

“Año del Buen Servicio al Ciudadano”

UNIVERSIDAD ANDINA NÉSTOR CÁCERES VELÁSQUEZ  
FACULTAD DE INGENIERÍA DE SISTEMAS  
C.A.P INGENIERÍA DE SISTEMAS



**INFORME DE PRÁCTICAS PRE-PROFESIONALES**

Prácticas realizadas en:  
**WELL DONE SOLUTIONS SAC**

Elaborado por:  
**MAMANI PAUCAR, wilder**

**PARA OPTAR EL GRADO DE BACHILLER EN INGENIERÍA DE SISTEMAS**

JULIACA - PERU  
2017

*A mi Familia por enseñarme el camino correcto.*

# Índice de figuras

---

5.1.	CU-1 Seguridad . . . . .	13
5.2.	CU-2 Creación de Usuarios . . . . .	14
5.3.	CU-3 Casos de Uso primordiales . . . . .	14
5.4.	DC-A Diagrama de clases . . . . .	15
5.5.	DC-B Diagrama de clases completo . . . . .	16
5.6.	DP-1 Diagrama BPM . . . . .	17
5.7.	DA-1 Arquitectura del Sistema . . . . .	18
5.8.	DA-2 Arquitectura del Sistema (Diagrama de secuencia) . . . . .	18
5.9.	Arquitectura de aplicaciones web modernas . . . . .	20
5.10.	Sitio Web de Bootstrap . . . . .	21
5.11.	Carga de página tradicional . . . . .	22
5.12.	Carga de página SPA . . . . .	23
5.13.	Componentes de AngularJS . . . . .	24
5.14.	Arquitectura Modelo-Vista-Controlador . . . . .	26
5.15.	Componentes de NodeJS . . . . .	27
5.16.	Arquitectura de Spring Framework . . . . .	28
5.17.	Asistente de Spring Initializr . . . . .	30
5.18.	Estructura de un proyecto Spring Boot . . . . .	30
7.1.	Pantalla contribución backend . . . . .	34
7.2.	Pantalla contribución frontend . . . . .	35
7.3.	Pantalla login . . . . .	36
7.4.	Pantalla nueva empresa . . . . .	36
7.5.	Pantalla nuevo paciente . . . . .	37
7.6.	Pantalla exámenes a realizar . . . . .	37
7.7.	Pantalla examen audiometría . . . . .	38
7.8.	Pantalla antecedentes ocupacionales . . . . .	38
7.9.	Pantalla backend código principal . . . . .	39
7.10.	Pantalla backend código controlador audiometría . . . . .	39
7.11.	Pantalla backend código entidad contacto . . . . .	40
7.12.	Pantalla frontend código principal . . . . .	40
7.13.	Pantalla frontend código controlador espirometría . . . . .	41

7.14. Pantalla frontend código comunicación con el backend . . . . .	41
7.15. Pantalla reporte audiometría . . . . .	42
7.16. Pantalla reporte oit . . . . .	43

# Índice de cuadros

---

5.1. Resumen de componentes AngularJS . . . . .	25
5.2. Módulos de Spring . . . . .	29
5.3. Proyectos del ecosistema Spring . . . . .	31

# Índice General

---

<b>1. Presentación</b>	<b>1</b>
1.1. Objetivo del Informe . . . . .	1
1.2. Periodo de Prácticas Pre Profesionales . . . . .	1
1.3. Institución y Área de Trabajo . . . . .	1
1.4. Funciones del Área de Trabajo . . . . .	1
1.4.1. Planificar, organizar, diseñar e implementar software a medida . . . . .	1
<b>2. Datos Generales del Practicante</b>	<b>2</b>
<b>3. Aspectos Generales de la Empresa</b>	<b>3</b>
3.1. Descripción . . . . .	3
3.2. Ubicación . . . . .	3
3.3. Teléfono . . . . .	3
3.4. Portal Web . . . . .	3
3.5. Actividades que Realiza . . . . .	3
3.5.1. Desarrollo de software . . . . .	3
3.5.2. Cursos presenciales . . . . .	4
3.5.3. Creación de portales corporativos . . . . .	4
<b>4. Actividades Realizadas</b>	<b>5</b>
4.1. Obtención de requisitos . . . . .	5
4.2. Análisis de requisitos . . . . .	6
4.3. Diseño de software . . . . .	6
4.4. Desarrollo Frontend y Backend . . . . .	6
4.5. Despliegue de sistemas en la Nube (vps) . . . . .	7
<b>5. Descripción del Proyecto Realizado</b>	<b>8</b>
5.1. Objetivo . . . . .	8
5.2. Justificación . . . . .	8
5.3. Planificación . . . . .	8
5.4. Metodología . . . . .	8
5.5. Equipo de Desarrollo . . . . .	9

5.5.1.	Gerente de Proyecto . . . . .	9
5.5.2.	Jefe de Proyecto . . . . .	9
5.5.3.	Analista de Sistemas . . . . .	9
5.5.4.	Arquitecto de Software . . . . .	9
5.5.5.	Analista Programador . . . . .	10
5.5.6.	Programador . . . . .	11
5.5.7.	Administrador de Bases de Datos . . . . .	11
5.5.8.	Diseñador . . . . .	11
5.5.9.	Documentador . . . . .	12
5.6.	Análisis del Sistema . . . . .	13
5.7.	Diseño del Sistema . . . . .	15
5.8.	Arquitectura del Sistema . . . . .	18
5.9.	Desarrollo del Sistema . . . . .	19
5.9.1.	Arquitectura de Aplicaciones Web Modernas . . . . .	19
5.9.2.	Bootstrap . . . . .	20
5.9.3.	AngularJS . . . . .	21
5.9.4.	NodeJS y ExpressJS . . . . .	27
5.9.5.	Spring y Spring Boot . . . . .	28
<b>6.</b>	<b>Conclusiones y Recomendaciones</b>	<b>32</b>
6.1.	Conclusiones . . . . .	32
6.2.	Recomendaciones . . . . .	32
<b>7.</b>	<b>Anexos</b>	<b>33</b>

---

# Presentación

---

## 1.1. Objetivo del Informe

Aplicar los conocimientos adquiridos durante los cinco años de formación profesional dentro de la carrera académico profesional de **Ingeniería de Sistemas** de la “**Universidad Andina Néstor Cáceres Velásquez**” de la ciudad de Juliaca.

## 1.2. Periodo de Prácticas Pre Profesionales

Periodo de prácticas realizadas: **1 de septiembre del 2015 - 29 de febrero del 2016.**

## 1.3. Institución y Área de Trabajo

Las prácticas se realizaron en la empresa “**Well Done Solutions SAC**” en el área de **Desarrollo de Software**.

## 1.4. Funciones del Área de Trabajo

El Área de trabajo se dedica netamente al desarrollo de software a medida.

### 1.4.1. Planificar, organizar, diseñar e implementar software a medida

La mayoría de las empresas aún usan sistemas de información manuales, siendo una limitante de cara a aquellas empresas que hacen uso intesivo de sistemas automatizados; a esto se suma la necesidad de agilizar procesos de negocios y entregar información precisa a usuarios y clientes. Brindamos soluciones de software a medida, haciendo uso de tecnologías emergentes en el mundo del desarrollo de software.

---

# Datos Generales del Practicante

---

<b>APELLIDOS Y NOMBRES</b>	: Mamani Paucar, wilder
<b>DOMICILIO ACTUAL</b>	: Jr. Mariano Melgar #657
<b>DNI</b>	: 73529058
<b>CELULAR</b>	: 951-503-009
<b>ÁREA DE TRABAJO</b>	: Desarrollo de Software
<b>CARGO</b>	: Analista Programador
<b>EMAIL</b>	: w11ld33r@gmail.com
<b>GITHUB</b>	: <a href="https://www.github.com/w11ld33r">https://www.github.com/w11ld33r</a>

---

# Aspectos Generales de la Empresa

---

## 3.1. Descripción

- **WELL DONE SOLUTIONS SAC** es una empresa privada ubicada en la ciudad de Juliaca, Departamento de Puno, Perú
- Inició su funcionamiento el 5 de Octubre del año 2014

## 3.2. Ubicación

Pasaje 1<sup>ero</sup> de Mayo 112 - Oficina 301

## 3.3. Teléfono

051 33-6975

## 3.4. Portal Web

<http://www.wdsolutions.pe>

## 3.5. Actividades que Realiza

La empresa “**Well Done Solutions SAC**” se dedica a brindar asesoría, desarrollo de software, cursos talleres, mantenimiento de servidores, entre otros servicios relacionados a la tecnología.

### 3.5.1. Desarrollo de software

Como se mencionó anteriormente, la empresa brinda soluciones de software a medida, haciendo uso de tecnologías emergentes en el mundo del desarrollo de software.

### **3.5.2. Cursos presenciales**

La mayoría de jóvenes tanto recién egresados como autodidactas, muchos de ellos solo aprendieron a construir programas sencillos y no saben cómo enfrentarse a proyectos reales. La empresa “**Well Done Solutions SAC**” dicta cursos presenciales al público en general, quienes quieran aprender o reforzar conocimientos de las últimas tendencias en lenguajes de programación o arquitectura de software, entre otros.

### **3.5.3. Creación de portales corporativos**

Tener presencia en internet es la mejor forma de darse a conocer ante mundo. Empresas privadas como públicas buscan tener dicha presencia para dar a conocer los productos o servicios que brindan. “**Well Done Solutions SAC**” brinda servicios de creación de portales corporativos a medida.

---

# Actividades Realizadas

---

En el área de desarrollo de software colaboré en las distintas etapas del desarrollo del sistema que se desarrolló. Reuniones con usuarios para la obtención de requisitos, análisis de requisitos, diseño del sistema, desarrollo del sistema y por último puesta en producción del sistema.

A lo largo del periodo de prácticas utilice herramientas que ayudan a ser productivos, tales son: Sistemas de control de versiones (Git, Bitbucket, Github, Gitlab), Entornos de Desarrollo Integrado (Eclipse, Netbeans, IntelliJ IDEA, Spring Tool Suite), editores de texto (Sublime Text, Atom, Visual Studio Code, Brackets), etc.

## 4.1. Obtención de requisitos

La obtención de requisitos es la etapa inicial y fundamental de cualquier tipo de proyecto de software que se quiera realizar. Se enfocan sólo en la visión del sistema que tiene el usuario. La funcionalidad del sistema, la interacción entre el usuario y el sistema, los errores que el sistema puede detectar y manejar son parte de los requisitos [Bruedgge and Dutoit, 2002].

La obtención de requerimientos incluye las siguientes actividades:

- **Identificación de actores.** Durante esta actividad se identifican los diferentes tipos de usuario que el sistema soportará.
- **Identificación de escenarios.** Aquí se observan a los futuros usuarios y se desarrollan un conjunto de escenarios posibles para las distintas funcionalidades del sistema.
- **Identificación de casos de uso.** Una vez de acuerdo el usuario con el equipo de desarrollo, se abstraen los escenarios en casos de uso.

Hubo reuniones con los usuarios, escuchándolos activamente, proponiendo ideas, debatiendo, descartando casos de uso innecesarios. Para llegar a un acuerdo. Dentro de esta labor el equipo tenía un panorama general de todo el sistema y a su vez en cada reunión se tenía ideas y modelos de funcionalidades listas para implementar; y presentarlos en la próxima reunión para ser aprobado por los interesados del proyecto.

Esta labor se me fue encomendada a medida que me iba familiarizando con el equipo, ya que es una parte crítica de un proyecto de software.

## 4.2. Análisis de requisitos

El análisis de requisitos se enfoca en la producción de un modelo del sistema. El análisis de requisitos le proporciona al diseñador del sistema una representación de información y función [Pressman, 2006]. Aunque puede ser que el modelo de análisis no sea comprensible para los usuarios, ayuda a que los diseñadores del sistema verifiquen la especificación del sistema producida durante la obtención de requisitos.

El análisis de requisitos produce tres modelos individuales:

- **Modelo funcional.** Representado por casos de uso y escenarios.
- **Modelo de objetos de análisis.** Representado por diagramas de clase y objetos.
- **Modelo dinámico.** Representado por diagramas de estado y de secuencia.

Fui partícipe de esta actividad progresivamente, opinaba de acuerdo a los conocimientos que tenía, pero el equipo siempre estuvo dispuesto a ayudarme, con lo cual despejaba dudas e inquietudes.

