UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

HXPLUS OCUPACIONAL.
SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS ORIENTADO A MEDICINA OCUPACIONAL

Presentado por:
Alejandro Tarazona
Realizado con la asesoría de:
Tutor Académico: Prof. Angela Di Serio
Tutor Industrial: Ing. Juan Albarrán

INFORME DE PASANTÍA

Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar por el título de
Ingeniero en Computación

Sartenejas, 11 de febrero de 2016

UNIVERSIDAD SIMÓN BOLÍVAR

DECANATO DE ESTUDIOS PROFESIONALES

COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

ACTA FINAL PASANTÍA LARGA

HXPLUS OCUPACIONAL.
SISTEMA DE GESTIÓN DE CONSULTAS MÉDICAS
ORIENTADO A MEDICINA OCUPACIONAL

Presentado por:

Alejandro Tarazona

Este Proyecto de Pasantías ha sido aprobado por el siguiente jurado examinador:

Ángela Di Serio

Juan Albarrán

PROFESOR 3

Sartenejas, 11 de febrero de 2016

Índice general

Índice de figuras

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 estpulado 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

Capítulo 2

Marco Teórico

En el presente capítulo se describen conceptos y patrones utilizados para el desarrollo del proyecto de patantía, 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 de negocio) la vista (Interfaz con el usuario u otros sistemas) y el controlador (Manejo de la informacion de negocio)[1].

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

1. **Modelo**
    - Almacenamiento de los datos.
    - Estado de la aplicación.
    - Recuperación de errores en los datos.
2. **Vista**
    - Presentación del modelo.
    - Puede acceder al modelo pero no cambiar su estado.
3. **Controlador**

- Reaccionar a las peticiones del cliente.
- Comunicar al modelo de las acciones ejecutadas.
- Direccionar a las vistas requeridas del lado del cliente.

2.2 Arquitectura Orientada a Servicios

SOA por sus siglas en inglés, *Service Oriented Architecture*. Es una filosofía de desarrollo la cual actúa como una guía de diseño y facilita el mismo orientado a la escalabilidad del sistema, esto es: facilita las actializaciones de uno o varios de los componentes MVC y minimiza el efecto que dicha actualización tiene sobre los demás.

Según Gartner[2] y González Quiroga[3], 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)[2]

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

En *Service Oriented Architecture*[4] definen SOA usando el poema de Saxe sobre los ciegos y el elfante.

”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

- Quien le toca la trompa cree que es una serpiente.
- Quien le toca los colmillos cree que son lanzas.

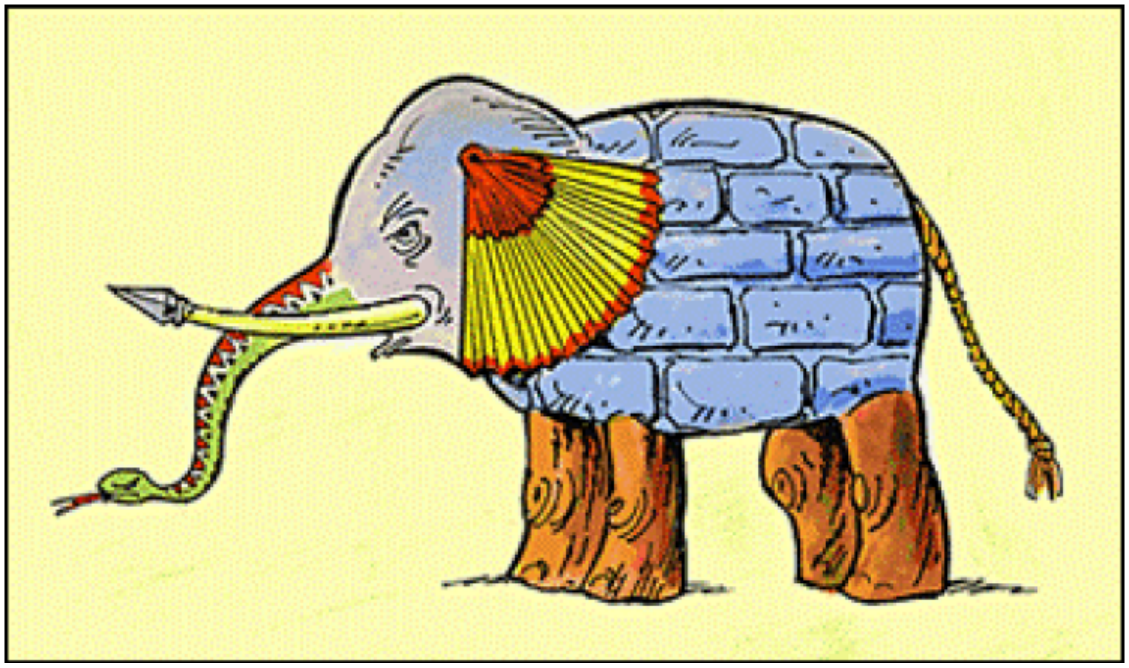

Figura 2.1: El elefante de Saxe

- 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."

