alérgeno[1] correspondiente.
4. Vacunas: Por último ser registran las vacunas que posee el paciente detallando el nombre y la potencia en caso en que aplique (vacunas de varias dosis o refuerzos).

El usuario debe estar previamente registrado y contratado por la empresa que contrata al doctor para poder crearse su historial médico.

- **Consultas Médicas:**

  Una vez creado el historial médico pueden realizarse consultas médicas. En ellas se registrará la información recogida durante dicha consulta y se almacenará en el sistema.

  Basándose en la información recopilada por Globinsoft S.A. para su proyecto "HxPlus" (principal antecedente del presente proyecto), queda establecida la estructura de una consulta de la siguiente forma:

  1. *SoapNote* (nota de revisión): Usada por los médicos para organizar las consultas y que consta de:
     (a) Subjective: Información subjetiva, comunmente redactada en lenguaje informal, que contiene lo que el paciente describe, dolencias y síntomas presentados.
     (b) Objective: Información objetiva recopilada por el médico durante la consulta. Redactada en lenguaje formal.
     (c) Assessment: Comentarios u observaciones realizados por el médico.
     (d) Plan: Plan acción a realizar para tratar las dolencias.
  2. Diagnóstico(s): Uno o varios diagnósticos realizados por el médico en la consulta.
  3. Instrucción(es): Una o varias instrucciones. Acciones que deben ser llevadas al pie de la letra en cuanto a conductas o hábitos del paciente, se incluye en este apartado regulaciones alimenticias y recomendaciones periódicas.

[1] Según el diccionario de la RAE: *alérgeno, na*: 1. adj. Perteneciente o relativo a los alérgenos. 2. m. Sustancia antigénica que induce una reacción alérgica en un organismo.

4. Prescripción(es): Instrucciones de uso de medicamentos registrados en el sistema junto con la información de dicho medicamento. Con esto se podrán generar los récipes médicos para la adquisición de fármacos que así lo requieran.
5. Signos Vitales: Se registran los signos vitales recabados en la consulta. Dado que puede variar los signos vitales pertinentes entre las distintas especialidades de la medicina, se deja a juicio del médico tratante qué signos vitales serán tomados en la consulta.
6. Solicitud y Recepción de exámenes médicos: El la primera se registra una solicitud de un examen dado. En una futura consulta puede darse por recibido en el sistema el examen médico y adjuntar el archivo respectivo para su almacenamiento. Se puede solicitar o recibir más de un examen médico por consulta.

El sistema debe ser capaz de almacenar y desplegar esta información ordenada cronológicamente a partir de la consulta médica más reciente.

- **Reportes Médicos:**

  El sistema debe poder generar de manera automática los reportes médicos con la información recopilada en las consultas a solicitud del médico.

Todos estos requerimientos fueron archivados en notas para la realización del *Product Backlog* utilizando el sistema Trello, para lo cual se procedió a crear la cuenta del equipo de desarrollo y se enlazó con la pizarra creada por el *Scrum Manager* para la gestión del proyecto.

Se dividió el proyecto en "módulos" y éstos a su vez en "vistas" las cuales pasaron a ser la unidad granular deseada para el manejo del proyecto, quedando así la primera versión de la lista del producto de la siguiente forma:

1. **Módulo de Autenticación:**
   - Vista de autenticación: Incluye inicio y cierre de sesión dentro del sistema.
2. **Módulo de Gestión de Usuarios:**
   - Vista de Lista de Usuarios: Lista de los usuarios registrados en el sistema. Esta vista será refinada en el futuro para que haga discriminación entre usuarios por empresa (ej. un usuario de una empresa A no podrá ver la lista de otra empresa B).
   - Vista de Detalles de Usuario: Vista de los datos personales de un usuario.
   - Vista de Edición de Usuario: Formulario lleno con los datos personales de un usuario dado que se usa para la actualización de dichos datos.

3. **Módulo de Doctores:**

    - Vista de Agregar Doctor: Formulario para el registro de un nuevo dorctor en la empresa.
    - Vista de Detalles de Doctor: Vista de los datos característicos de un doctor.
    - Vista de Edición de Doctor: Formulario lleno con los datos profesionales de un doctor dado. Se usa para la actualización de dichos datos.