## 4.3. Diseño de software

El Diseño de software es un proceso mediante el cual los requisitos se convierten en un plano para construir el software. Al inicio el plano representa una visión general del software. A medida que se va avanzando el proceso, se van representando partes del software a profundidad.

Parte del diseño de software se encarga de describir la descomposición en subsistemas desde el punto de vista de responsabilidades, dependencias, flujo de control, control de acceso y almacenamiento de datos.

Forme parte de esta labor activamente, ya que tenía un buen conocimiento acerca de patrones de diseño, bases de datos, descomposición de sistemas, etc.

## 4.4. Desarrollo Frontend y Backend

La parte de programación, es donde utilizamos todos los diseños y diagramas para empezar a codificar en algún lenguaje de programación. El desarrollo frontend y backend requiere de habilidades y conceptos como: conocimientos de algoritmos, estructuras de datos, pruebas unitarias,

pruebas end-to-end, integración continua, etc.

Ingresé a la empresa desarrollando únicamente en el lado del frontend, conforme avanzaba en conocimientos, comencé a desarrollar frontend y backend, los cuales me permitieron conocer las tecnologías actuales que dominan el mercado del desarrollo de software empresarial.

## 4.5. Despliegue de sistemas en la Nube (vps)

En esta última actividad se pone en producción el sistema en su totalidad. Servicios como *Google Cloud Platform*, *Digital Ocean*, *Amazon Web Services*, *etc*, requieren de habilidades de administración de servidores que ofrecen estas empresas. Distino a un hosting compartido, las plataformas como *Google Cloud Platform* brindan un espacio de almacenamiento dedicado exclusivo y la libertad de configurar el servidor de acuerdo a necesidades específicas. También permite escalar a medida que el sistema crece.

Gracias a los conocimientos en servidores linux pude asumir esta tarea, la de configurar, instalar paquetes y dejar todo listo para el funcionamiento correcto en la nube.

*Conforme el proyecto avanzaba, pude desenvolverme adecuadamente en todas las actividades mencionadas.*

---

# **Descripción del Proyecto Realizado**

---

Durante el periodo de prácticas se desarrolló un Sistema de Salud Ocupacional (basado en web), realizando un trabajo en equipo; por lo que el practicante estaba a cargo del desarrollo de ciertas funcionalidades del proyecto.

## **5.1. Objetivo**

Análisis, Diseño e Implementación de un Sistema de Salud Ocupacional para la Clínica Del Valle de la ciudad de Juliaca.

## **5.2. Justificación**

El sistema es necesario para la Clínica del Valle, porque les permitirá ahorrar tiempo y recursos, como también reducirá la tasa de errores; es necesario para los pacientes, porque se les brindará un servicio más rápido y con resultados precisos.

## **5.3. Planificación**

La construcción del sistema tuvo una duración aproximada de seis meses. La planificación era flexible en cuanto a reuniones (cada dos semanas) con el cliente. Se llevó a cabo una primera reunión con el cliente, quién requería las funcionalidades primordiales/urgentes del sistema. Pasada las dos semanas se tenía un producto mínimo viable y funcional. Las iteraciones nos permitía ir añadiendo más funcionalidades.

## **5.4. Metodología**

Se usó una metodología ágil apoyado en el conjunto de ideas que nos brinda “kanban” y el Análisis y Diseño Orientado a Objetos.

## 5.5. Equipo de Desarrollo

### 5.5.1. Gerente de Proyecto

- El Gerente de proyecto coordina con el usuario líder
- Monitorea el desempeño de los recursos y toma acciones correctivas
- Mantiene informado al usuario líder, actúa como enlace entre el personal del proyecto y otros tomadores de decisiones

### 5.5.2. Jefe de Proyecto

El jefe de proyecto asigna los recursos, gestiona las prioridades, coordina las interacciones con los clientes y usuarios, y mantiene al equipo del proyecto enfocado en los objetivos. El jefe de proyecto también establece un conjunto de prácticas que aseguran la integridad y calidad de los artefactos del proyecto. Además, el jefe de proyecto se encargará de supervisar el establecimiento de la arquitectura del sistema. Gestión de riesgos. Planificación y control del proyecto.

El Jefe de Proyecto tiene las siguientes características:

- Experiencia en el diseño y desarrollo de aplicaciones para Internet (Intranet/Extranet)
- Experiencia Liderando proyectos para Internet
- Dominio de las metodologías más eficaces para el análisis de la información
- Ha participado en distintos proyectos dirigiendo y cumpliendo otro tipo de perfiles lo que hacen de él un elemento con la capacidad de distribuir las tareas de un proyecto
- Posee también, conocimientos especializados en áreas específicas del proyecto como puede ser: diseño, implantación y administración de Bases de Datos, servidores Web

### 5.5.3. Analista de Sistemas

El Analista de Sistemas es la persona encargada de la captura, especificación y validación de requisitos, interactuando con el cliente y los usuarios mediante entrevistas. Elabora el Modelo de Análisis y Diseño. Colabora en la elaboración de las pruebas funcionales y el modelo de datos.

### 5.5.4. Arquitecto de Software

El Arquitecto es el encargado de definir el esquema de trabajo, la tecnología, gestión de requisitos, gestión de configuración e impacto en los cambios.

### **5.5.5. Analista Programador**

El Analista Programador está encargado de la construcción de prototipos, desarrollo de los componentes. Colabora en la elaboración de las pruebas funcionales y en las validaciones con el usuario.

Dentro de este perfil tenemos los siguientes:

#### **Analista Programador Junior**

- Egresado de la Universidad
- Conocimientos en Tecnología JAVA
- Conocimientos en Desarrollo JAVA
- Capacidad para trabajar en equipo

#### **Analista Programador Estándar**

- Egresado de la Universidad
- Experiencia en Análisis, Diseño y Desarrollo
- Conocimientos en Tecnología JAVA
- Experiencia en desarrollo JAVA (2 años)
- Capacidad para trabajar en equipo

#### **Analista Programador Senior**

- Egresado de la Universidad
- Experiencia en Análisis, Diseño y Desarrollo
- Conocimientos en Tecnología JAVA
- Conocimientos de arquitecturas
- Conocimiento de frameworks
- Experiencia en desarrollo JAVA (+4 años)
- Capacidad para trabajar en equipo

### **5.5.6. Programador**

El programador es aquel que posee una amplia experiencia en desarrollo de aplicaciones para Internet. Posee un criterio amplio y experiencia necesaria para el desarrollo de las aplicaciones del proyecto, de manera que resulten ser óptimas en tiempo de ejecución y uso de recursos.

Se encarga de investigar aquellos puntos del proyecto que necesiten una mejor visión, de manera que encuentre la mejor alternativa de solución que colabore con un óptimo desarrollo del proyecto. Tiene una base sólida en el conocimiento de sistemas operativos, herramientas de conectividad y desarrollo sobre distintas bases de datos, lo que le permite cumplir con los requerimientos del proyecto de manera eficaz.

Posee el conocimiento de las últimas herramientas de desarrollo en Internet, de manera que pueda utilizarlas eficientemente durante el desarrollo de sus aplicaciones.

### **5.5.7. Administrador de Bases de Datos**

El Administrador de Base de Datos se encarga de autorizar el acceso a la base de datos, de coordinar y vigilar su empleo, se encarga de la instalación y configuración de los motores de Base de datos. Realiza además las siguientes funciones:

- Mejoras a los motores de Base de Datos.
- Manejo de seguridad: a nivel de Base de Datos definiendo la estructura y sus objetos (tablas, campos, procedimientos, usuarios, roles, triggers, etc.), y a nivel de usuarios definiendo el acceso de los diferentes usuarios a los objetos de la Base de Datos
- Monitoreo del espacio físico y los archivos lógicos para optimizar la Base de Datos
- Respaldos y Recuperación de Información
- Transferir e Importar Datos y estructuras de Base de Datos externas
- Mantenimiento a la Base de Datos
- Implementación de Réplicas

### **5.5.8. Diseñador**

Diseña la interfaz de la aplicación, trabaja desde el análisis para tener una visión integral del proyecto. Analiza la situación, define la estrategia de comunicación, la estructura lógica del site, la arquitectura de la información, jerarquiza los contenidos de acuerdo a su importancia. Define el estilo gráfico y de sus principales páginas aprovechando al máximo todas las herramientas de comunicación en Internet.

### **5.5.9. Documentador**

Elabora todos los documentos requeridos para el sistema.

## 5.6. Análisis del Sistema

Se requiere un Sistema de Salud Ocupacional donde se puedan registrar empresas, pacientes (asociados o no a una empresa) que, deben ser sometidos a pruebas médicas de acuerdo a un perfil de exámenes previamente registrados. Terminada las pruebas, el sistema debe imprimir dichas pruebas junto con los datos calculados automáticamente.<sup>1</sup>

A continuación se muestran algunos diagramas:

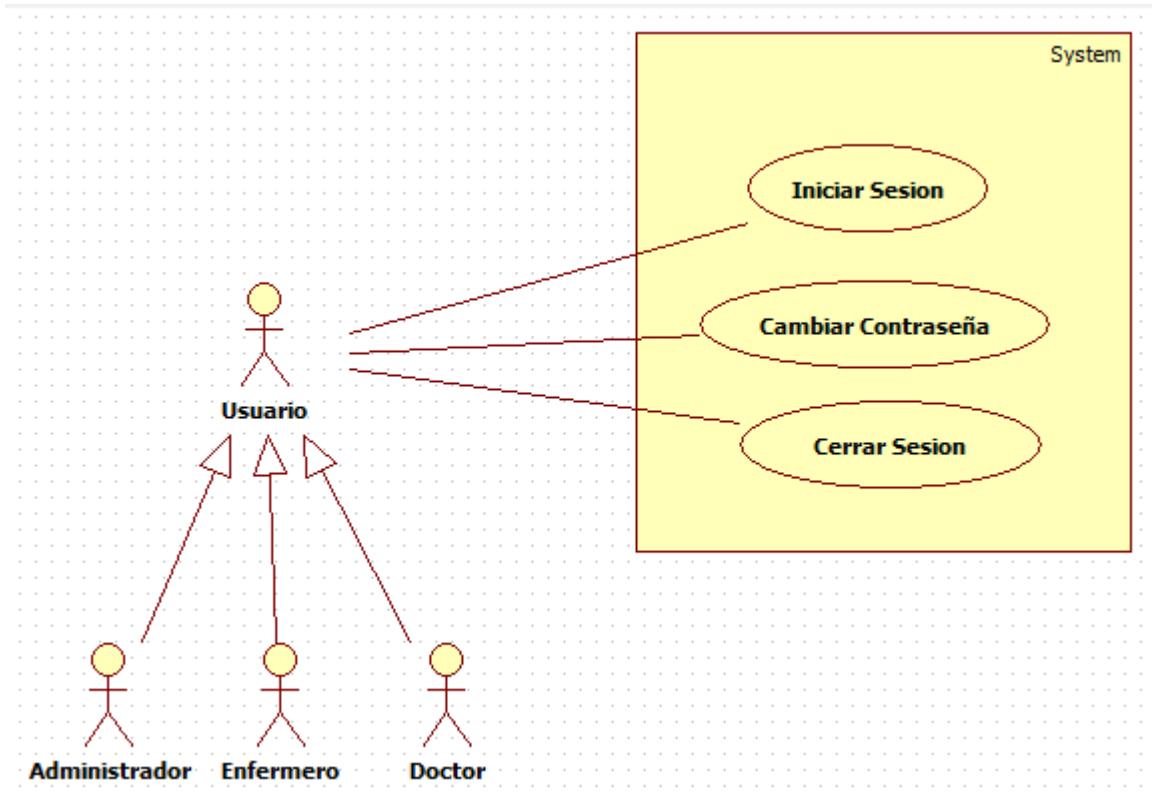


Figura 5.1: CU-1 Seguridad

<sup>1</sup>El párrafo es un resumen del documento de requerimientos real.