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 filosifía 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 las (potenciamente) distintas vistas pueden interactuar con el controlador y este a su vez con el modelo.

2.3 Autenticación Basada en *Tokens*

Es una forma de autenticación ligera, que va de la mano con SOA, que usa *tokens*, o fichas, cifradas 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 decifrada y se verifican las credenciales del usuario en cuestión[5].

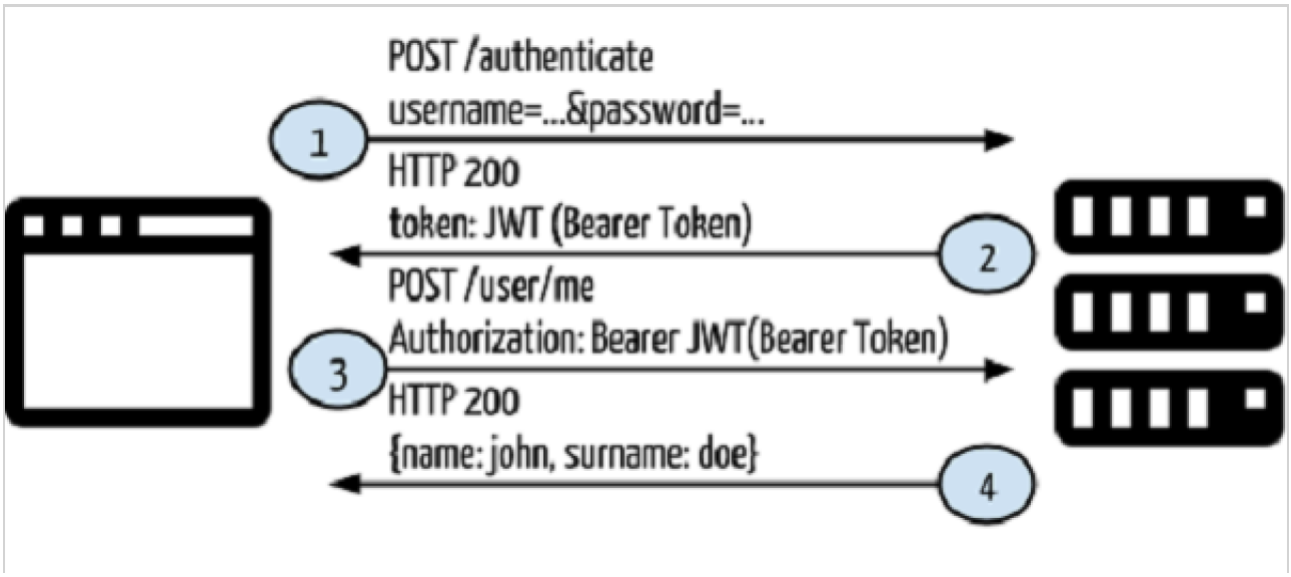

Figura 2.2: Autenticacion basada en fichas.

Entre los datos almacenados en las fichas pueden 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 decifrado de las fichas.

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, como muestra en la figura.