4. **Módulo de Pacientes:**

    - Vista de Añadir Nuevo Paciente: Al añadir un nuevo paciente, el doctor puede añadir un nuevo paciente a su lista de pacientes atendidos y el mismo puede provenir de las listas de usuarios atendidos previamente y que estén contratados por la empresa o ser un paciente totalmente nuevo, en tal caso pasaría a la siguiente vista.
    - Vista de Lista de Pacientes Atendidos: El doctor tiene a su disposición una lista de pacientes que ha atendido con anterioridad, ordenada por fecha de última consulta médica y se mostrarán los pacientes atendidos más recientemente primero.
    - Vista de Creación de Historia Médica: Formulario con el cual se añaden los datos necesarios para la creacción de una nueva historia médica al sistema. Usado en el caso de que un paciente esté siendo atendido por primera vez desde su registro en el sistema (no importa si ha cambiado de puestos de trabajo, sólo se discrimina por el puesto actual).

5. **Módulo de Consultas Médicas:**

    - Vista de Historial de Consultas Médicas del Paciente: Una lista de las consultas que ha tenido el paciente ordenadas cronológicamente empezando desde la última consulta.
    - Vista de Agregar Consulta Médica: Formulario con los datos requeridos para almacenar una consulta médica nueva.
    - Vista de Revisión de Consulta Médica: En esta vista se despliegan los datos almacenados de una consulta médica ya realizada. También se accede a los archivos previamente almacenados (exámenes médicos recibidos) y a la funcionalidad de generación de reportes.
    - Vista de Generación de Reportes Médicos: En esta vista de podrán elegir los reportes que serán generados por el sistema.

Todo esto quedó plasmado en el sistema de pizarras de Trello asociado a la cuenta del equipo de desarrollo.

Basándose en los requerimientos recopilados, el equipo de desarrollo, realizó un diagrama de base de datos el cual a su vez fue modificado durante el desarrollo del proyecto. La versión final del mismo puede encontrarse en A.1.

Del análisis de riesgos del proyecto, se obtuvo la siguiente lista de riesgos:

1. **Equipo de desarrollo pequeño:** Dado que se cuenta con un sólo integrante del equipo de desarrollo los avances en la implementación del proyecto deben ser pequeños con la intención de que sean funcionales y estén probados.

2. **Falta de diversidad en el conocimiento:** También por ser un grupo pequeño, pueden surgir, durante el desarrollo, impedimentos por falta de dominio de las herramientas. En tal caso se deberá realizar una fase de capacitación para las fases impedidas lo cual podría retrasar las entregas.

3. **Tiempo dedicado a los *Sprint*:** Si los *Sprint* son establecidos en una o dos semanas, los avances podrán verse rápidamente, aunque serían poco avance. Por otro lado, mantener el tiempo de *Sprint* en tres o cuatro semanas, se verían pocos avances y la retroalimentación sería escasa. Para solventar tal riesgo, se designó un tiempo de *Sprint* variable, quedando establecido un tiempo de dos semanas para aquellas vistas en las que el equipo de desarrollo no precise capacitación y de tres a cuatro semanas para las que si sea necesaria la capacitación.

En paralelo a la realización de este trabajo, se realizó la descarga Eclipse Kepler, descargado de la página oficial de Eclipse[31], Apache, Tomcat (ambos descargados de [16, 17] respectivamente) y las librerías de AngularJS.

Se procedió a la configuración de Maven y se agregaron las dependencias iniciales 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.

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

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

Se realizaron los cambios dentro del archivo “occupational-servlet.xml” que permiten el uso de anotaciones para el direccionamiento interno de los procesos.

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

También, 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 Fase de Implementación

En esta fase se realizó en sí el desarrollo del sistema. Hubo modificaciones en el diagrama de base de datos y los requerimientos que cambiaron, se reflejaron en el *product backlog* conforme fueron sucediendo.

La implementación se subdividió en Sprints y, como se menciona anteriormente, estos fueron de duración variable.

Cabe destacar que dado que los módulos referentes al manejo de empresas (“Compañía”, “Agregar Usuario”, “Contratos”) están por fuera del alcance del proyecto, se utilizó una base de datos de pruebas generada exclusivamente para las pruebas. También fueron generados datos de prueba para el medicamentos y laboratorios, por las mismas razones expresadas con anterioridad.

5.2.1 Primer Sprint: Vista de Autenticación de Usuarios

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.