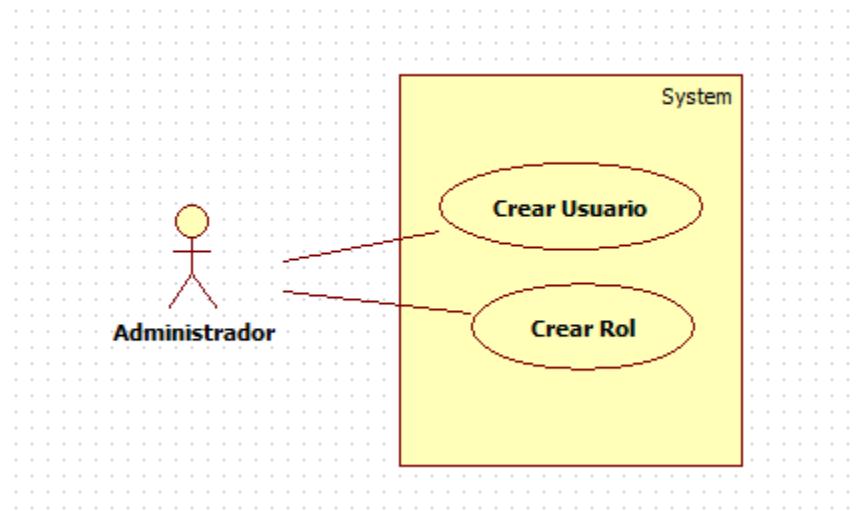


Figura 5.2: CU-2 Creación de Usuarios

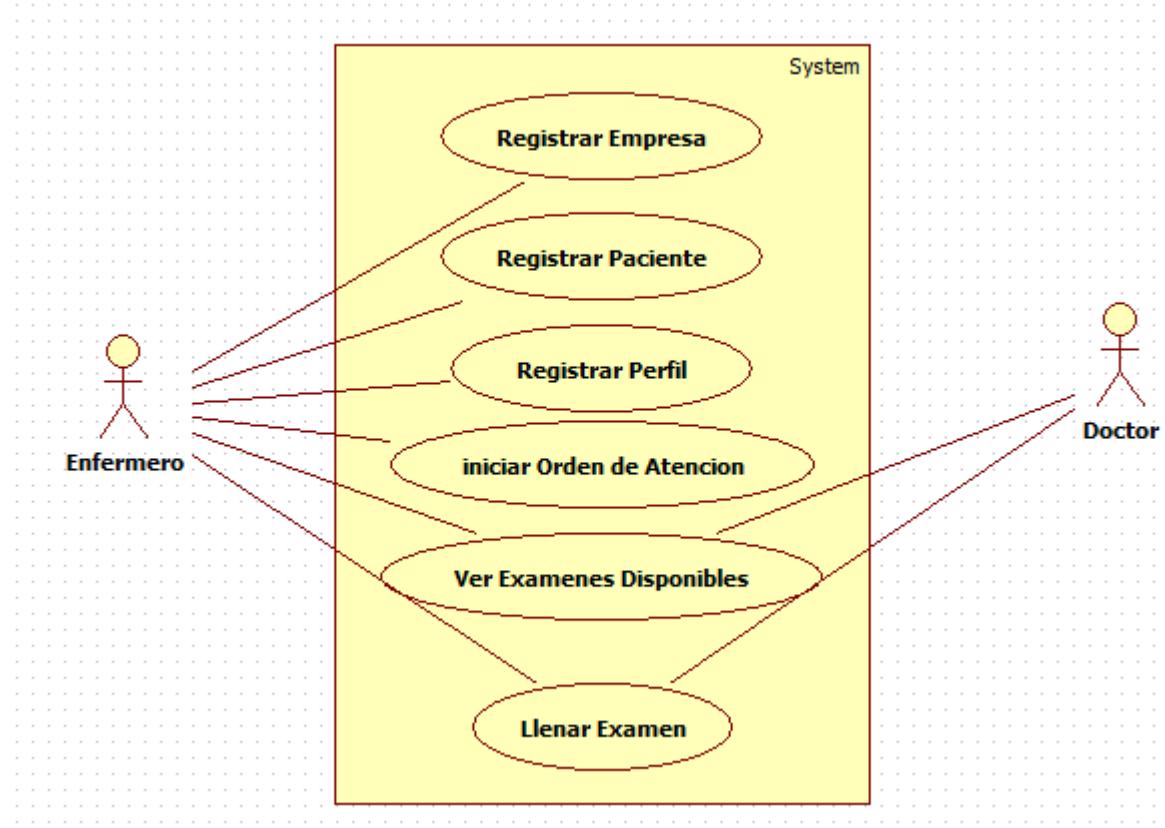


Figura 5.3: CU-3 Casos de Uso primordiales

## 5.7. Diseño del Sistema

A continuación se muestra parte del diagrama de clases y el diagrama de procesos (BPM):<sup>2</sup>

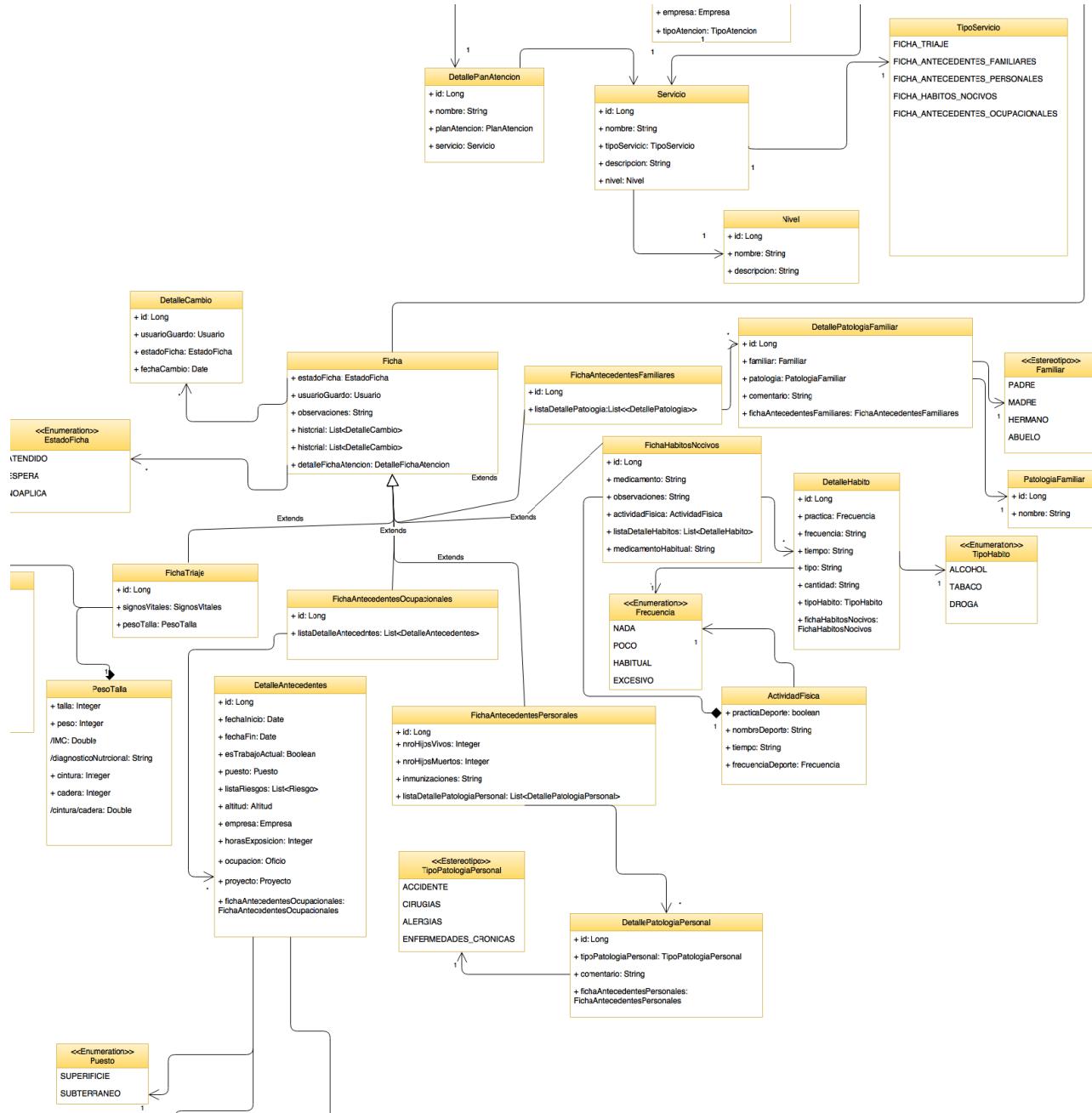


Figura 5.4: DC-A Diagrama de clases

<sup>2</sup>Se puede encontrar los diagramas completos y legibles en <https://www.github.com/w1lld33r/informe-practicas/tree/master/imgs>

