

A mi papá.

A mi tío Carlos y a mi abuelo León.

A mi abuela Alicia.

A mis amigos.

A mi amor, Patricia.

“Dreams come true”.

Agradecimientos

Siento que tengo mucho que agradecer pero no hay palabras suficientes para hacerlo.

Primeramente a mi tío Carlos y a mi abuelo León que me han apoyado desde que decidí salir del nido, sin su apoyo y sus consejos no estaría acá.

A mi papá, gracias por estar ahí en mis tormentas y por esas semillas de sabiduría que me diste. Gracias desde el fondo de mi corazón.

A Patricia, gracias por ser mi calma y compañía.

A mis amigos por estar ahí para escucharme y poder drenar las rabietas o frustraciones de cada trimestre y celebrar los logros también. ¡Salud!

Índice general

Introducción	2
1 Entorno Empresarial	4
1.1 Antecedentes	4
1.2 Misión	4
1.3 Visión	5
1.4 Estructura organizacional	5
2 Marco Teórico	7
2.1 Modelo Vista Controlador (MVC)	7
2.2 Arquitectura Orientada a Servicios	9
2.3 Autenticación Basada en <i>Tokens</i>	10
2.4 Inversión de Control	12
2.5 Inyección de Dependencias	12
3 Marco Metodológico	14
3.1 Roles	14
3.1.1 Product Owner	14
3.1.2 Scrum Master	15
3.1.3 Development Team	16

3.2 Eventos	17
3.2.1 Sprint	17
3.2.2 Sprint Planning	18
3.2.3 Daily Sprint Meeting	18
3.2.4 Sprint Review	19
3.2.5 Sprint Retrospective	19
3.3 Artefactos	19
3.3.1 Product Backlog	20
3.3.2 Sprint Backlog	20
4 Marco Tecnológico	21
4.1 Herramientas para el desarrollo de la aplicación	21
4.1.1 Eclipse	21
4.1.2 apache	22
4.1.3 Tomcat	22
4.1.4 Java	22
4.1.5 JSON	23
4.1.6 JPA	23
4.1.7 Angular JS	24
4.1.8 JavaScript	25
4.1.9 MySQL	25
4.1.10 SPRING	26
4.1.11 Maven	26
4.1.12 Hibernate	27
4.1.13 iText	28

4.2 Herramientas para el control de versiones y planificación	28
4.2.1 Git	28
4.2.2 GitHub	29
4.2.3 Trello	29
5 Desarrollo de la Aplicación	30
5.1 Fase de Inicio	30
5.1.1 Requerimientos Generales	31
5.1.2 Requerimientos Específicos	31
5.2 Fase de Implementación	36
5.2.1 Primer Sprint: Vista de Autenticación de Usuarios	37
5.2.2 Segundo Sprint: Vista de Lista de Usuarios	37
5.2.3 Tercer Sprint: Vista de Detalles de Usuario	38
5.2.4 Cuarto Sprint: Vista de Edición de Usuario	38
5.2.5 Quinto Sprint: Vista de Añadir Nuevo Paciente y Revisón de la Edición de Usuario	40
5.2.6 Sexto Sprint: Vista de Lista de Pacientes Atendidos	40
5.2.7 Septimo Sprint: Vista de Creación de Historia Médica	42
5.2.8 Octavo Sprint: Vista de Agregar Consulta Médica	43
5.2.9 Noveno Sprint: Vista de Historial de Consultas Médicas del Paciente	46
5.2.10 Décimo Sprint: Vista de Revisión de Consulta Médica	47
5.2.11 Décimo Primer Sprint: Vista de Generación de Reportes Médicos	48
5.3 Fase de Cierre	49
5.3.1 Dificultades generales encontradas	49
5.3.2 Resultados Generales	50
6 Conclusiones y Recomendaciones	51

Bibliografía	51
A Diagrama ER-E y Glosario de términos	54
B Diagrama de Casos de Uso	67