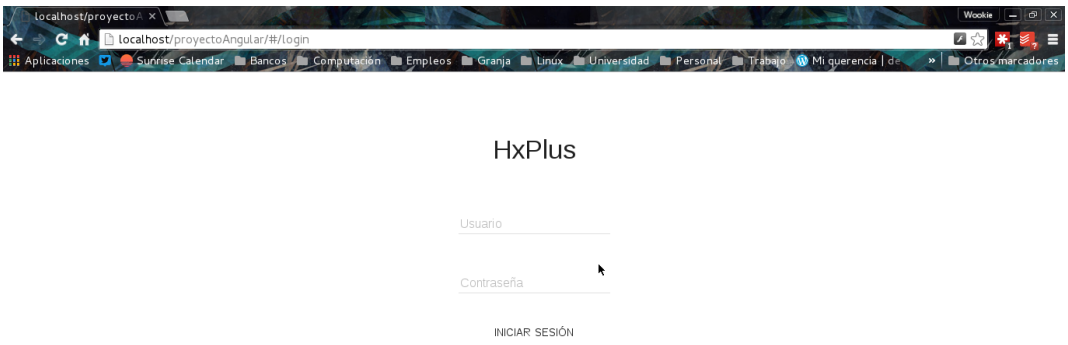

Figura 5.1: Pantalla de Autenticación de usuarios

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

Las pruebas sobre esta vista fueron realizados evaluando la generación de la ficha de parte del servidor en *back end* y su posterior uso para el acceso a las demás páginas del sistema. La ficha fue generada con éxito y verificado el cambio de ficha en cada inicio de sesión, se dió la aprobación de la vista para proseguir al siguiente Sprint.

5.2.2 Segundo Sprint: Vista de Lista de Usuarios

Esta vista contiene una lista con los nombres de los usuarios registrados en el sistema. Para el alcance del sistema se implementó una lista general sin discriminaciones sobre dicha lista y fue utilizada en

las pruenas de la vista de “Detalles de Usuario”. Se plantea que a futuro sea utilizada por usuarios especiales dentro del sistema y que manejen dichas listas dentro de una empresa.

A este Sprint se le asignó una semana de lapso para su entrega dada la importancia relativamente baja de la vista no se refinó y su UI está muy poco refinada.

Las pruebas de la vista se basaron en el orden de aparición de los usuarios (ordenados alfabeticamente) y la actualización de la lista al agregar un nuevo usuario.

5.2.3 Tercer Sprint: Vista de Detalles de Usuario

Esta vista es la muestra de los datos ingresados al sistema del usuario elegido. Es un paso previo a la edición del usuario, si este así lo desea y se mantiene con los datos actuales del usuario.

A este Sprint, debido a su simpleza, se le dedicó sólo una semana de tiempo para realizar las pruebas necesarias.

Por ser una vista sencilla, las pruebas se realizaron correlacionando los datos ingresados de los usuarios con lo que se muestra en pantalla.

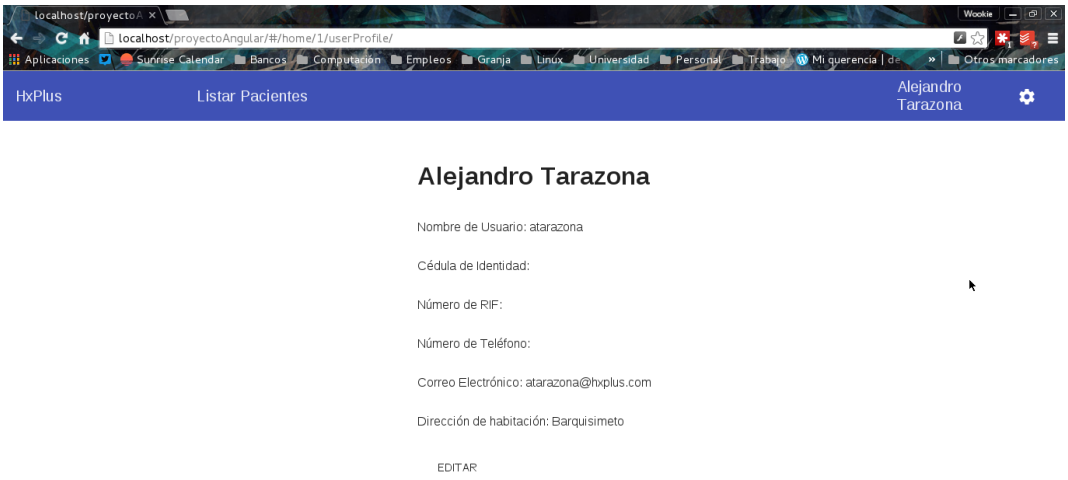

Figura 5.2: Pantalla de Revisión de Usuario.