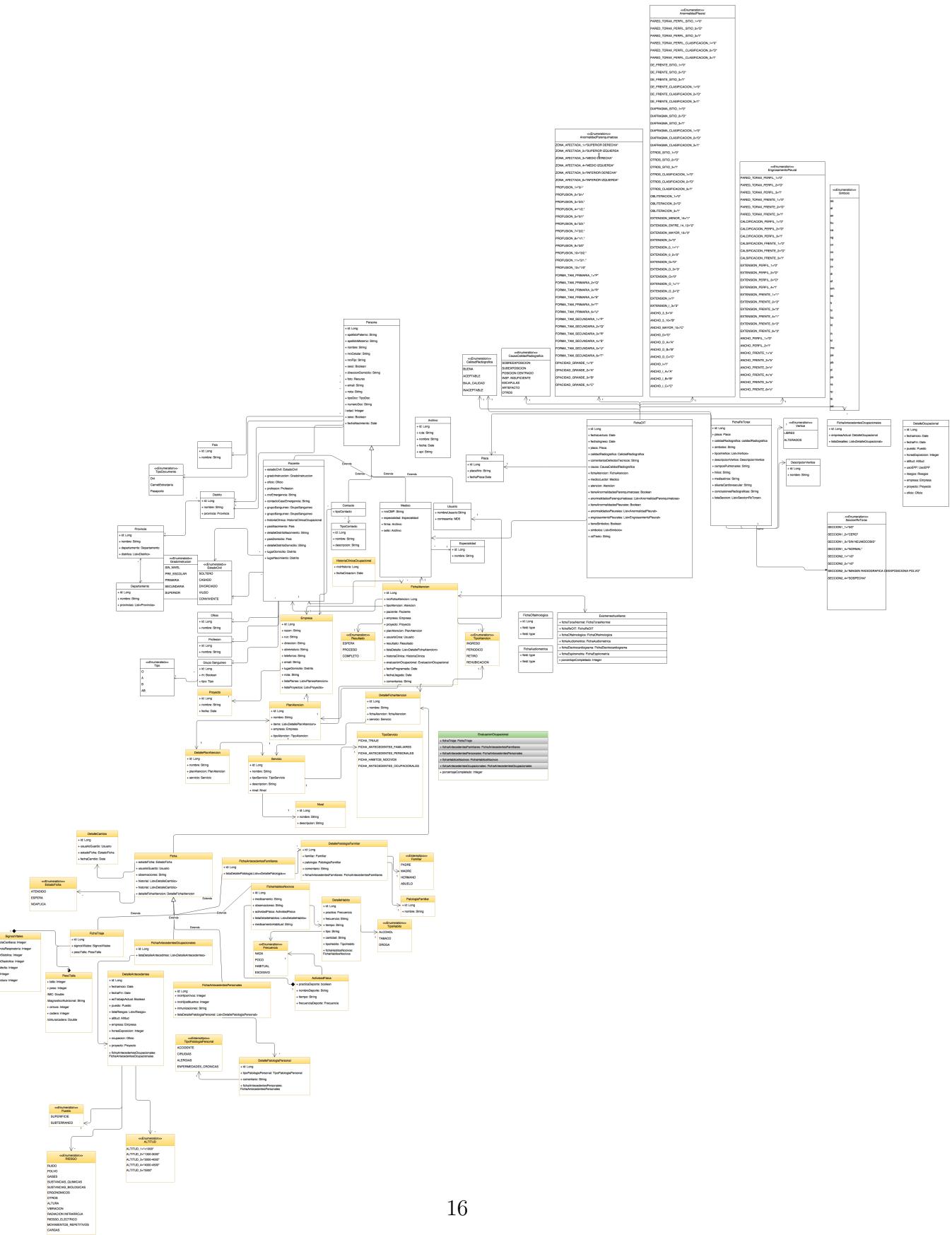


Figura 5.5: DC-B Diagrama de clases completo

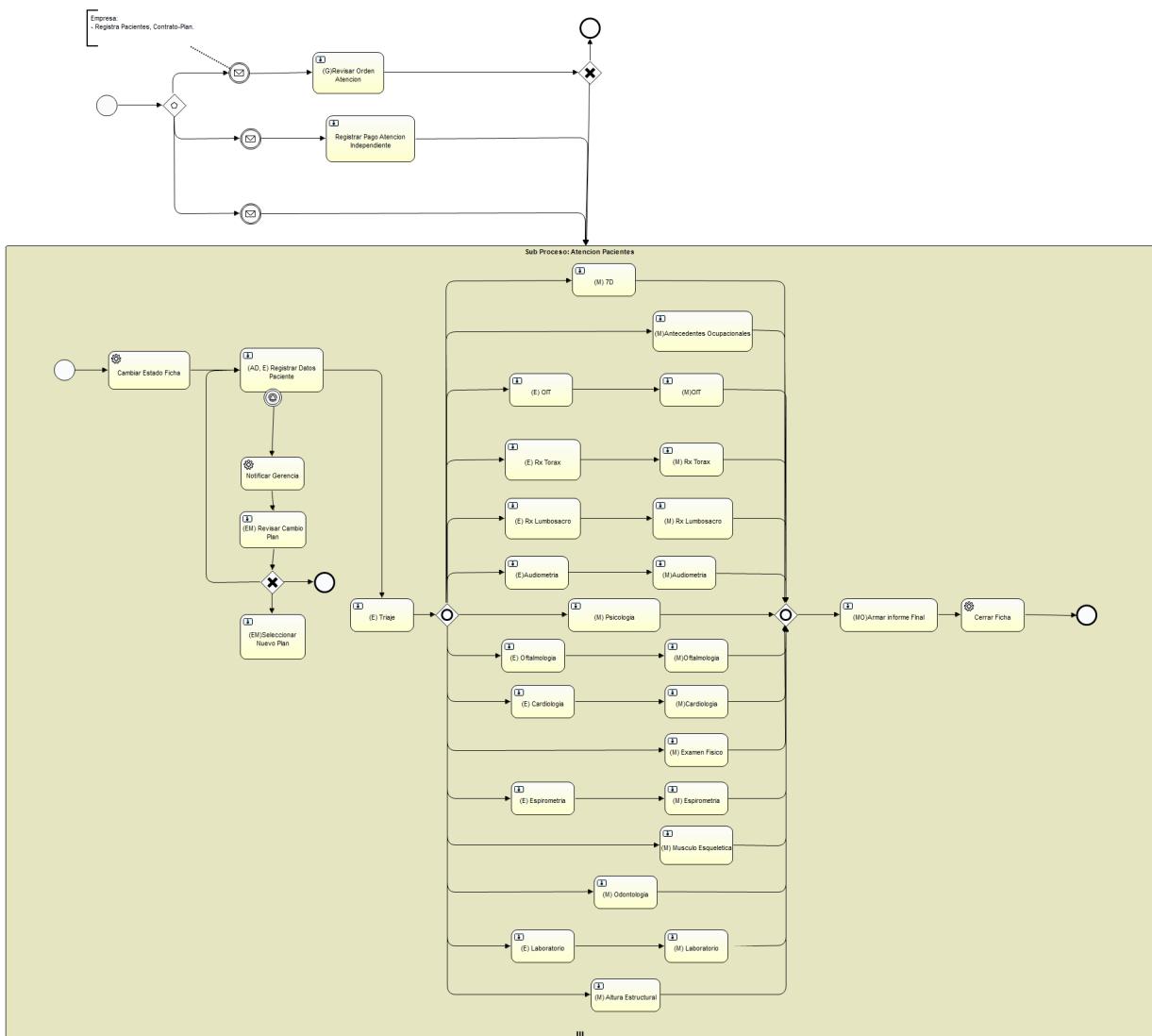


Figura 5.6: DP-1 Diagrama BPM

## 5.8. Arquitectura del Sistema

El Sistema sigue una arquitectura REST (Representational state trasfer) ó de servicios web RESTful cliente-servidor que, funciona bajo el protocolo HTTP, figura 5.7. Básicamente el backend se encarga de proporcionar recursos (previa autorización) al frontend.

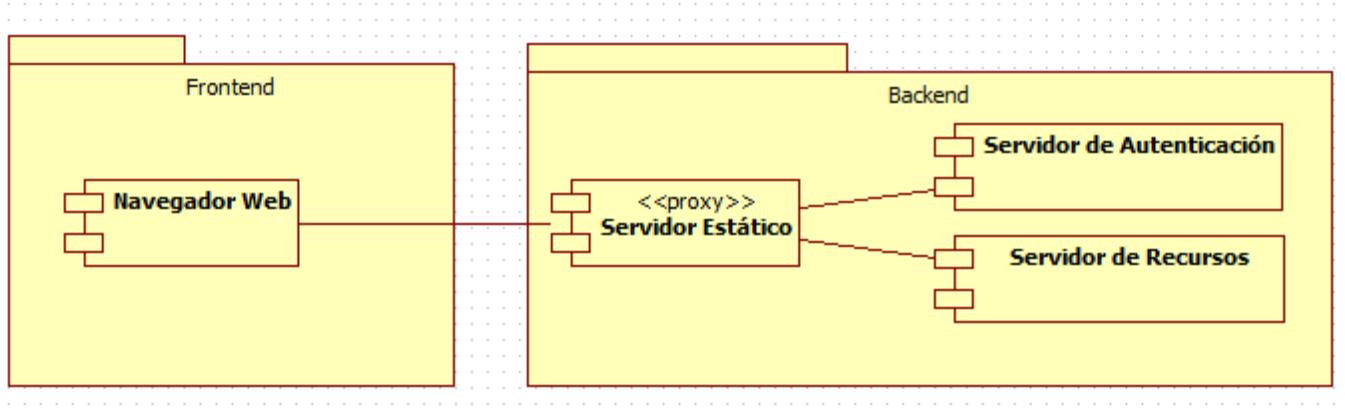


Figura 5.7: DA-1 Arquitectura del Sistema

En la figura 5.8 se muestra detalladamente la interacción de los componentes (frontend y backend).

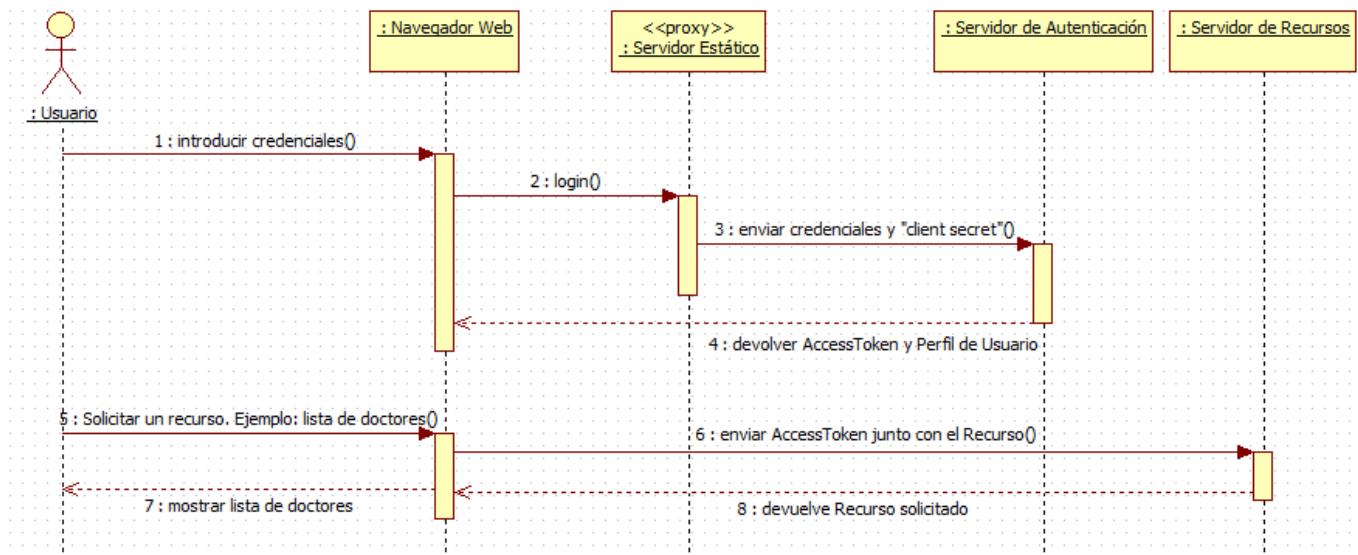


Figura 5.8: DA-2 Arquitectura del Sistema (Diagrama de secuencia)

## 5.9. Desarrollo del Sistema

Para el desarrollo del sistema se utilizó el siguiente stack de tecnologías (frontend y backend):

- HTML y CSS (Bootstrap)
- Javascript (AngularJS)
- NodeJS (ExpressJS)
- Java (Spring y Spring Boot)

### 5.9.1. Arquitectura de Aplicaciones Web Modernas

A principios de la web, las “aplicaciones web” no existían. La web consistía de contenido estático e imágenes.

Ventajas:

- Bajo consumo computacional en el servidor
- No se necesitan conocimientos avanzados en programación

Desventajas:

- Actualización de contenido engorroso
- Personalización inexistente, todas las personas que visitan la web, ven el mismo contenido
- Interfaz de usuario muy básica, comparado con las aplicaciones actuales

Las aplicaciones web modernas están compuestas de clientes frontend y una infraestructura backend distribuida (para servir contenido) a través del uso de APIs RESTful (figura 5.9).

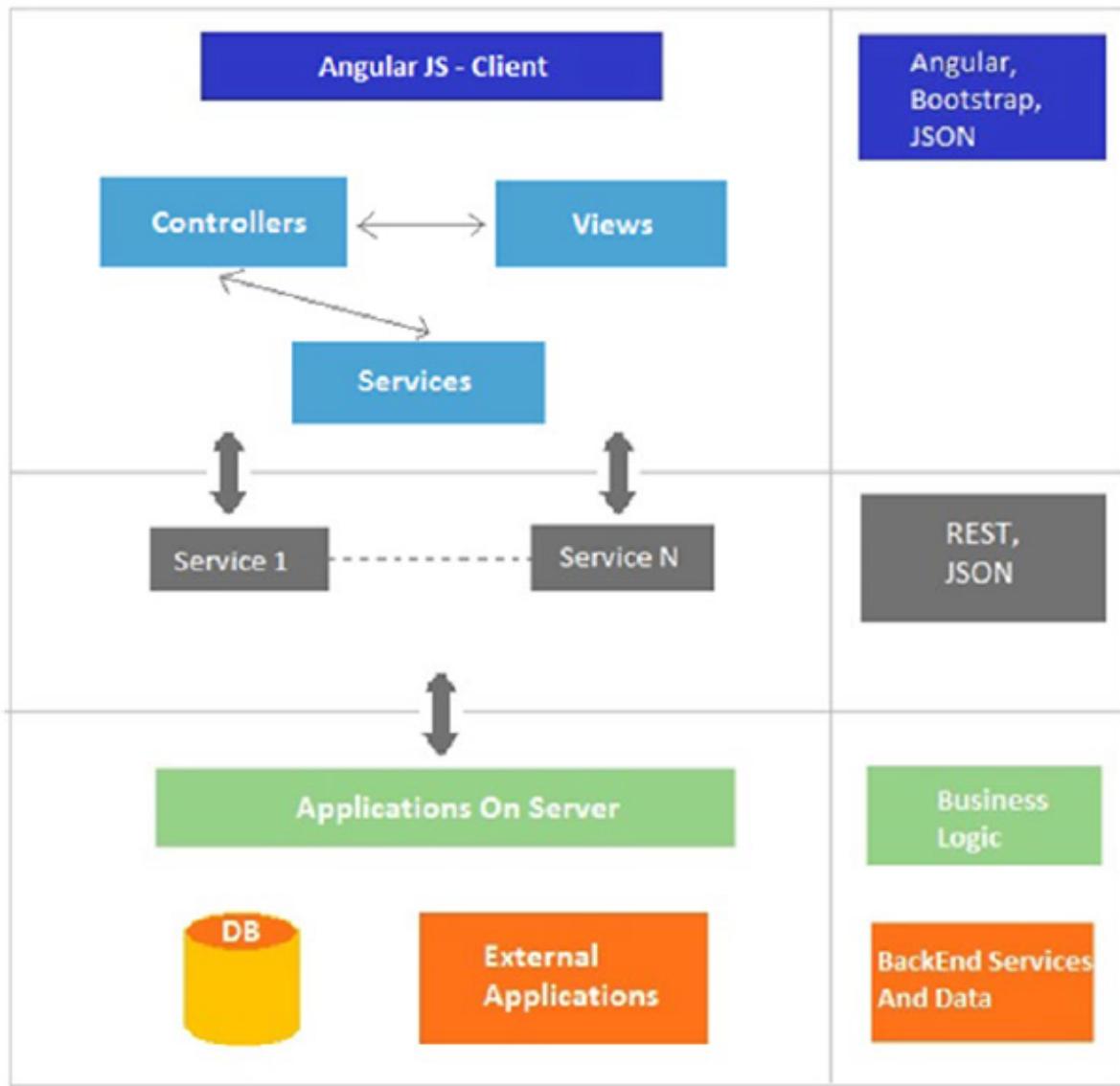


Figura 5.9: Arquitectura de aplicaciones web modernas

### 5.9.2. Bootstrap

Bootstrap es un framework CSS que facilita la creación de sitios web Responsive Web Design. Bootstrap viene con un conjunto de componentes de interfaz web listos para usar y una guía de estilos bastante completa, también provee grillas para el correcto maquetado de una web profesional.