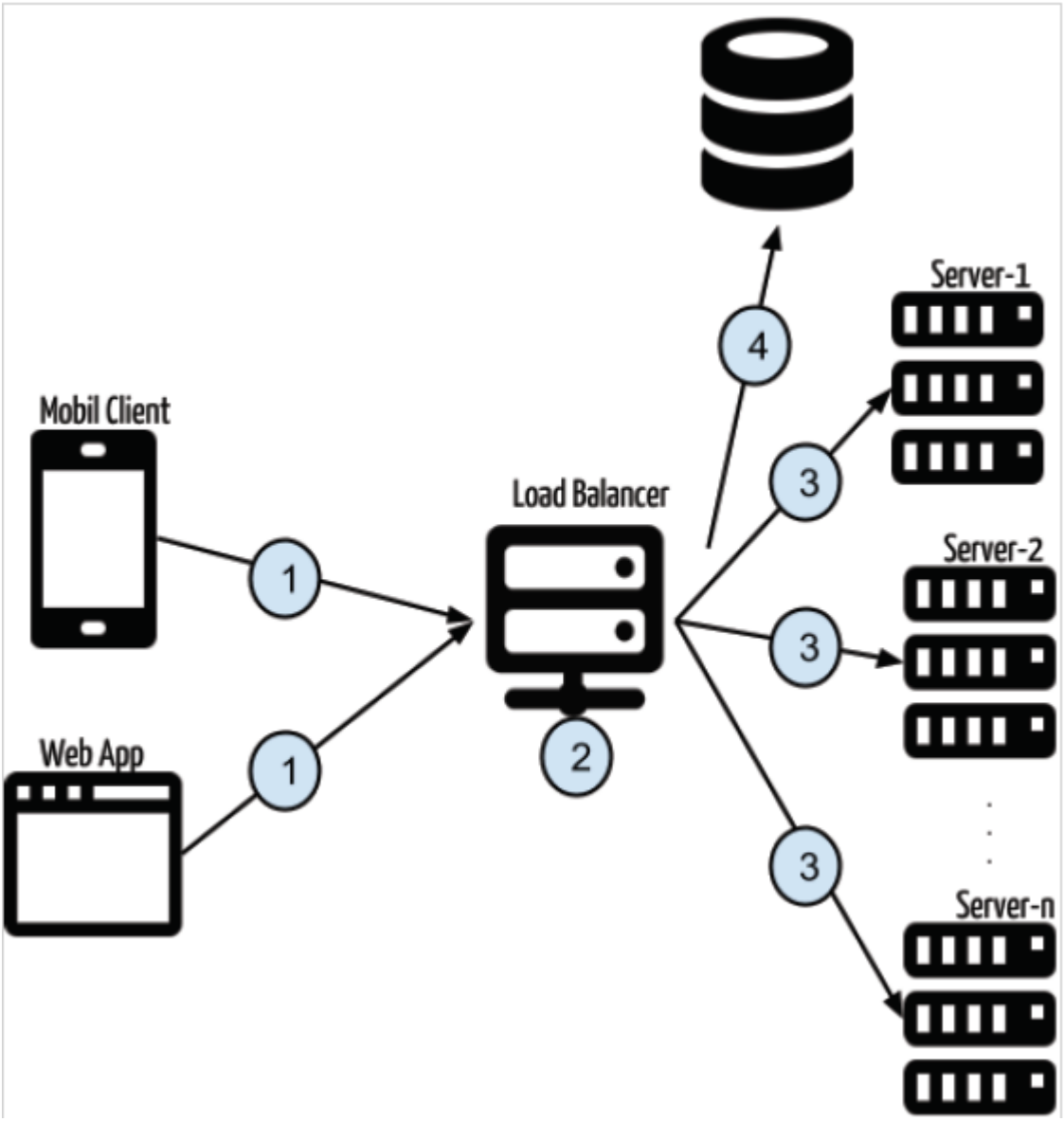

Figura 2.3: Portabilidad de la Autenticación Basada en Fichas.

Capítulo 3

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.

3.1 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.

3.1.1 Eclipse (Kepler)

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 que use el proyecto.

3.1.2 Java

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

3.1.3 JSON

Por *JavaScript Object Notation*, es un formato de intercambio de datos, ligero[6] que permite una sencilla comunicación entre la vista y el controlador.

"JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos[7]."

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.

3.1.4 JPA

Por "Java Persistence API", proporciona un modelo de persistencia basado en objetos de Java planos para hacer la correspondencia con las entidades en la base de datos[8]. En la práctica, hace transparentes las consultas y acciones realizadas sobre la base de datos.

3.1.5 Angular JS

Es un *framework* o conjunto de librerias orientada a facilitar el desarrollo web de aplicaciones dinámicas. "AngularJS le permite extender el vocabulario HTML para su aplicación"[9].

3.1.6 JavaScript

JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico[10].

Es un lenguaje de programación orientado a crear contenidos dinámicos para páginas web del lado del cliente. Tanto JSON como AngularJS, previamente mencionados, usan JavaScript en su codificación.

3.1.7 MySQL

Manejador de bases de datos, *open source*, ofrece alto rendimiento, eficiencia y seguridad en el almacenamiento y recuperación de datos[11]. Además de poseer librerías ampliamente usadas para Java las cuales ayudan al desarrollo debilmente acoplado del *back-end*.

3.1.8 SPRING

Framework para el desarrollo de aplicaciones y provee inversión de control, de código abierto para la plataforma Java. Permite integración con Hibernate, JPA y JSON[12].

3.1.9 Maven

Es un sistema centralizado para la gestión y compilado d proyectos desde una sección de control llamada "POM"[13]. Gestiona desde ahí la descarga de librerías necesarias y vínculos con las mismas.

3.1.10 Hibernate

Es un gestoor de persistencia que provee una implementación de JPA para hacer correspondencias de objetos planos de Java con la base de datos[14]. Gestiona las comunicaciones a nivel de nombres de entidades y atributos con su respectiva contraparte de Java, clases con sus atributos.

3.1.11 iText

Herramienta de generación de PDF dinámicos[15]. Proporciona una serie de comandos que permiten la generación de archivos PDF usando la información enviada a través de estos.