5.2.4 Cuarto Sprint: Vista de Edición de Usuario

Esta vista muestra un formulario, similar al formulario de registro de usuario, lleno previamente con los datos de la vista de "Detalles de Usuario". Si el formulario cambia y estos cambios son enviados, el sistema los almacena y regresa a la vista anterior, en caso de éxito, y muestra los datos actualizados del usuario. En caso de algún fallo, el sistema debe dar el mensaje correspondiente al cambio ilegal y mantenerse en la vista con los datos originales del usuario. En caso de modificaciones de la contraseña, el usuario no podrá ver la contraseña previa y el campo estará vacío; este campo se llenará en caso de desear modificarla y se necesita también una verificación de doble escritura de la contraseña para tal fin.

Las pruebas de esta vista fueron:

1. Alterar la información del usuario eliminando campos requeridos (nombre de usuario, contraseña, primer nombre, primer apellido, número de cédula y correo electrónico).
2. Cambiar la contraseña con errores en la doble verificación de escritura.
3. Cambiar el nombre de usuario, número de cédula o correo electrónico por valores ya registrados en el sistema.
4. No alterar ningún dato del usuario y enviar el formulario con los datos previos.
5. Verificar que los datos fuesen actualizados en la vista anterior en caso de haber verifiaciones válidas en el usuario dado.

A partir del caso 3 se detectó que el sistema realizaba una modificación completa del usuario en la base de datos con los datos enviados en el formulario, lo cual podría devenir en problemas de procesamiento a futuro con grandes cargas de datos y es innecesario en el caso 4 por la misma naturaleza de la no modificación. Se acordó en el siguiente Sprint, revisar esta funcionalidad y agregar un detector a nivel de cliente para enviar sólo los datos modificados por el usuario en la vista.

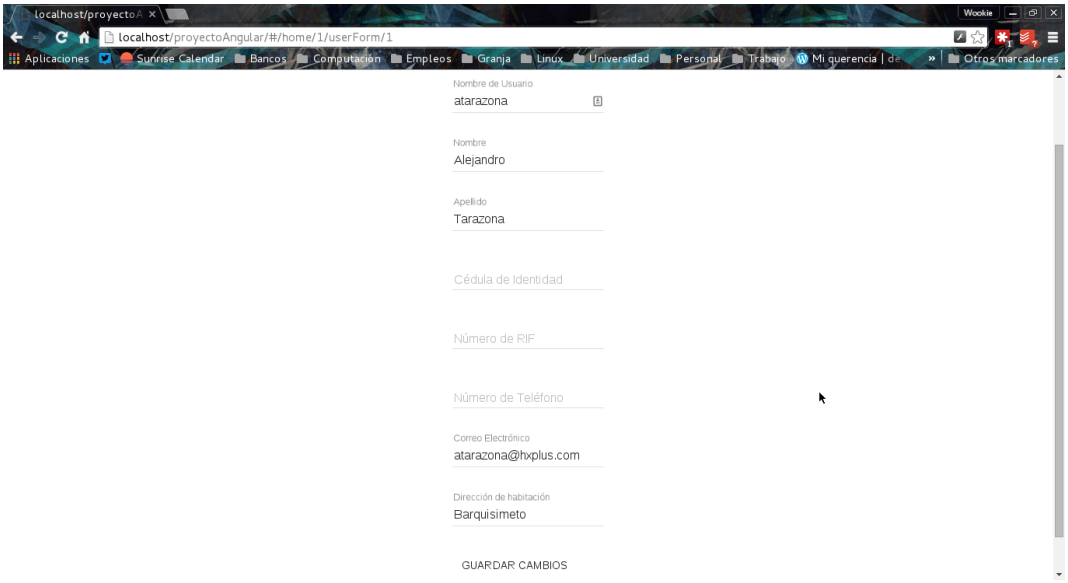

Figura 5.3: Pantalla de Edición de Usuario.

5.2.5 Quinto Sprint: Vista de Agregar Doctor

5.2.6 Sexto Sprint: Vista de Detalles del Doctor

5.2.7 Septimo Sprint: Vista de Edición de Doctor

5.2.8 Octavo Sprint: Vista de Añadir Nuevo Paciente

5.2.9 Noveno Sprint: Vista de Lista de Pacientes Atendidos

5.2.10 Décimo Primer Sprint: Vista de Creación de Historia Médica

5.2.11 Décimo Primer Sprint: Vista de Historial de Consultas Médicas del Paciente

5.2.12 Décimo Segundo Sprint: Vista de Revisión de Consulta Médica

5.2.13 Décimo Tercer Sprint: Vista de Generación de Reportes Médicos

5.3 Fase de Cierre

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.