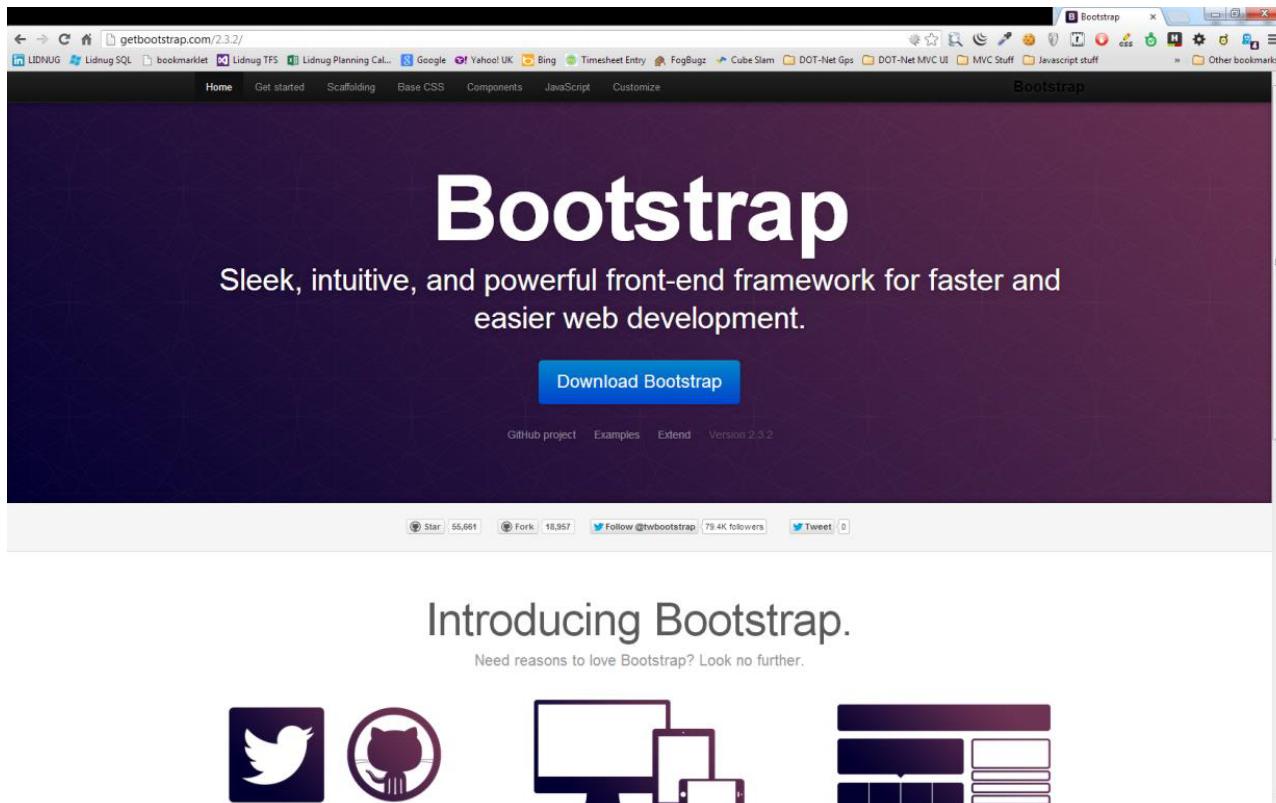


Figura 5.10: Sitio Web de Bootstrap

### 5.9.3. AngularJS

AngularJS es un framework de código abierto, soportado por Google. AngularJS permite a los desarrolladores crear aplicaciones web SPA (Single Page Applications), lo que facilita la experiencia de usuario, al no tener que cargar la aplicación web completa dos ó más veces.

A diferencia de las aplicaciones web SPA, en las aplicaciones web tradicionales (como se ilustra en la figura 5.11), un cliente web hace una petición al servidor y éste responde con toda la página web completa; luego, el usuario hace click en un enlace interno, el servidor vuelve a cargar la página web completa y se produce un retraso en la carga de la página, lo que produce una mala experiencia de usuario.<sup>3</sup>

<sup>3</sup>Fuente de la imagen: <https://www.codeschool.com/course/shaping-up-with-angularjs>

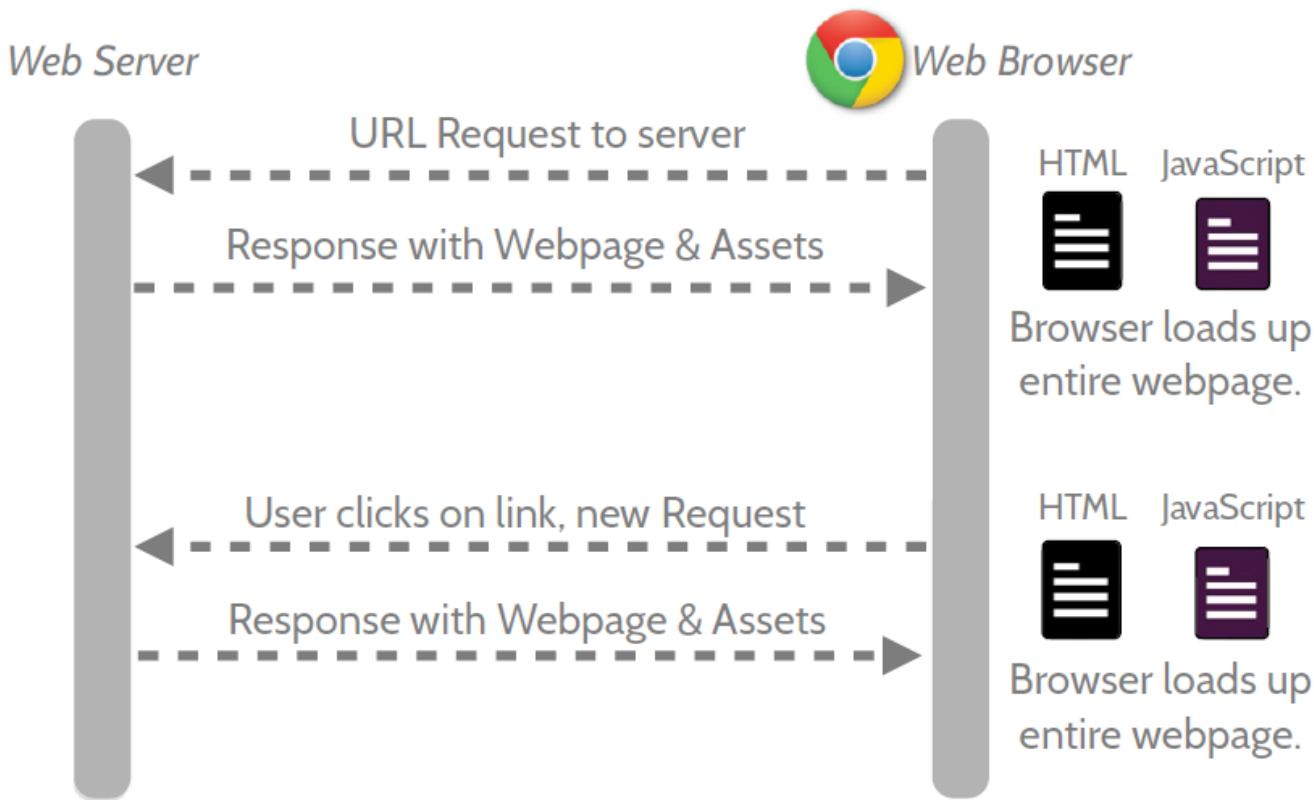


Figura 5.11: Carga de página tradicional

En las aplicaciones SPA, el cliente solicita una web y el servidor inicialmente sirve la aplicación web completa (una sola vez); luego, el usuario hace click en un enlace interno, el servidor solo responde con el recurso solicitado, más no con la página web completa. En la figura 5.12 se muestra esta interacción.<sup>4</sup>

<sup>4</sup>Fuente de la imagen: <https://www.codeschool.com/course/shaping-up-with-angularjs>

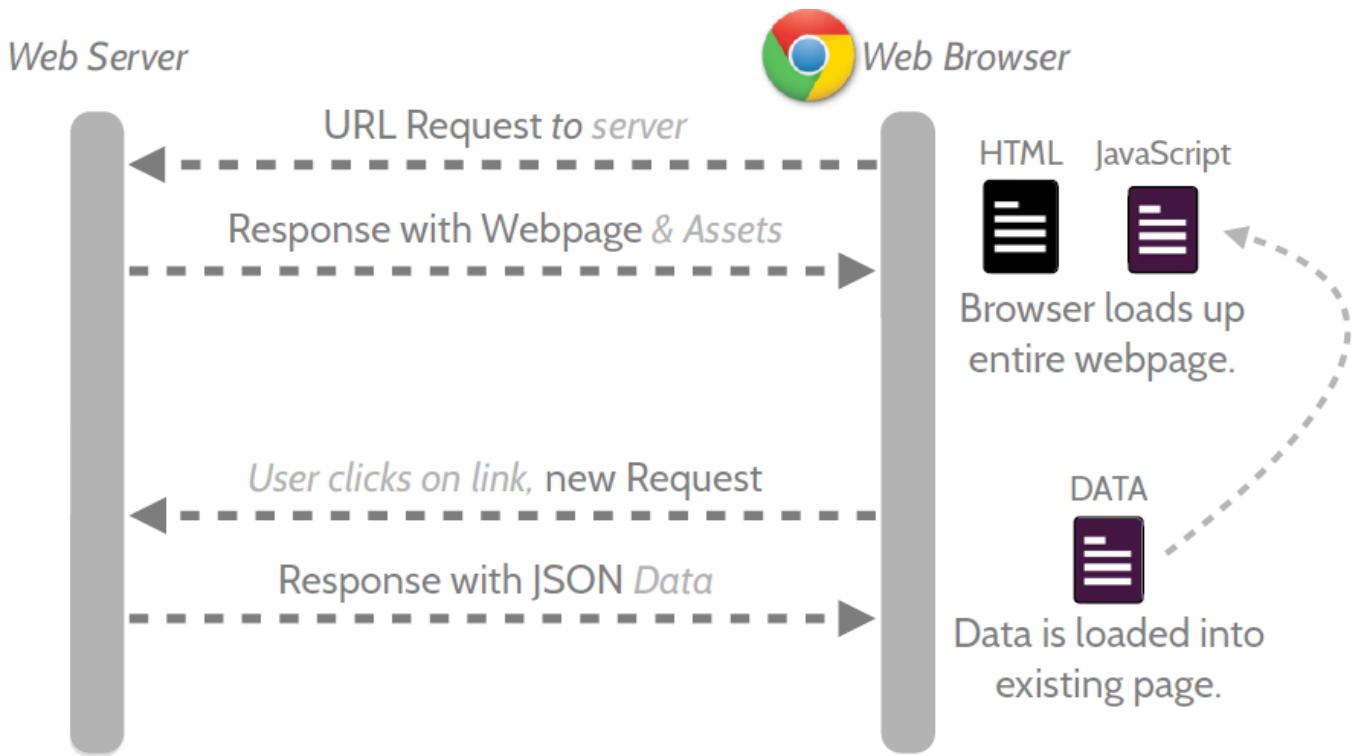


Figura 5.12: Carga de página SPA

## Componentes

AngularJS ofrece una estructura modular y escalable para construir aplicaciones web, figura 5.13.