3.2 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.

3.2.1 Git

Sistema de manejo de control de versiones, *open source*, que provee la capacidad de crear tanto repositorios locales como remotos, ambos pueden estar sincronizados.

3.2.2 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.

3.2.3 Trello

Herramienta diseñada para la gestión de tareas usando listas o tablas. Cuenta con una interfaz intuitiva y de fácil aprendizaje.

Capítulo 4

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

4.1 Roles

4.1.1 Scrum Master

Juan Albarrán. Representante de Globinsoft S.A. y líder del proyecto *HxPlus Ocupacional*.

4.1.2 Product Owner

Juan Albarrán.

4.1.3 Development Team

Alejandro Tarazona. Pasante encargado del desarrollo del proyecto y autor del presente libro.

4.2 Eventos

Esta sección describe los eventos determinados por la metodología seleccionada y cómo fueron establecidos para

4.2.1 Sprint

Unidad mínima de desarrollo, usualmente determinada por una tarea corta o un período de tiempo pequeño. 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.

4.2.2 Sprint Planning

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.

4.2.3 Daily Sprint Meeting

Diariamente se realizaron reuniones para evaluar el avance durante el día en cuestión y la aclaratoria de dudas puntuales que fueron surgiendo en el desarrollo.

4.2.4 Sprint Review

Junto con las reuniones de *Sprint Planning* se realizó la evelauación o *Review* del Sprint de la semana en cuestión.

4.2.5 Sprint Retrospective

4.3 Artefactos

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

4.3.1 Sprint Backlog

Por cada Sprint se realizó una reunión de planificación y una de evaluación, las conclusiones, para cada sprint, se ven reflejadas en el *Sprint Blacklog*.

4.3.2 Product Backlog

La colección de todos los *Sprint Backlog*, genera el *Product Backlog*. En él se puede evaluar el desarrollo del proyecto de manera global.

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 las 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 A.
   - Descarga del *IDE* Eclipse

     Para el desarrollo se utilizo Eclipse Kepler, descargado de la página oficial de Eclipse[16]
   - 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.

- 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 *frontend*

- 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. Esto, en conjunto con las anotaciones de *SPRING* facilita la creación de las diferentes consultas y acciones a realizar en la base de datos.

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 vienen con métodos y procedimientos estipulados por JPA.

- 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 de cada una de las clases, dejando para futuros *Sprint* la tarea de modificar los mismos para cada una de las tareas, 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.

   - 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.

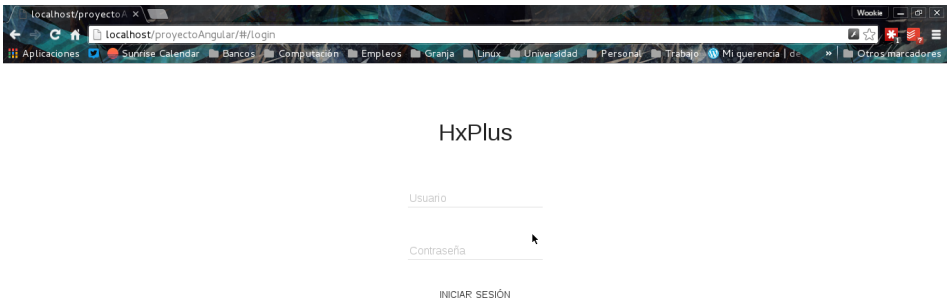

Figura 5.1: Pantalla de Autenticación de usuarios

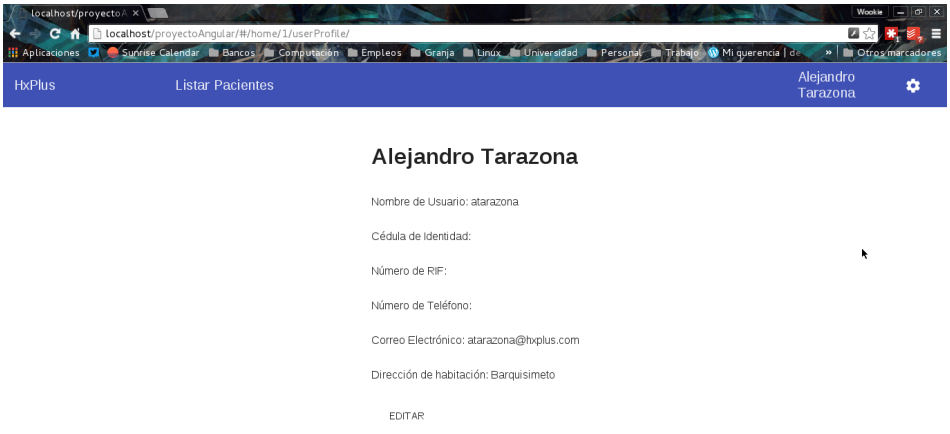

Figura 5.2: Pantalla de Revisión de Usuario.