Índice de cuadros

5.1	Artefactos de Maven: Spring	35
5.2	Artefactos de Maven: JSON	36
5.3	Artefactos de Maven: Hibernate	36
5.4	Artefactos de Maven: Autenticación	37
A.1	Glosario de términos del diagrama ER-E	65

Índice de figuras

1.1	Estructura Organizacional de Globinsoft	6
2.1	Representación gráfica del patrón Modelo Vista Controlador[2]	8
2.2	El elefante de Saxe	10
2.3	Autenticacion basada en fichas.	12
2.4	Ejemplo de inyección de dependencias en controladores.	13
2.5	Ejemplo de inyección de dependencias en servicios.	13
3.1	Esquema de trabajo de SCRUM	17
4.1	Ejemplo de lista de empleados genérica en <i>JSON</i>	24
4.2	Artefacto de Maven y descripción de su contenido.	26
5.1	Pantalla de Autenticación de usuarios	37
5.2	Pantalla de Revisión de Usuario.	38
5.3	Pantalla de Edición de Usuario.	39
5.4	Vista de Agregar pacientes que ya han sido atendidos por algún médico.	41
5.5	Lista de Pacientes del Médico	41
5.6	Vista de creación de nueva historia médica para un paciente nuevo.	43
5.7	Resumen General	44
5.8	Signos Vitales	45

5.9	Solicitud de Examen Médico	45
5.10	Recepción de Examen Médico	46
5.11	Vista de Historial de Consultas Médicas del Paciente	47
5.12	Vista de Revisión de consulta. Resumen	48
5.13	Vista de Revisión de consulta. Signos Vitales	48
5.14	Vista de Generación de Reportes Médicos. Informe	49
5.15	Archivo de informe médico generado por el sistema	49
5.16	Vista de Generación de Reportes Médicos. Reposo	49
5.17	Archivo de reposo médico generado por el sistema	49
A.1	Diagrama ER-E de la base de datos.	66
B.1	Diagrama de Casos de uso implementados.	67

Listas 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
PDF	Portable Document Format
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 exposición 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 verisión, 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, brindando 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 farmacéuticas 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.

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 interreactúa 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.

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 siguientes 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.”

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.

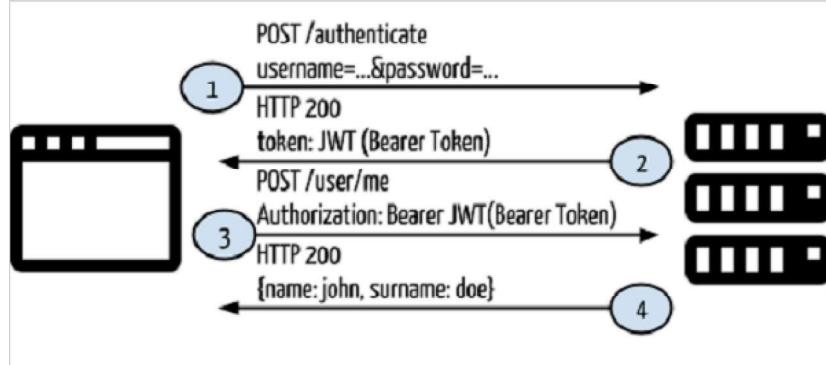


Figura 2.3: Autenticacion basada en fichas.

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 función 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 predefinidos, 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 probados¹.

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).

¹Con 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 procesos 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 involucrados 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 impedimentos 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 cuanto 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 eficiencia 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. Cierre

Tal y como se describe en el capítulo 5.

3.2.2 Sprint Planning

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

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

3.2.3 Daily Sprint Meeting

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

En este punto, Scrum, hace una diferenciación clave entre los elementos que están comprometidos (“*committed*”) 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 orden 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 desarrollo 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 realizaorón 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 retrospectiva. En su lugar se atendieron los inconvenientes, dudas y revisiones 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 móviles 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 Community 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 construye 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 encerrados 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 características 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 cliente. “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 bidireccional 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 realicen 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 google¹ 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 débilmente acoplado del *back-end*.