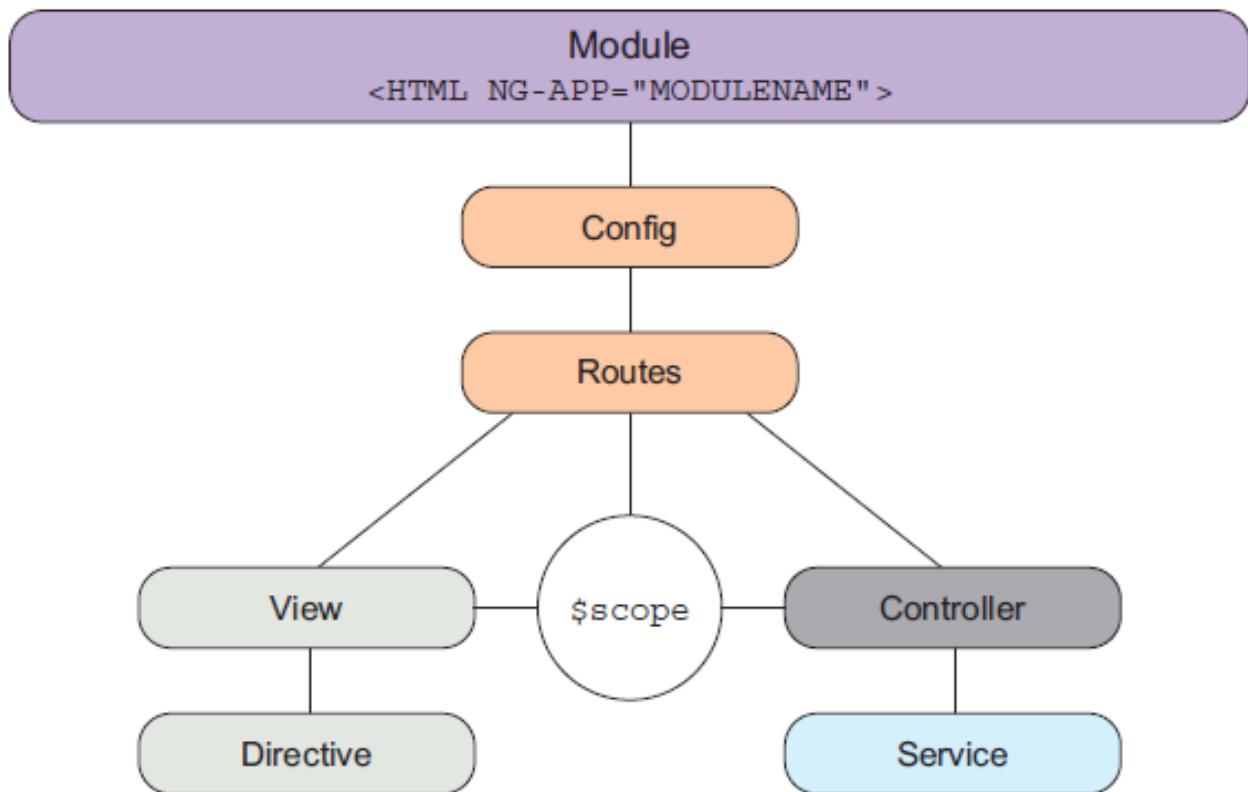


Figura 5.13: Componentes de AngularJS

<i>Componente</i>	<i>Propósito</i>
Module	Sirve como un contenedor que ayuda a organizar el código. Los Módulos pueden contener submódulos, haciendo fácil la composición de funcionalidades dentro de una aplicación web.
Config	El bloque de configuración permite ejecutar código javascript antes de que se ejecute la aplicación.
Routes	Permite definir rutas para navegar a estados específicos de la aplicación.
View	La vista es la plantilla HTML que mostrará la aplicación.
\$scope	El \$scope es básicamente el pegamento entre la vista y el controlador
Controller	El controlador es responsable de la definición de métodos y propiedades. La vista es quien interactúa con el controlador.
Directive	Es una extensión de una vista, permite crear elementos HTML personalizados que encapsulan comportamientos.
Service	Proporcionan funcionalidades comunes a una aplicación.

Cuadro 5.1: Resumen de componentes AngularJS

## Arquitectura

AngularJS sigue una arquitectura MVC (Modelo-Vista-Controlador) para crear aplicaciones web. MVC es un patrón de arquitectura que separa los datos, lógica y presentación (figura 5.14). Propuesto por *Trygve Reenskaug* y después fue implementado en el lenguaje de programación Smalltalk en los setenta [Scott, 2016].

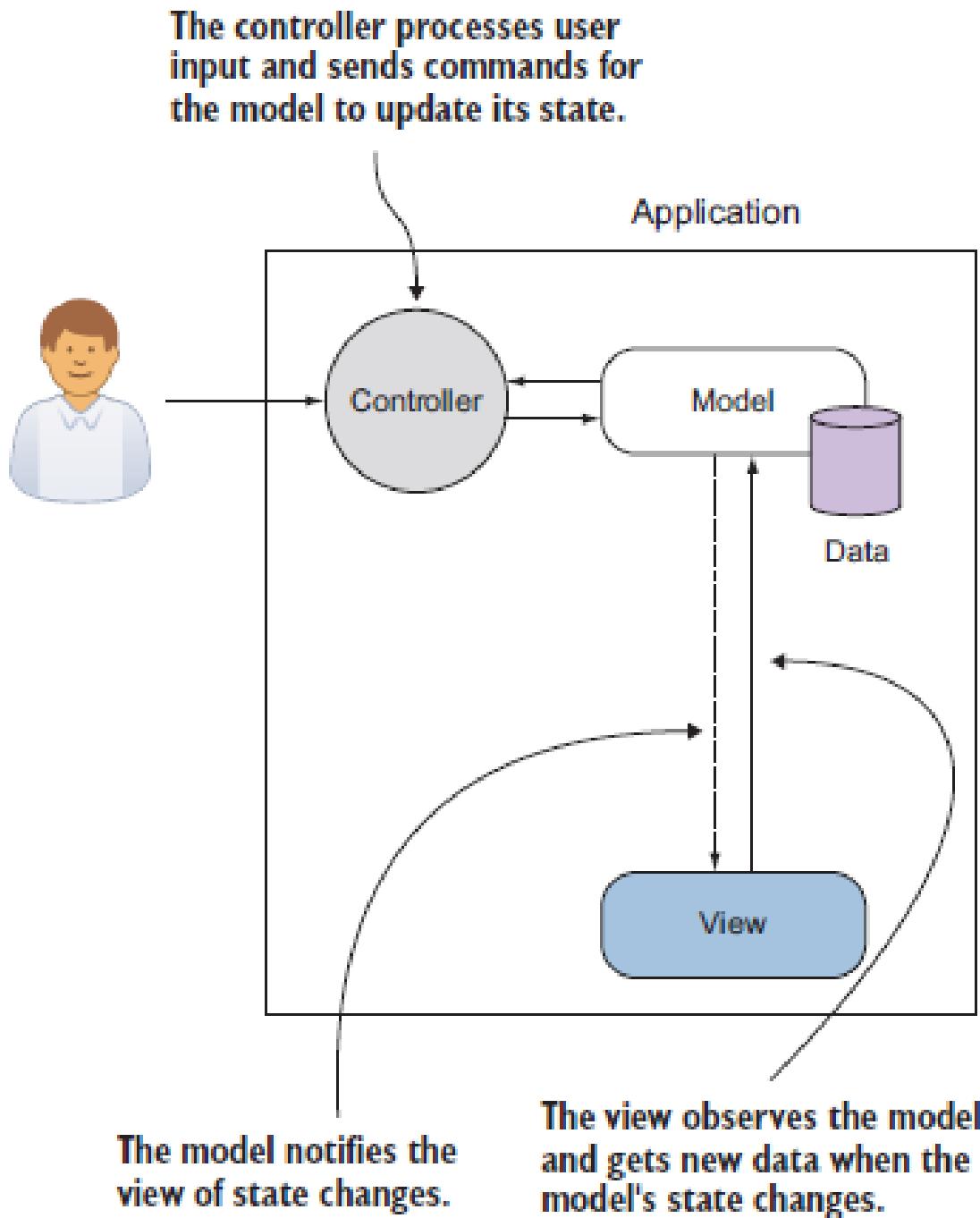


Figura 5.14: Arquitectura Modelo-Vista-Controlador

#### 5.9.4. NodeJS y ExpressJS

NodeJS no es un lenguaje de programación, es una plataforma capaz de ejecutar aplicaciones complejas escritas en Javascript; se ejecuta en el lado del servidor como cualquier otro lenguaje (Java, Python, Ruby, etc). Para el sistema fue usado como un proxy, y a la vez como un servidor estático.

NodeJS está construido sobre el motor de javascript V8 creado por Google. NodeJS posee varias librerías escritas en C, junto con módulos para la manipulación de datos binarios. Puede acceder a funciones del sistema operativo, etc. Éstas librerías permiten a NodeJS acceder a archivos, ejecutar comandos del sistema, escuchar/responder a peticiones de red.

La figura 5.15 muestra los componentes principales de NodeJS:

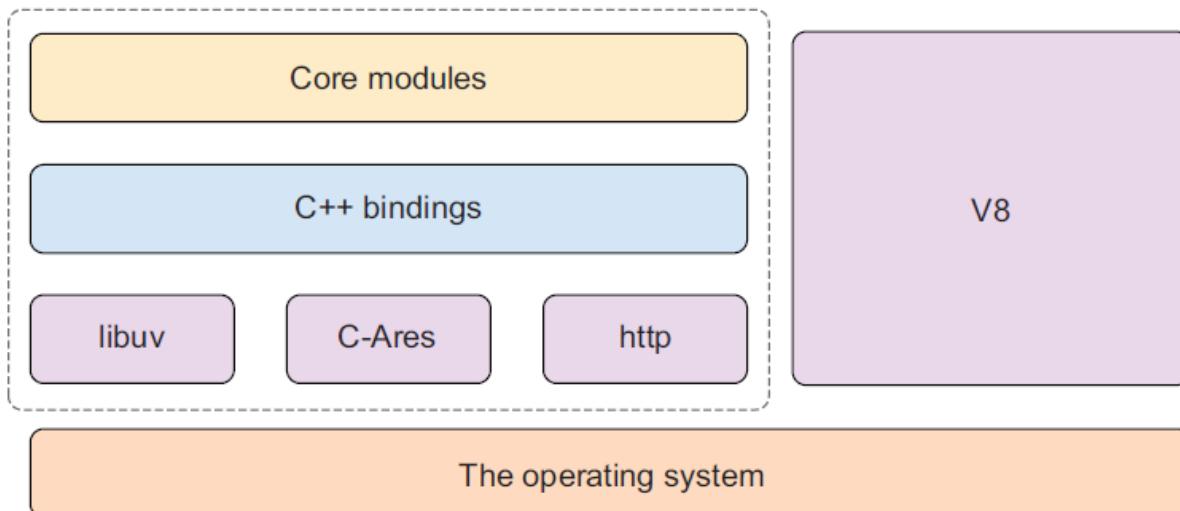


Figura 5.15: Componentes de NodeJS

ExpressJS es un framework web para NodeJS. Proporciona una API sencilla de utilizar, y agiliza el desarrollo de proyectos backend.

### 5.9.5. Spring y Spring Boot

Spring es un framework java para el desarrollo de aplicaciones a nivel empresarial, inició como una alternativa liviana al estándar **Java Enterprise Edition (JEE)**. Spring ofrece características como inyección de dependencias y un modelo de programación orientada a aspectos.

#### Arquitectura

Como se muestra en la figura 5.16, la arquitectura de Spring es modular, lo que permite elegir únicamente los módulos que necesitemos, sin necesidad de traer el framework completo.