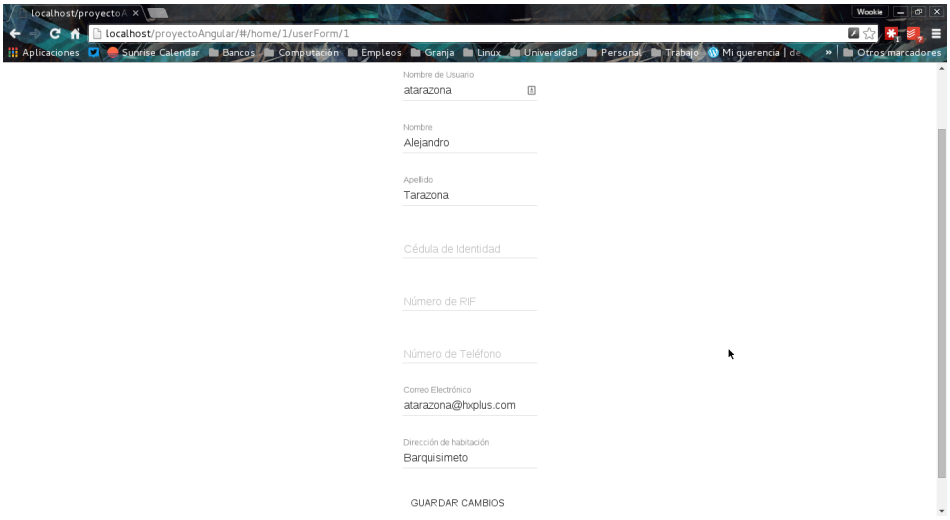

Figura 5.3: Pantalla de Edición de Usuario.

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 la foto del usuario y 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).

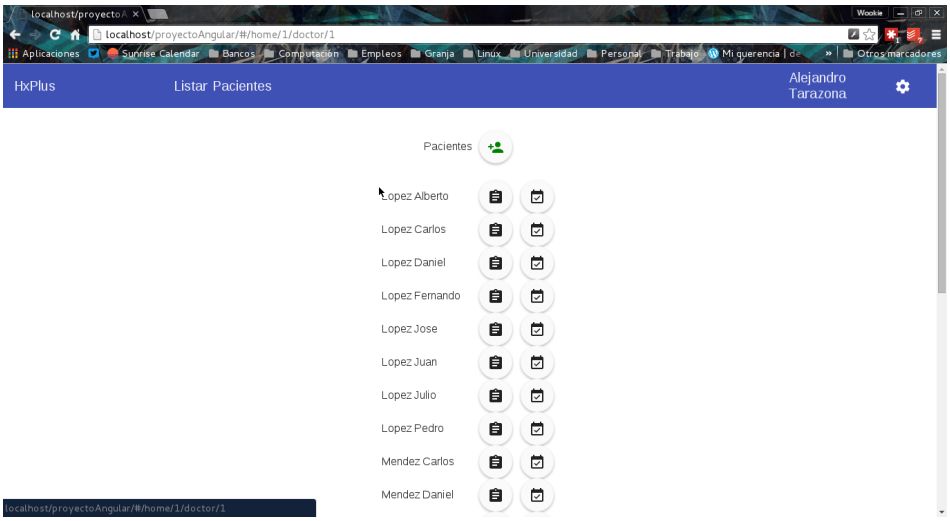

Figura 5.4: Lista de Pacientes del Médico

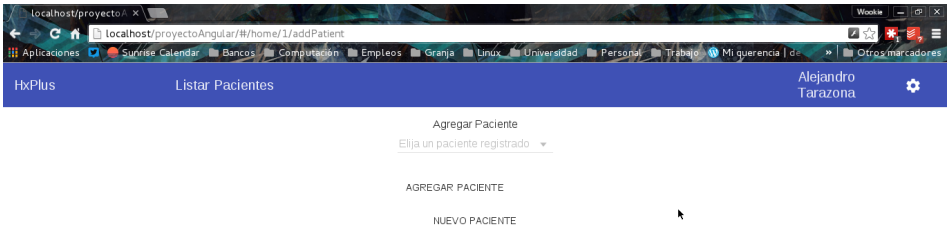

Figura 5.5: Vista de Agregar pacientes que ya han sido atendidos por algún médico.