Apéndice A

Diagrama ER-E y Glosario de términos

Entidad	Descripción	Atributo	Descripción	Tipo
USER	Información de los usuarios registrados en el sistema.	id	Identificador del usuario en el sistema.	Clave, autogenerado, BIGINTEGER.
		username	Nombre de usuario a utilizar en el sistema.	VARCHAR.
		password	Contraseña privada del usuario en el sistema.	VARCHAR.
		ci	Documento de identificación del usuario en el país. Número de cédula de identidad	BIGINTEGER.
		rif	Documento de identificación del usuario en el país. Número de registro de información fiscal.	BIGINTEGER.

		firstName	Primer nombre del usuario.	VARCHAR.
		lastName	Primer apellido del usuario.	VARCHAR.
		sex	Sexo del usuario.	“m” o “f”.
		birthdate	Fecha de nacimiento del usuario.	DATETIME.
		email	Dirección de correo electrónico del usuario.	VARCHAR.
		address	Dirección de habitación del usuario.	Monovaluado, VARCHAR.
		phoneNumber	Número de teléfono del usuario.	Monovaluado, VARCHAR.
		idpost	Identificador del puesto asignaddo al usuario.	Monovaluado, BIGINTEGER.
		idcostcenter	Identificador del centro de costo donde trabaja el usuario.	BIGINTEGER.
		idcompany	Identificador de la compañía donde trabaja el usuario.	BIGINTEGER.
		photo	Fotografía tipo carnet del usuario.	BIGINTEGER.
COMPANY	Información de las empresas registradas en el sistema.	id	Identificador de la compañía registrada.	Clave, autogenerado, BIGINTEGER.
		companyname	Nombre de la compañía.	VARCHAR.

		rif	Documento de identificación de la compañía en el país. Número de registro de información fiscal.	BIGINTEGER.
		description	Descripción de la compañía.	VARCHAR.
		idcostcenter	Identificador del centro de costos que es sede principal de la compañía.	BIGINTEGER.
COSTCENTER	Centro de costos de una empresa. Se toma como centro de costos una sede o un almacén de la empresa.	id	Identificador del centro de costos.	Clave, autogenerado, BIGINTEGER.
		idcompany	Identificador de la compañía a la que pertenece el centro de costos.	BIGINTEGER.
		address	Dirección del centro de costos.	VARCHAR.
		phoneNumber	Número de teléfono principal del centro de costos.	VARCHAR.
DEPARTMENT	Departamentos de una compañía.	id	Identificador del departamento en el sistema.	BIGINTEGER.
		name	Nombre del departamento.	VARCHAR.
		description	Descripción del departamento.	VARCHAR.

		idcompany	Compañía a la que pertenece el departamento	BIGINTEGER.
POST	Puestos de trabajos ofrecidos en una caompañía.	id	Identificador del puesto en el sistema.	BIGINTEGER.
		name	Nombre del puesto.	VARCHAR.
		description	Descripción de las funciones del puesto.	VARCHAR.
		iddepartment	Departamento al que pertenece el puesto.	BIGINTEGER.
PATIENT	Usuarios que han sido atendidos por médicos de una empresa.	id	Identificador del paciente. También usado como número de historial médico dentro del sistema.	Clave, autogenerado, BIGINTEGER.
		iduser	Usuario asociado.	BIGINTEGER.
HABIT	Parte del historial médico. Son hábitos mencionados por el paciente al momento de la creación de su historial médico.	id	Identificador del hábito en el sistema.	Clave, autogenerado, BIGINTEGER.
		name	Nombre del hábito.	VARCHAR.
		frecuency	Frecuencia con la cual practica el hábito.	VARCHAR.
		idpatient	Paciente que está practicando el hábito.	BIGINTEGER.

BACKGROUND	Parte del historial médico. Enfermedades previas o actuales, condiciones hereditarias o predisposiciones a enfermedades diagnosticadas al paciente.	id	Identificador del trasfondo en el sistema.	Clave, autogenerado, BIGINTEGER.
		idpatient	Paciente que presenta el trasfondo.	BIGINTEGER.
		name	Nombre del trasfondo o enfermedad.	VARCHAR.
		description	Descripción de síntomas o tratamientos llevados a cabo en virtud del trasfondo.	VARCHAR.
ALLERGY	Parte del historial médico. Alergias manifestadas por el paciente.	id	Identificador de la aleragia en el sistema.	BIGINTEGER.
		name	Nombre de la alergia. En caso de no conocerlo se indica a qué es alergico el paciente.	VARCHAR.
		description	Descripción más detallada de la alergia.	VARCHAR.
		severity	Severidad en caso que aplique y que haya sido diagnosticada por parte de un doctor.	VARCHAR.
		idpatient	Paciente que presenta la alergia.	BIGINTEGER.