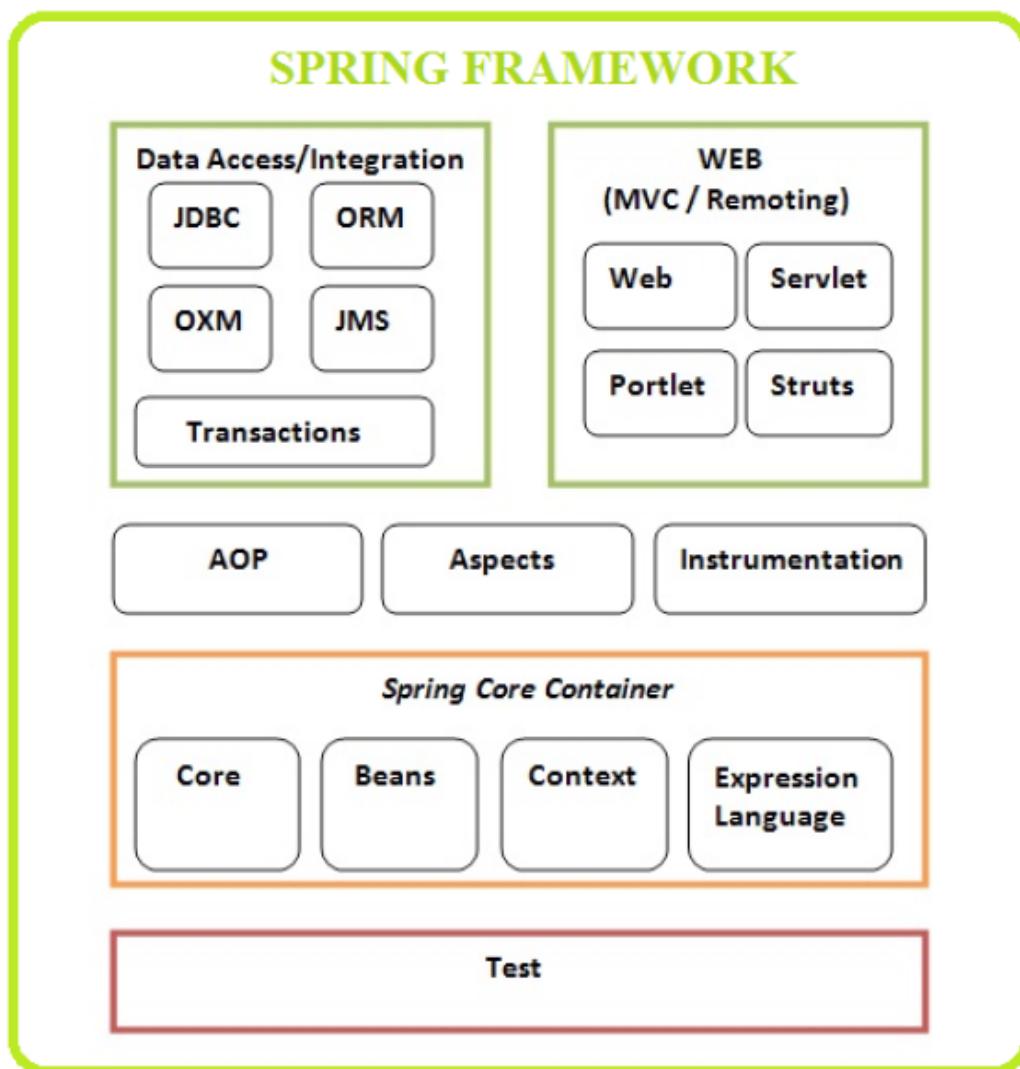


Figura 5.16: Arquitectura de Spring Framework

<b>Módulo</b>	<b>Propósito</b>
Core Container	Proporciona características de IoC (Inversión de Control) e Inyección de dependencias.
AOP & Instrumentation	Proporciona un mecanismo para introducir nuevas funcionalidades a un código existente sin modificar su diseño.
Data Access/Integration	Se encarga de gestionar el acceso a bases de datos en una aplicación.
Web	Proporciona un buen soporte para la creación de aplicaciones web robustas.
Test	Ayuda a realizar pruebas unitarias y de integración usando JUnit y TestNG.

Cuadro 5.2: Módulos de Spring

Spring brinda muchas facilidades a la hora de programar, pero la configuración es un tanto compleja. Spring Boot simplifica todas las configuraciones iniciales para comenzar un proyecto web de manera rápida. Cabe mencionar que Spring Boot no es una herramienta ni una alternativa a Spring, es un proyecto que necesita de Spring para facilitarnos la vida.

Spring Boot trae algo de magia al desarrollo de aplicaciones con Spring. Existen cuatro características fundamentales:

- **Configuración Automática:** Spring Boot proporciona funcionalidades a nuestro proyecto de manera automática.
- **Dependencias Iniciales:** De acuerdo al tipo de proyecto, Spring Boot facilita las librerías necesarias para iniciar un proyecto.
- **Interfaz de Línea de Comandos:** Esta característica opcional permite escribir trozos de código directamente en la terminal e ir probando funcionalidades.
- **Actuador (*The Actuator*):** Muestra características de una aplicación en tiempo de ejecución.

La página oficial (<http://start.spring.io>) del proyecto Spring Boot ofrece un asistente para la creación de proyectos utilizando maven o gradle:

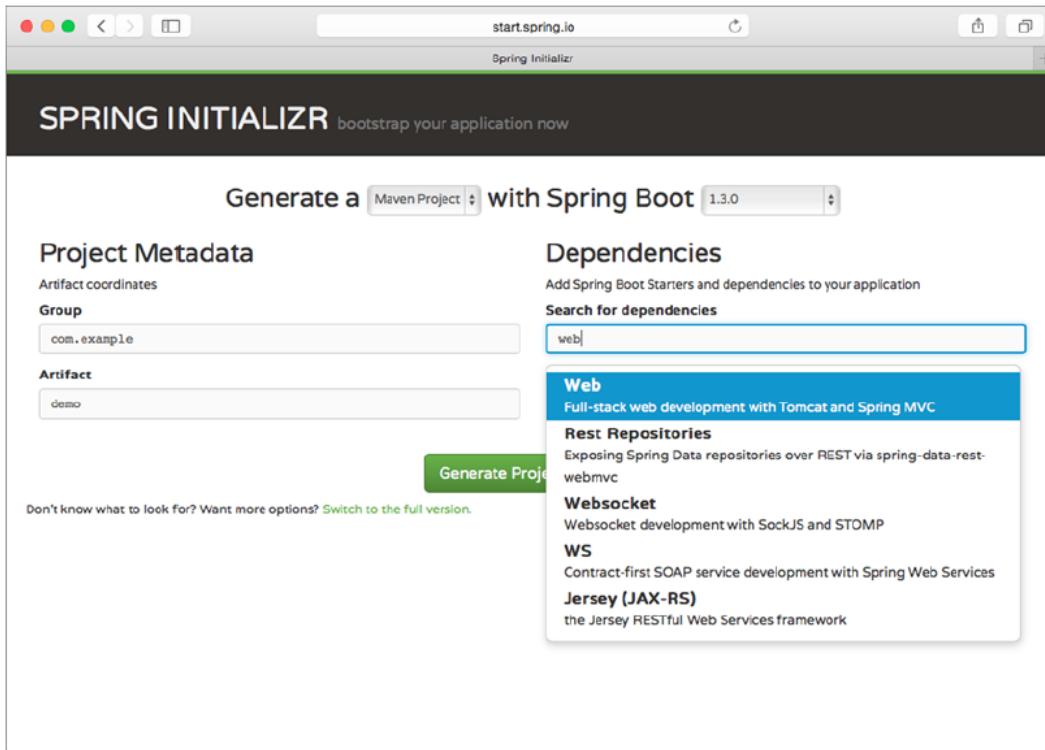


Figura 5.17: Asistente de Spring Initializr

Después de generar el proyecto, obtenemos un archivo .zip el cual podemos importar a IDEs como Eclipse o NetBeans. La estructura de un proyecto Spring Boot es muy sencilla como se muestra en la figura 5.18.

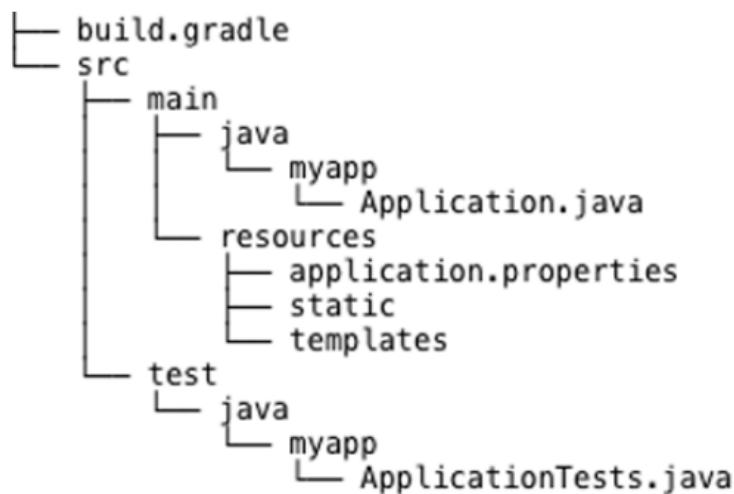


Figura 5.18: Estructura de un proyecto Spring Boot

El ecosistema Spring cuenta con otros proyectos que vale la pena conocer, a continuación se listan algunos de ellos:

<i>Proyecto</i>	<i>Descripción</i>
Spring Cloud	Proporciona un conjunto de herramientas para sistemas distribuidos. Útil para la construcción y despliegue de microservicios.
Spring Data	Proporciona una aproximación consistente para el acceso a bases de datos: Relacionales, no-relacionales, map-reduce, etc.
Spring Batch	Simplifica y optimiza el procesamiento de altos volúmenes de operaciones en bloque.
Spring Security	Protege y asegura nuestras aplicaciones de una forma simple. Soporte para autenticación y autorización.
Spring Hateoas	Simplifica la creación de servicios RESTful y sigue el principio HATEOAS.
Spring Rest Docs	Documenta automáticamente nuestros servicios RESTful.
Spring Social	Conecta fácilmente nuestras aplicaciones con APIs de terceros: Facebook, Twitter, Linkedin y más.

Cuadro 5.3: Proyectos del ecosistema Spring

---

# Conclusiones y Recomendaciones

---

## 6.1. Conclusiones

- Las prácticas realizadas en todo el proceso del desarrollo de software, obtención de requisitos, análisis de requisitos, diseño de sistemas, desarrollo e implementación; son actividades que el ingeniero de sistemas debe dominar, para realizar proyectos que ayuden a las personas y por ende a la sociedad.
- Uno de los obstáculos al que se enfrenta un prácticante es el de trabajar en equipo, transmitir ideas con claridad, pero que con la práctica se logra superar.
- La oportunidad que me dio la empresa **WELL DONE SOLUTIONS SAC**, de hacerme partícipe en la construcción de un proyecto real desde un comienzo, fue un reto y que, además me fortaleció como ingeniero de sistemas.

## 6.2. Recomendaciones

- Se recomienda a la empresa **Well Done Solutions SAC** abra las puertas a nuevos estudiantes quienes estén a punto de realizar sus prácticas pre profesionales, para que los futuros ingenieros de sistemas estén muy bien capacitados.
- Se recomienda a la empresa **Well Done Solutions SAC** brinde talleres informativos a estudiantes de los distintos semestres de la facultad de ingeniería de sistemas de esta casa de estudios.
- Se recomienda a la Carrera Académico Profesional de **Ingeniería de Sistemas** realizar convenios con empresas de desarrollo de software, y así el estudiante podrá enfrentarse a proyectos reales.
- Se recomienda a los estudiantes de la Carrera Académico Profesional de **Ingeniería de Sistemas** descargar este documento y su código fuente (hecho en L<sup>A</sup>T<sub>E</sub>X) en el siguiente repositorio: <https://www.github.com/w11ld33r/informe-practicas>, para que puedan tener una noción de este informe y poder adecuarlo a sus necesidades.