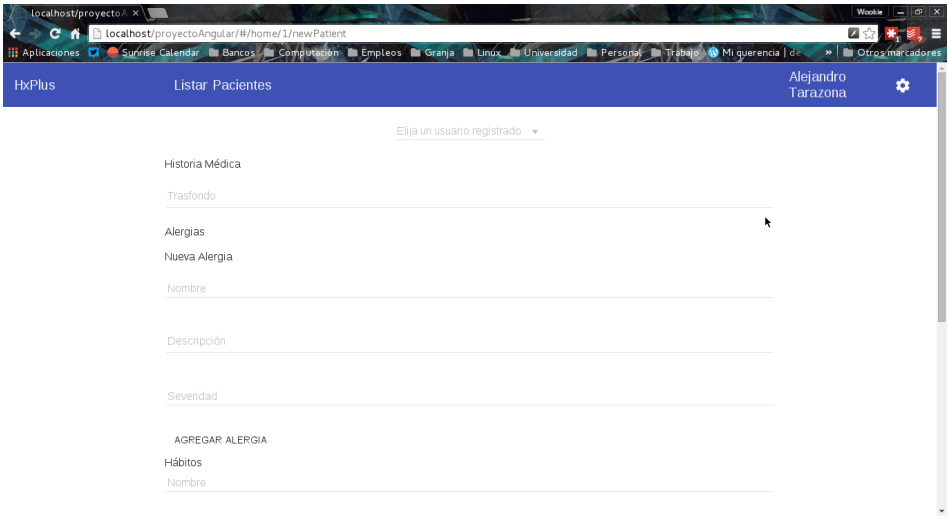

Figura 5.6: Vista de creación de nueva historia médica para un paciente nuevo.

– 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).

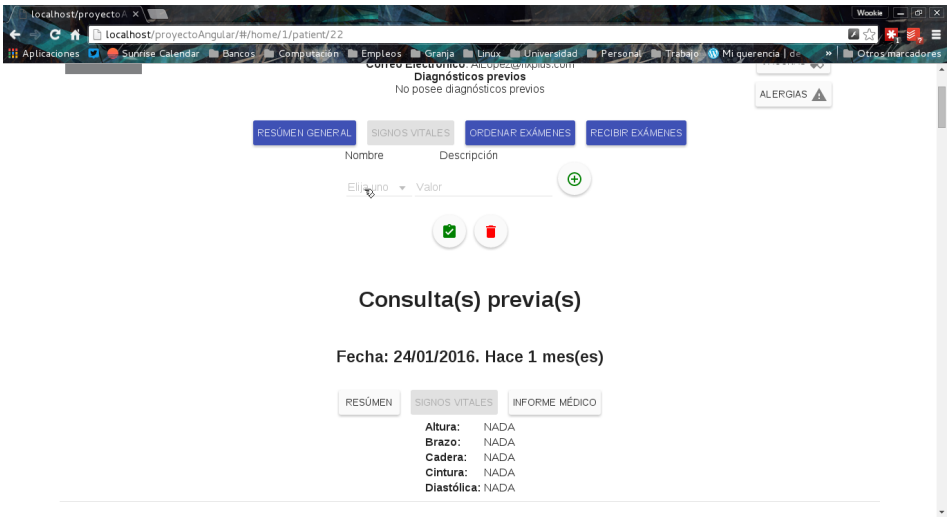

Figura 5.7: *Paarámetros Fisiológicos*

- 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).