VACCINE	Parte del historial médico. Historial de vacunas del paciente.	id	Identificador de la vacuna en el sistema.	BIGINTEGER.
		name	Nombre de la vacuna.	VARCHAR.
		potency	Pontencia de la misma en caso que aplique.	VARCHAR.
		idpatient	Paciente que tiene la vacuna.	BIGINTEGER.
DOCTOR	Usuarios que tienen el puesto de doctores en alguna empresa registrada en el sistema.	id	Idenficador del dorctor en el sistema.	BIGINTEGER.
		iduser	Datos del usuario en el sistema.	BIGINTEGER.
		regnumber	Documento de identificación del doctor ante el colegio de médicos. Número de registro.	BIGINTEGER.
VITALSIGN	Signos vitales registrados en una consulta.	id	Identificador de la nota de signo vitale en el sistema.	BIGINTEGER.
		idconsult	Identificador de la consulta dónde se tomaron los signos vitales.	BIGINTEGER.
		name	Nombre del signo vital tomado.	VARCHAR.
		description	Descripción de lo observado o valores tomados.	VARCHAR.

SOAPNOTE	Parte de una consulta. Nota de revisión.	id	Identificador de la nota de revisión en el sistema.	Clave, autogenerado, BIGINTEGER.
		subjective	Parte subjetiva de la nota de revisión de la consulta. Lo que el paciente expresa y son sus sítomas de manera informal.	VARCHAR.
		objective	Parte objetiva de la nota de revisión. Lo que el doctor observa en la consulta. Lenguaje formal.	VARCHAR.
		plan	Plan de acción a realizar por el paciente.	VARCHAR.
		comments	Comentarios adicionales por parte del doctor.	VARCHAR.
		iddiagnostic	Diagnóstico realizado durante la consulta, si aplica.	BIGINTEGER.
FILE	Archivos almacenados en el sistema. Fotos de usuario, logos de compañías o resultados de exámenes.	id	Identificador del archivo en el sistema.	BIGINTEGER.
		idconsult	Consulta en la cual se realizó la carga del archivo. Esto en caso de ser resultado de un examen.	BIGINTEGER.
		filename	Nombre con el que se guarda el archivo.	VARCHAR.

		type	Tipo de archivo. Si es imágen o archivo de formato portátil.	VARCHAR.
		filedata	Datos del archivo.	LONGBLOB.
DIAGNOSTIC	Parte de una soapnote. Diagnóstico realizado.	id	Identificador del diagnóstico en el sistema.	BIGINTEGER.
		idconsult	Consulta en que fue realizado el diagnóstico.	BIGINTEGER.
		details	Detalles del diagnóstico realizado.	VARCHAR.
		idexam	Examen por el cual se detectó una enfermedad o dolencia diagnósticada. Si aplica.	BIGINTEGER.
LABORATORY	Laboratorio encargado de fabricar un medicamento.	id	Identificador del laboratorio en el sistema.	Clave, autogenerado, BIGINTEGER.
		name	Nombre del laboratorio.	VARCHAR.
		phoneNumber	Número de teléfono principal del laboratorio.	VARCHAR.
DRUG	Medicamento que es registrado en el sistema para complementar los tratamientos médicos.	id	Identificador del medicamento en el sistema.	Clave, autogenerado, BIGINTEGER.
		name	Nombre del medicamento.	VARCHAR.
		description	Descripción de la presentación del medicamento.	VARCHAR.

		idlaboratory	Laboratorio que fabrica el medicamento	BIGINTEGER.
INDICATION	Indicaciones o posología de uso de un medicamento.	id	Identificador de la indicación en el sistema.	Clave, autogenerado, BIGINTEGER.
		description	Descripción o explicación de la indicación.	VARCHAR.
PRESCRIPTION	Prescripción médica. Necesaria para elaborar los récipes médicos. Contiene información de un medicamento y sus indicaciones.	id	Identificador de la prescripción en el sistema.	Clave, autogenerado, BIGINTEGER.
		date	Fecha en que se realizó la prescripción.	DATETIME.
		iddoctor	Doctor responsable de la prescripción.	BIGINTEGER.
		idconsult	Consulta en que fue realizado la prescripción.	BIGINTEGER.
		iddrug	Medicamento prescrito.	BIGINTEGER.
		idindication	Indicación a seguir por parte del paciente para el medicamento prescrito.	BIGINTEGER.