Para “HxPlus Ocupacional” se eligió 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>

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 sus 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.cfg.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 posterior 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 “raíz” y los archivos y directorios que lo conformen, incluye recursivamente los archivos y directorios dentro del directorio raíz, 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 raíz 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 raíces, dadas implicaciones de permisos de ejecución dentro del sistema operativo elegido. Estos repositorios fueron llamados “occupational” y “proyectoAngular” refiriéndose 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/atarazana> 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

El presente capítulo describe las etapas del desarrollo de la aplicación. Además se presentan las dificultades técnicas encontradas a lo largo del mismo y el resultado final alcanzado.

El desarrollo se dividió en tres fases:

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

Previo a la realización de estas fases, se llevó a cabo una fase para la descarga e inicialización de las herramientas a ser utilizadas en el desarrollo de la aplicación. Además se crearon las cuentas para el manejo de los repositorios del proyecto

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

5.1 Fase de Inicio

Objetivo General: Llevar a cabo el levantamiento de requerimientos e identificación de los módulos a desarrollar en el proyecto.

Objetivos Específicos:

- Levantamiento de los requerimientos del sistema.

- Identificación de los módulos que conformarán el sistema a desarrollar.
- 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:

5.1.1 Requerimientos Generales

- El sistema debe poseer mecanismos de autenticación para la verificación de identidad.
- El sistema debe proveer una plataforma de almacenamiento de historiales médicos con asociaciones entre la empresa en la que trabaja el paciente, la sucursal o “centro de costos”¹ en el cual trabaja el paciente al momento de la consulta.
- Mantener los historiales a través del tiempo, sin importar los cambios de “centro de costos”, que presente el paciente, ni el médico que lo atienda.
- En casos de algún cambio en el centro de costos donde trabaja el paciente, 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.

5.1.2 Requerimientos Específicos

- Usuarios:

Fue modelado como “Usuario” todo aquel empleado de una empresa registrada en el sistema. Para efectos del presente proyecto, los usuarios son los trabajadores de las empresas. Se hace una diferencia especial a dos tipos específicos de usuario: el doctor y el paciente (no excluyentes) que son la línea principal de trabajo del sistema. El sistema debe ser capaz de mantener una base de datos de usuarios con los atributos especificados en el Apéndice A.

La categoría 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

¹Se usa el término *Centro de Costos* para referirse a una sucursal o sede donde haya personal trabajando.

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.

Los “Pacientes” por otro lado representan a todo el personal que haya sido atendido y tenga su historia médica registrada en el sistema. Cabe destacar que un “Doctor” puede también ser paciente, aunque no puede ser su propio paciente.

- **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 hábitos 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 alérgeno² correspondiente.
4. Vacunas: Por último se 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:

²Según 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.

- (a) Subjective: Información subjetiva, comúnmente 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.
 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: En 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 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 ni éstos aparecerán en su lista).
- 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.

3. 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 creación 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 puesto de trabajo, sólo se discrimina por el puesto actual).

4. 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 ser consultado en el Apéndice A.

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 desde 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 (cuadro 5.3).

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 (versión final en Apéndice A.1) 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” y la gestión de doctores) están por fuera del alcance del proyecto, se utilizó una base de datos de pruebas generada exclusivamente para tal fin. También fueron generados datos de prueba para medicamentos y laboratorios, por las mismas razones expresadas con anterioridad.

A este Sprint, dada la poca complejidad que presenta, se le asignó una semana de duración pero, debido a los errores encontrados en la implementación, se le asignó una semana adicional de revisión adicional dentro del siguiente Sprint.

5.2.5 Quinto Sprint: Vista de Añadir Nuevo Paciente y Revisión de la Edición de Usuario

Se realizó la revisión de la “Vista de Edición de Usuario” debido a que presentaba sobrecarga de datos y en el caso de la no modificación del usuario se reescribían los datos con los mismos valores. El problema fue solventado agregando un detector de cambios con el cual sólo los datos modificados son enviados en el formulario, con esto se logra, no sólo reducir la cantidad de datos enviados al servidor si no que también se elimina la sobrecarga en caso de enviar un formulario con datos idénticos a los que fueron recibidos.