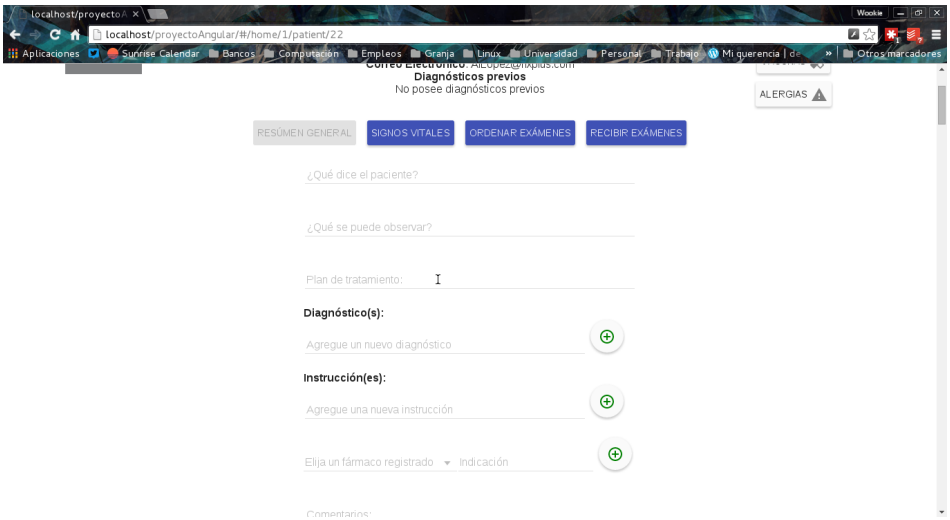

Figura 5.8: *Soap Note*

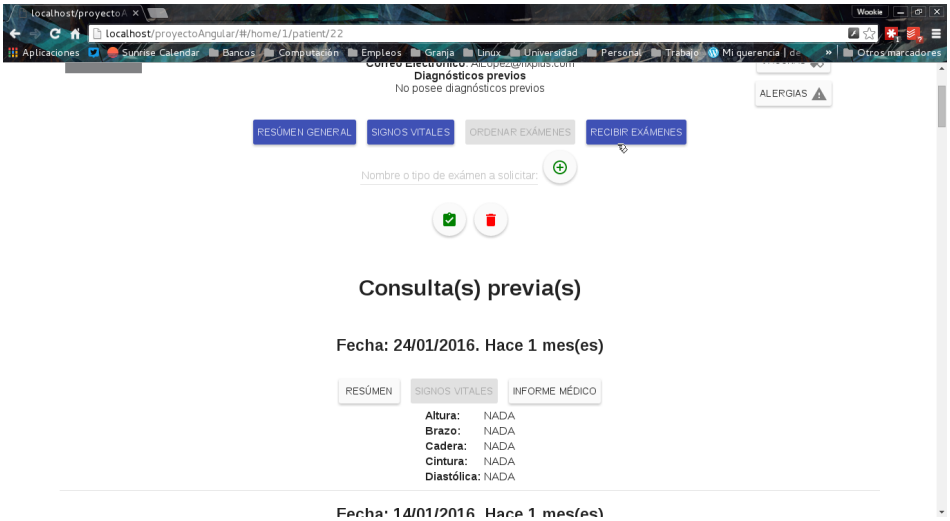

Figura 5.9: *Solicitud de Exámenes*

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.

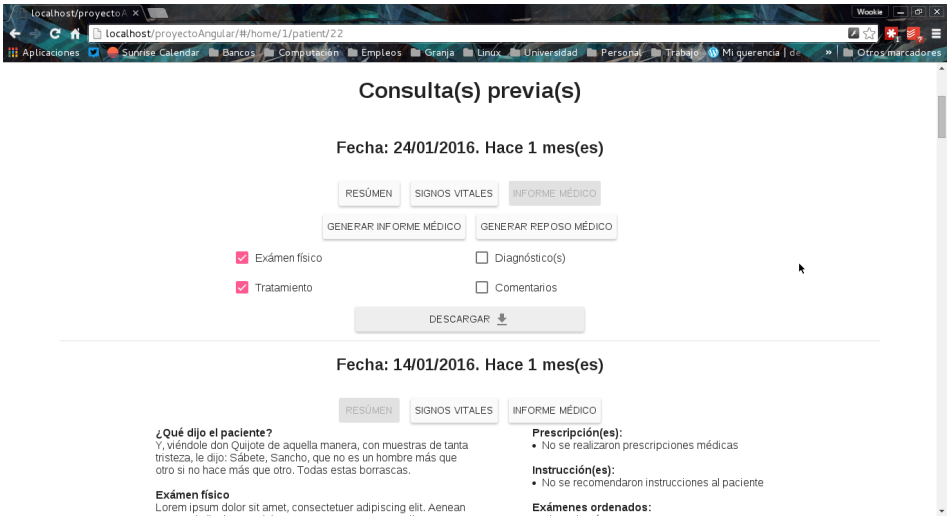

Figura 5.11: Sección de solicitud de informes médicos.

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.

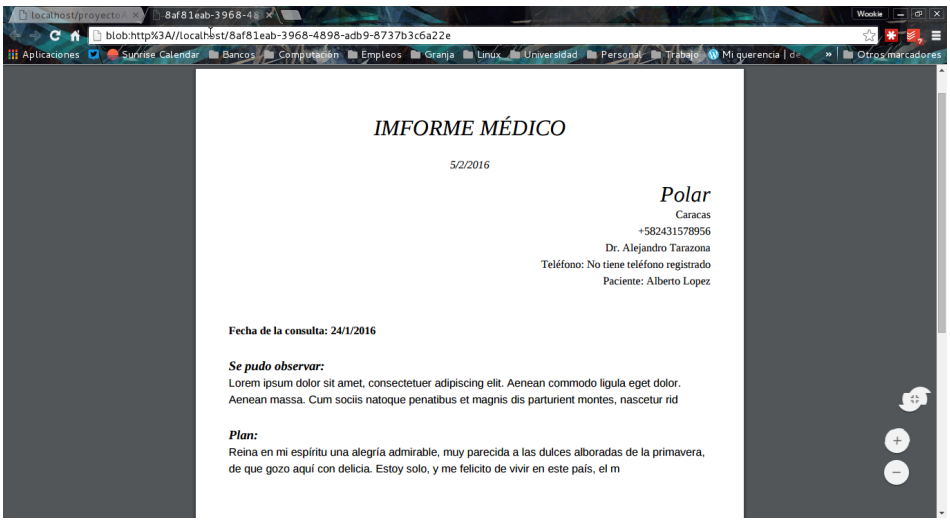

Figura 5.12: Informe modelo

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.