INSTRUCTION	Instrucciones médicas para el tratamiento que no contienen medicamentos. También recomendaciones sobre hábitos del paciente.	id	Identificador de la instrucción en el sistema.	Clave, autogenerado, BIGINTEGER.
		instruction	Información de la instrucción a seguir por parte del paciente.	VARCHAR.
		idconsult	Consulta asociada a la indicación realizada.	BIGINTEGER.
EXAM	Examenes médicos que sean solicitados a través del sistema.	id	Identificador del examen en el sistema.	BIGINTEGER.
		ordered	Consulta en la cual fue ordenado el examen.	BIGINTEGER.
		type	Tipo o nombre del examen solicitado.	VARCHAR.
		results	Resultados obtenidos el examen.	BIGINTEGER.
CONSULT	Datos referentes a una consulta médica.	id	Identificador de la consulta en el sistema.	BIGINTEGER.
		idpatient	Paciente atendido durante la consulta.	BIGINTEGER.
		consultdate	Fecha en la que se realizó la consulta.	DATETIME.
		idsoapnote	Nota de revisión de la consulta.	BIGINTEGER.

		idprescription	Prescripción realizada en la consulta.	BIGINTEGER.
		iddoctor	Doctor tratante.	BIGINTEGER.
ATTENDS	Relación de atención entre doctores y pacientes.	id	Identificador de la relación en el sistema.	BIGINTEGER.
		iddoctor	Doctor tratante.	BIGINTEGER.
		idpatient	Paciente atendido.	BIGINTEGER.
RECIEVE_EXAM	Datos de recepción de un examen solicitado.	id	Identificador de la relación en el sistema.	BIGINTEGER.
		idconsult	Consulta en la cual se recibió el examen.	BIGINTEGER.
		idexam		BIGINTEGER.
HAVE_INST	Relación entre un diagnóstico realizado y las instrucciones que recomienda el doctor.	id	Identificador de la relación en el sistema.	BIGINTEGER.
		idinstruction	Instrucción asociada.	BIGINTEGER.
		iddiagnostic	Diagnóstico asociado.	BIGINTEGER.
CONTRACT	Contrato de trabajo entre un usuario y la empresa.	id	Identificador del contrato en el sistema.	
		begindate	Fecha de firma o comienzo del contrato.	DATETIME.
		enddate	Fecha de culminación del contrato.	DATETIME.
		idpost	Puesto otorgado mediante el contrato.	BIGINTEGER.
		iduser	Usuario contratado.	BIGINTEGER.

		idcostcenter	Centro de costos asignado para la ejecución del contrato por parte del contratado.	BIGINTEGER.
HAVEPOST	Relación entre un centro de costos y los puestos de trabajo que están ofrecidos en el mismo.	id	Identificador de la relación en el sistema.	BIGINTEGER.
		idpost	Puesto ofertado.	BIGINTEGER.
		idcostcenter	Centro de costos que ofrece el puesto.	BIGINTEGER.