Por otro lado, en la “Vista de Añadir Nuevo Paciente”, el doctor tiene la opción de elegir entre:

1. Un paciente que ya posea historia médica en el sistema:

Se creó una lista de pacientes cuyos historiales médicos estén registrados en el sistema y serán agregados a la lista de pacientes del doctor.

2. Crear la historia médica de un usuario, ya registrado, que no la posea.

Para este caso se planificó agregar un botón de redirección a la “Vista de Crear Historia Médica”, descrita posteriormente, y que actualizaría la lista de pacientes con historia médica y dejaría en selección el nuevo paciente ya creado.

Para este Sprint la fase de pruebas de esta vista fue aceptación de la UI y revisión de la acción mencionada en el punto 1 dentro de la base de datos dado que no se cuenta con la vista necesaria y esta será desarrollada en el próximo Sprint.

Se dedicó, en total, dos semanas a este Sprint, dejando para futuros Sprint las pruebas mencionadas.

5.2.6 Sexto Sprint: Vista de Lista de Pacientes Atendidos

En esta vista se presentan por orden alfabético los pacientes que han sido atendidos por el doctor. Además, se presenta la opción de revisar su historial médico con lo cual se desplegará la “Vista de Historial de Consultas Médicas del Paciente” (ver punto 5.2.9).

5.2.7 Septimo Sprint: Vista de Creación de Historia Médica

En esta vista se presenta un formulario dinámico⁴ con los campos requeridos para la creación de la historia médica del paciente.

Los campos que admiten más de un sólo valor son:

- Alergias.

- Nombre
 - Descripción
 - Severidad

- Hábitos.

- Nombre
 - Frecuencia

- Vacunas.

- Nombre
 - Potencia

Se realizó la subdivisión de acuerdo a los avances realizados por Globinsoft S.A. y están encapsuladas de tal forma que se pueden agregar o eliminar de forma independiente una alergia, un hábito o una vacuna a la lista respectiva sin la necesidad de refrescar o recargar la página. Todos los datos recaudados son enviados una y sólo una vez al servidor una vez seleccionada la opción de envío del formulario.

Las enfermedades diagnosticadas previamente son un único valor dado que puede no ser precisa la información suministrada por el paciente.

Debido a la complejidad de la implementación de los formularios dinámicos, se asignaron tres semanas para la implementación exitosa de esta vista. Las pruebas fueron realizadas basándose en:

1. Respuesta a agregación o eliminación de valores en los campos dinámicos.
2. Carga y comunicación con el servidor.
3. Revisión de la creación de la historia médica (por verificar luego de la implementación del punto 5.2.9).

⁴Flexible en la cantidad de valores que puede aceptar un mismo campo. Puede verse como un ejemplo de un campo multivaluado de un diagrama ER-E

- Aceptación de UI.

<p>Fecha: 24/01/2016. Hace 4 mes(es)</p> <p>RESUMEN</p> <p>SÍGNOS VITALES</p> <p>INFORME MÉDICO</p> <p>Prescripción(es):</p> <ul style="list-style-type: none"> • No se realizaron prescripciones médicas <p>Instrucción(es):</p> <ul style="list-style-type: none"> • No se recomendaron instrucciones al paciente <p>Exámenes ordenados:</p> <ul style="list-style-type: none"> • No se ordenaron exámenes <p>Exámenes recibidos:</p> <ul style="list-style-type: none"> • No se recibieron exámenes
--

Figura 5.12: Vista de Revisión de consulta. Resumen

5.2.11 Décimo Primer Sprint: Vista de Generación de Reportes Médicos

Esta vista se anexa a la “Vista de Revisión de Consulta Médica” como un apartado final. En ella el doctor puede generar un reporte con los datos de la consulta dada, seleccionando los que desea plasmar en dicho reporte, y descargar un archivo en formato PDF para ser entregado de manera digital o impreso directamente. También puede optar por generar un reposo médico, indicando la cantidad de días asignados a dicho reposo y descargando el archivo.

