UNIVERSIDAD SIMÓN BOLÍVAR

DECANATO DE ESTUDIOS PROFESIONALES

COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN

ACTA FINAL PROYECTO DE GRADO

NOMBRE DEL PROYECTO DE GRADO

Presentado por:

AUTOR

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

_____

PROFESOR 1

_____

PROFESOR 2

_____

PROFESOR 3

Sartenejas, 16 de enero de 2016FECHA (dd/mm/aa)

Índice general

Índice de cuadros

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

1.3 Visión

1.4 Estructura organizacional

1.5 Cargo ocupado por el pasante

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*)

A parte del patrón MVC que garantiza el bajo acoplamiento del sistema, se tiene la filosofía de desarrollo orientada a servicios la cual actúa como una guía de desarrollo y facilita el mismo orientado a la escalabilidad del sistema, esto es: facilita las actializaciones de alguno 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.

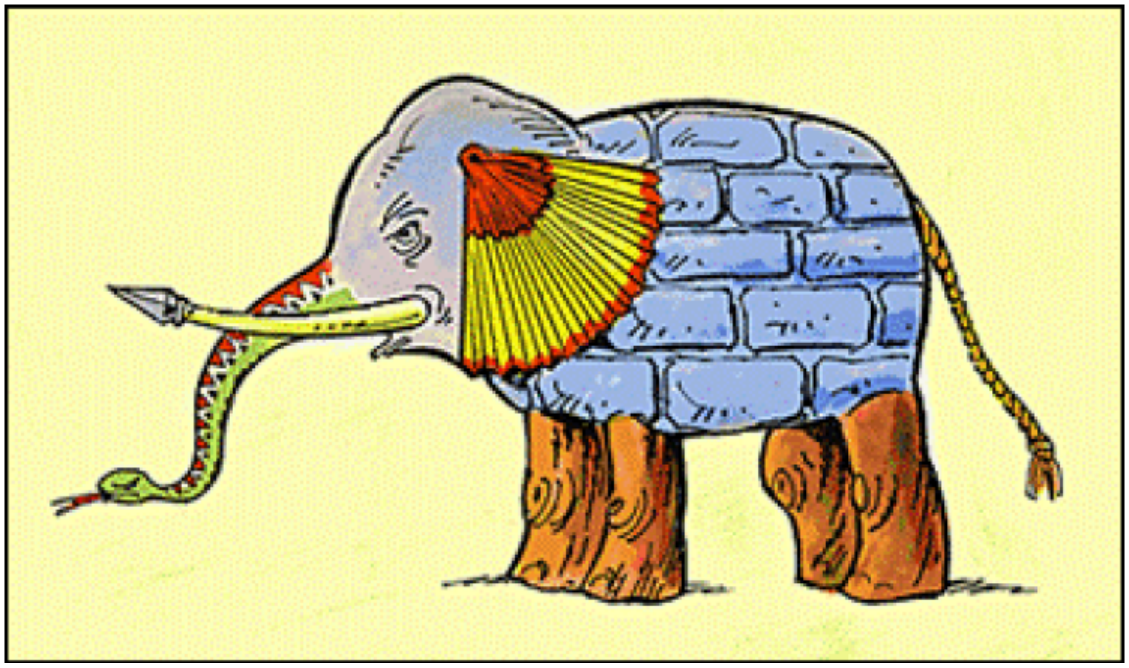

Figura 2.1: El elefante de Saxe

- 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.”

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.

Capítulo 3

Marco Tecnológico

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 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.

3.1.2 JSON

Por *JavaScript Object Notation*, es un formato de intercambio de datos, ligero[5] 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[6]."

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.

3.1.3 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"[7].

3.1.4 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[8].

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.

3.1.5 MySQL

Manejador de bases de datos, *open source*, ofrece alto rendimiento, eficiencia y seguridad en el almacenamiento y recuperación de datos[9]. Además de poseer librerías ampliamente usadas para Java las cuales ayudan al desarrollo debilmente acoplado del *back-end*.

3.1.6 SPRING

3.1.7 Maven

3.1.8 Hibernate

3.1.9 iText

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. En el caso del proyecto referido en el presente libro, fue dividido en 4 listas: ”Pendiente”, ”Impedido”, ”En desarrollo” y ”Tarminado”; para referirse al estado actual de las tareas en cada una de las listas.

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

5.1 Fase de Preparación

5.1.1 Primer Sprint

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

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 del proyecto en Trello para la gestión del proyecto.
- Creación de repositorios, tanto local como remoto, para el almacenamiento de la aplicación.
- Diseño de la Base de Datos.
- Configuración de Spring e Hibernate.

5.2.2 Segundo Sprint: Configuración de la aplicación

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, Assets, Plan*)
  - Diagnóstico
  - Examen
  - Consulta
  - Récipe
  - Medicamento
  - Laboratorio

- Crear los repositorios. Interfaces que usa *Spring* para manejar la conexión con la base de datos.
- 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

1. Objetivos
   (a) Configuración de *SPRING* e *Hibernate*
2. Actividades
   - Configuración de las características de *SPRING* para trabajar con anotaciones.
   - Descargar y configurar las librerías internas de la herramienta.
   - Descargar la librería JSON para la comunicación con el *frontend*.
   - Descargar y configurar *Hibernate* para la comunicación con la base de datos.

5.2.4 Cuarto Sprint: Consultas de Usuarios y Vista del doctor

1. Objetivos
   - Creación de clases, repositorios y servicios básicos.
2. Actividades

5.2.5 Quinto Sprint: Consulta del Paciente

1. Objetivos
   - Creación de controladores básicos.
2. Actividades

5.2.6 Sexto Sprint: Generar informes

1. Objetivos
    - Autenticación e implementación de la autenticación basada en tokens.
2. Actividades

5.2.7 Septimo Sprint: Generar informes

1. Objetivos
    - Creación, consultas, edición y eliminación de usuarios.
2. Actividades

5.2.8 Octavo Sprint: Generar informes

1. Objetivos
    - Consultas del módulo de usuarios.
2. Actividades

5.3 Fase de Cierre

5.3.1 Décimo Sprint

1. Objetivos
2. Actividades

5.3.2 Sprint

1. Objetivos
2. Actividades

5.3.3 Dificultades generales encontradas

5.3.4 Resultados

Conclusiones y Recomendaciones

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] 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).

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

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

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

[9] 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).