Cuadro A.1: Glosario de términos del diagrama ER-E

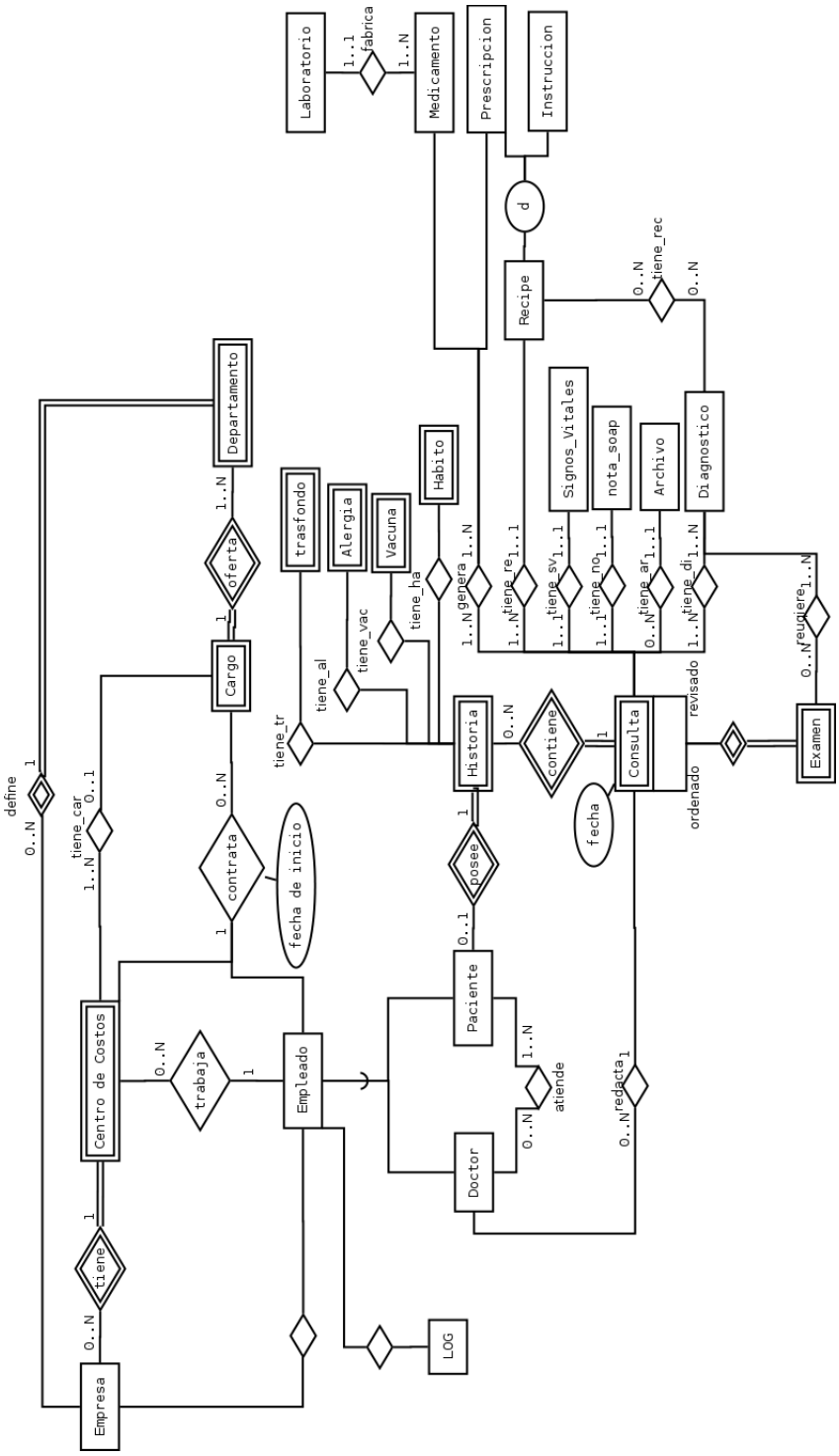

Figura A.1: Diagrama ER-E de la base de datos.

Apéndice B

Diagrama de Casos de Uso

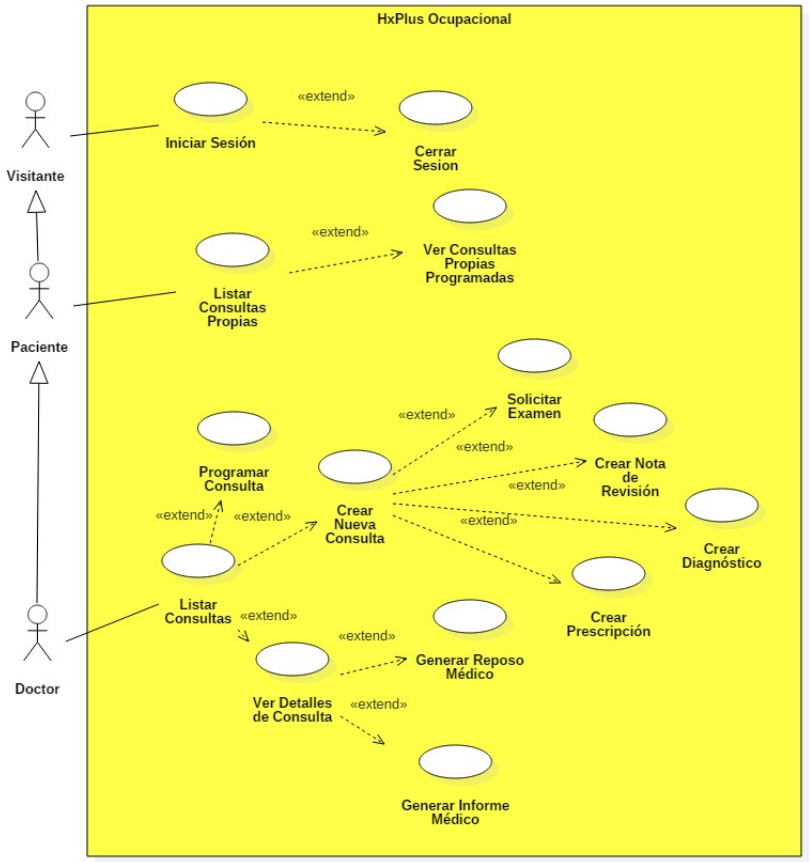

Figura B.1: Diagrama de Casos de uso implementados.

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/.

[32] *Code School a Pluralsight company*. URL: http://www.codeschoool.com.