Apéndice A

Diagrama ER-E y Glosario de términos

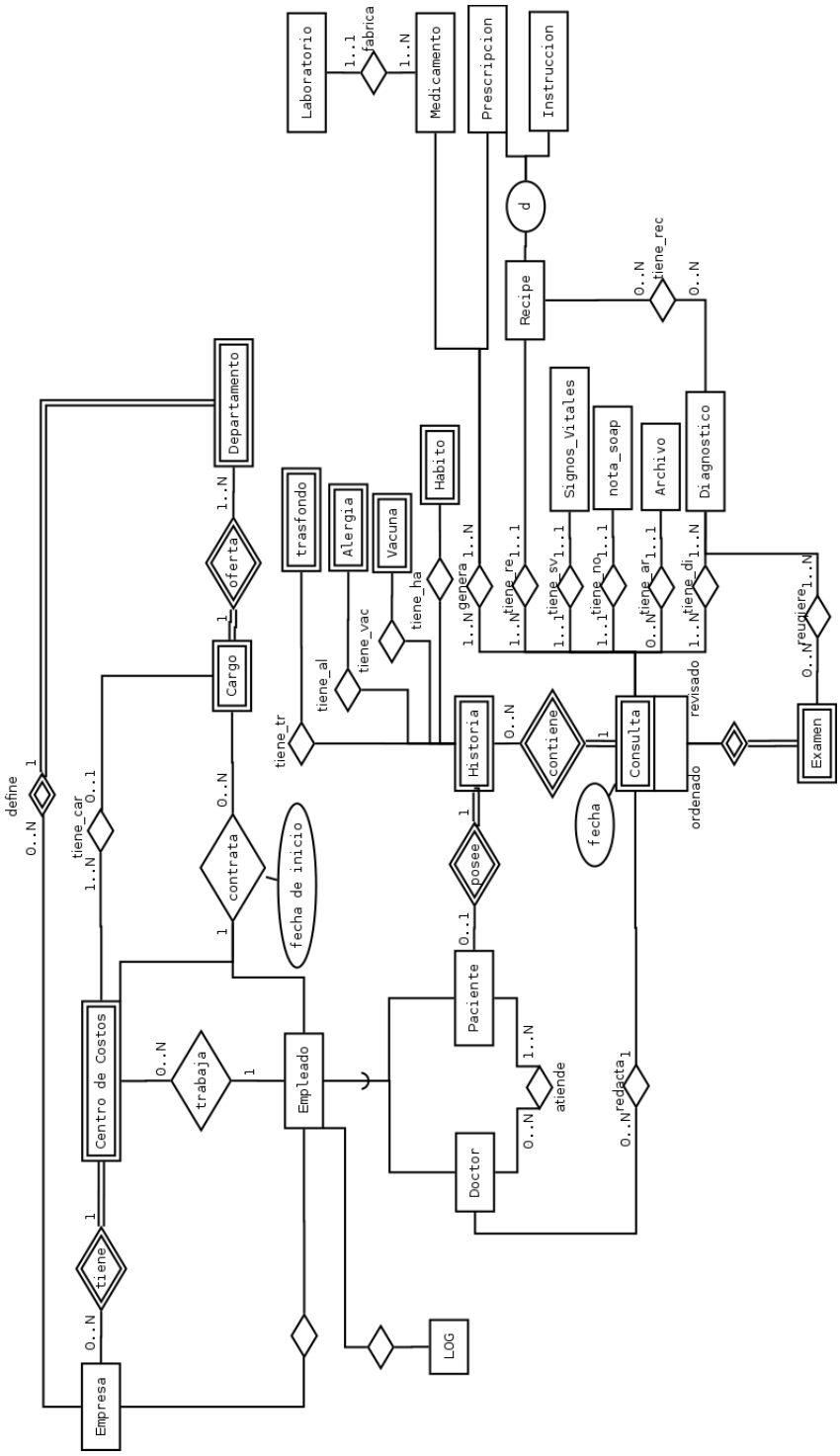

Figura A.1: Diagrama ER-E de la base de datos.

Bibliografía

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

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

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

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

[5] 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).

[6] 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).

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

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

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

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

[11] 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).

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

[13] Apache Maven Project. *Welcome to Apache Maven*. URL: `https://maven.apache.org/` (visitado 30-06-2015).

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

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

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