En este punto el equipo de desarrollo precisó una semana de capacitación por la utilización de iText (ver punto 4.1.13) herramienta que fue la elegida para la generación de archivos PDF y que hasta el momento no había sido utilizada en el desarrollo.

Tomando en cuenta la semana de capacitación, este Sprint contó con una duración de dos semanas y las pruebas fueron basadas en:

- Aceptación de UI.
 - Correctitud de la información en el informe médico generado.
 - Correctitud de la información en los reposos médicos.
 - Aceptación del formato visual de ambas clases de archivos.

RESÚMEN	SIGNOS VITIALES	INFORME MÉDICO
Altura: NADA Brazo: NADA Cadera: NADA Cintura: NADA Diastólica: NADA		

Figura 5.13: Vista de Revisión de consulta.
Signos Vitales

Fecha: 24/01/2016. Hace 4 mes(es)

INFORME MÉDICO

Exámen físico Diagnóstico(s)
 Tratamiento Comentarios

DESCARGAR

Figura 5.14: Vista de Generación de Reportes Médicos. Informe

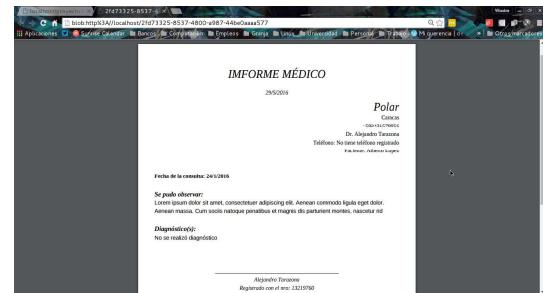


Figura 5.15: Archivo de informe médico generado por el sistema

Fecha: 24/01/2016. Hace 4 mes(es)

REPOSO MÉDICO

El empleado tiene un reposo de
10 días a partir del día de la consulta

DESCARGAR

Figura 5.16: Vista de Generación de Reportes Médicos. Reposo

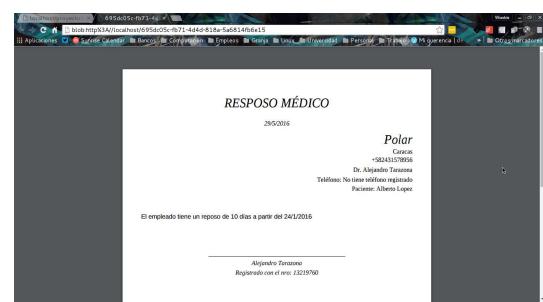


Figura 5.17: Archivo de reposo médico generado por el sistema

5.3 Fase de Cierre

5.3.1 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, Hibernate, 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 al principio.
- Falta de información 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 ámbito 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.

5.3.2 Resultados Generales

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

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

Capítulo 6

Conclusiones y Recomendaciones

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

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

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

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

Bibliografía

- [1] Ejemplos TIW. *Modelo-Vista-Contolador*. URL: <http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html> (visitado 05-12-2015).
- [2] Universidad de Alicante. *Modelo vista controlador (MVC)*. URL: <http://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>.
- [3] Gartner. *Predicts 2003: SOA is changing Software*. 2002.
- [4] María González Quiroga. “ESTUDIO DE ARQUITECTURAS DE REDES ORIENTADAS A SERVICIO”. PROYECTO FINAL DE CARRERA. Universidad Politécnica de Catalunya, 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-dependencias/> (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 community. *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>.

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 asignado 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.
		companynamne	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 compañía.	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, autogenerada, 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 alergia en el sistema.	BIGINTEGER.
		name	Nombre de la alergia. En caso de no conocerlo se indica a qué es alérgico 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	Identificador del doctor 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 vital 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 síntomas 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 imagen o archivo de formato portátil.	VARCHAR.
		filedata	Datos del archivo.	LONGBLOB.
DIAGNOSTIC	Parte de una soap-note. 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	BIGINTTEGER.
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.	BIGINTTEGER.
		idconsult	Consulta en que fue realizado la prescripción.	BIGINTTEGER.
		iddrug	Medicamento prescrito.	BIGINTTEGER.
		idindication	Indicación a seguir por parte del paciente para el medicamento prescrito.	BIGINTTEGER.

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

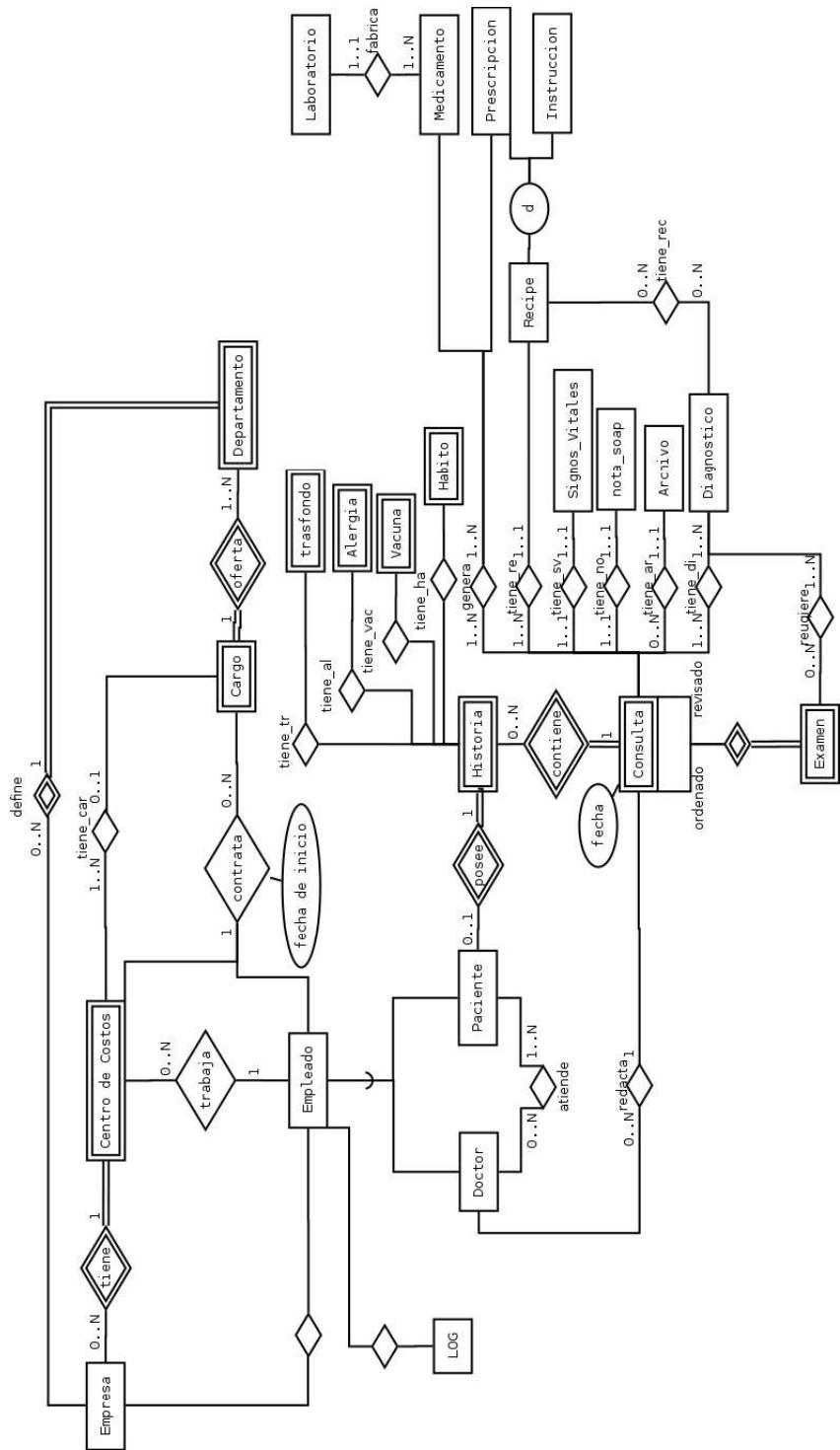


Figura A.1: Diagrama ER-E de la base de datos.