---

## **Anexos**

---



Figura 7.1: Pantalla contribución backend



Figura 7.2: Pantalla contribución frontend

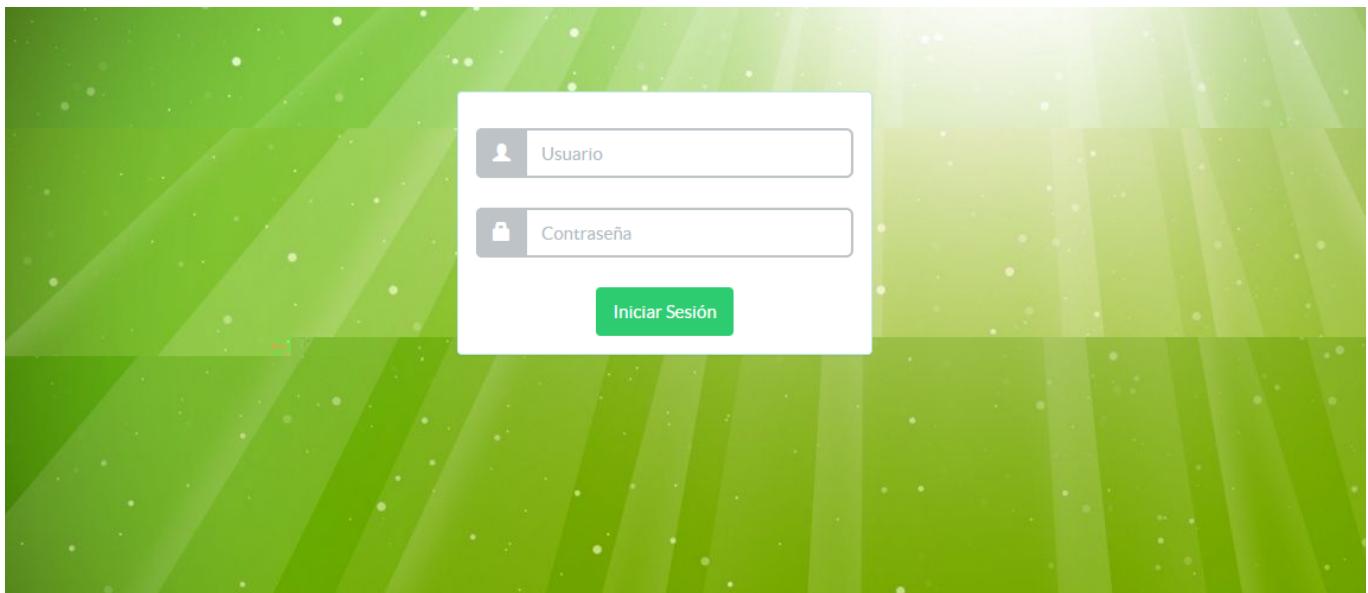


Figura 7.3: Pantalla login

### Nueva Empresa

RUC: 65498798465 | Rázón Social: JULSA SAC | Abreviatura: JUL

Actividad Económica: Transporte | Tipo: Empresa

Distrito: JULIACA - SAN ROMAN - PUNO | Dirección: Terminal Terrestre Juliaca

Teléfono: 684651654 | Fax: | Correo Electrónico: contacto@julsa.com

Nota:

Activar Campos



Seleccionar archivo: julsa.png

Figura 7.4: Pantalla nueva empresa

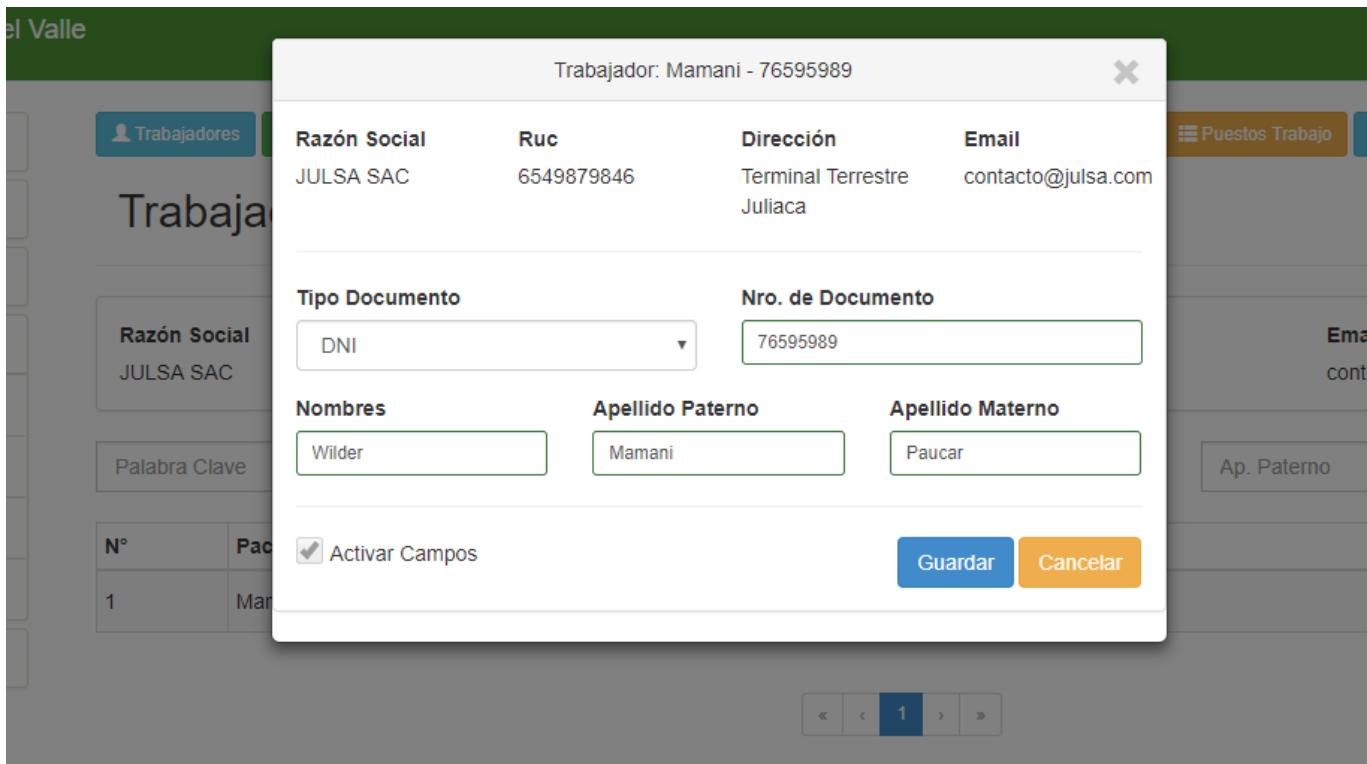


Figura 7.5: Pantalla nuevo paciente

Nº	Paciente	Actividad	Empresa	Orden	Plan	PuestoTrabajo	Creado	Acciones
1	MAMANI PAUCAR, Wilder	(M)Antecedentes Ocupacionales	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	
2	MAMANI PAUCAR, Wilder	(M) Examen Fisico	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	
3	MAMANI PAUCAR, Wilder	(M) Musculo Esqueletica	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	
4	MAMANI PAUCAR, Wilder	(M) Odontologia	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	
5	MAMANI PAUCAR, Wilder	(M) 7D	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	
6	MAMANI PAUCAR, Wilder	(M) Altura Estructural	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	
7	MAMANI PAUCAR, Wilder	(M) Psicologia	JULSA SAC	0002	PERFIL A-Ingreso	Administrador	15 noviembre 2017 02:17	

Figura 7.6: Pantalla exámenes a realizar

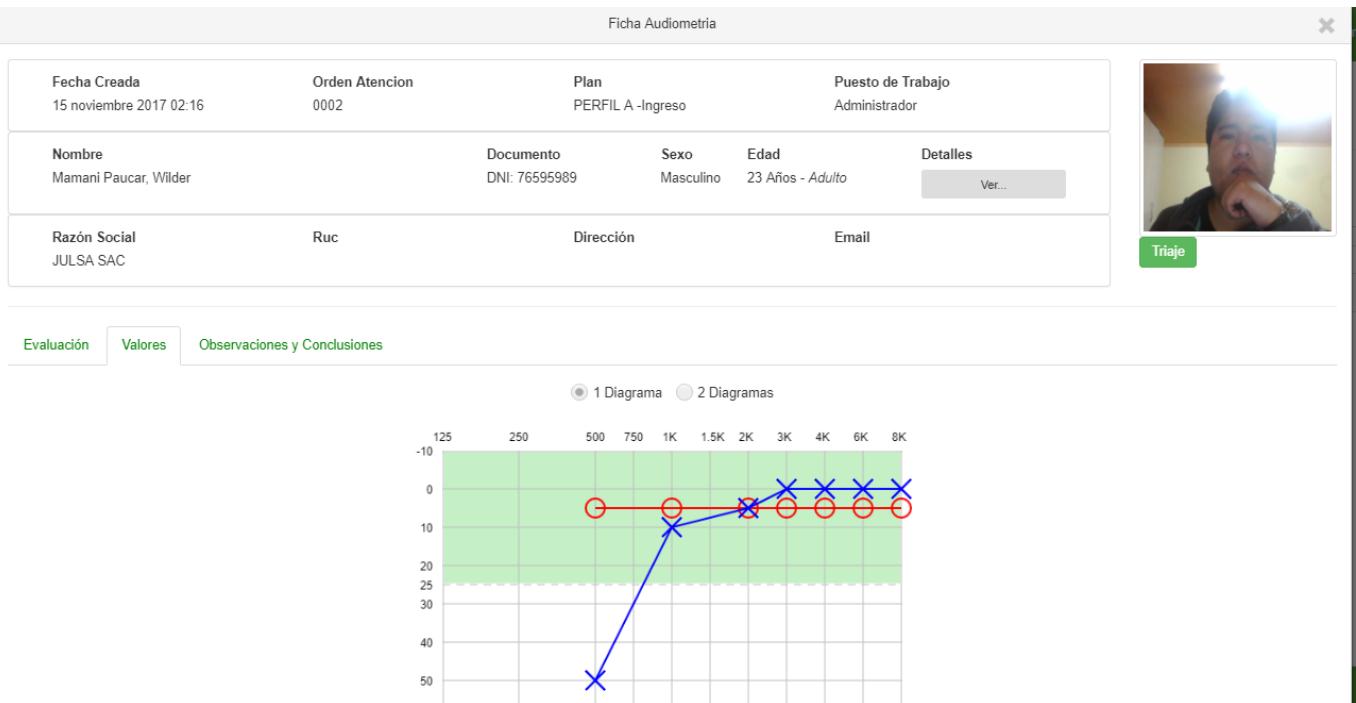


Figura 7.7: Pantalla examen audiometría

NUEVO ANTECEDENTE OCUPACIONAL

Fecha Creada 15 noviembre 2017 02:16	Orden Atencion 0002	Plan PERFIL A -Ingreso	Puesto de Trabajo Administrador
Nombre Mamani Paucar, Wilder	Documento DNI: 76595989	Sexo Masculino	Edad 23 Años - Adulto
Razón Social JULSA SAC	Ruc	Dirección	Email


Ver...

Tráje

**Trabajo Actual**

No hay Registros para mostrar

**Lista de Antecedentes Ocupacionales**

No hay Registros para mostrar    + Nuevo    Sin antecedentes Ocupacionales

Ver PDF

Activar Campos.

Liberar Tarea    Guardar    Sair

Figura 7.8: Pantalla antecedentes ocupacionales

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a tree of Java files under the package com.pe.wds.clinica. In the center, the code editor shows the ClinicaResourceServerMain.java file. The code is a main method that runs a Spring application, setting up endpoint services and mapping requests to endpoints.

```

1 package com.pe.wds.clinica.config;
2
3 import java.util.HashMap;
4
5 @EnableAutoConfiguration
6 @ComponentScan(basePackages = { "com.pe.wds.clinica" })
7 public class ClinicaResourceServerMain {
8
9     public static void main(String[] args) {
10         // ConfigurableApplicationContext context =
11         // SpringApplication.run(ClinicaResourceServerMain.class, args);
12         SpringApplication sa = new SpringApplication(ClinicaResourceServerMain.class);
13         sa.addListeners(new ApplicationPidFileWriter());
14         sa.addListeners(new EmbeddedServerPortFileWriter());
15         ConfigurableApplicationContext context = sa.run(args);
16         EndpointService endpointService = context.getBean(EndpointService.class);
17
18         RequestMappingHandlerMapping handlerMapping = context.getBean(RequestMappingHandlerMapping.class);
19
20         Map<RequestMappingInfo, HandlerMethod> methods = handlerMapping.getHandlerMethods();
21         Map<String, Endpoint> data = new HashMap<>();
22         for (Map.Entry<RequestMappingInfo, HandlerMethod> entry : methods.entrySet()) {
23             RequestMappingInfo key = entry.getKey();
24             PatternsRequestCondition patternsCondition = key.getPatternsCondition();
25             String pattern = patternsCondition.getPatterns().iterator().next().toString();
26             Endpoint endpoint = data.get(pattern);
27             if (endpoint == null)
28                 endpoint = new Endpoint(pattern);
29             data.put(pattern, endpoint);
30         }
31     }
32 }

```

Figura 7.9: Pantalla backend código principal

The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays a tree of Java files under the package com.pe.wds.clinica. In the center, the code editor shows the TareaAudiometriaCtrl.java file. This controller handles GET requests for task IDs, retrieving tasks, execution IDs, and attention variables from services, and then assembling FichaAudiometria objects based on the results.

```

67
68     @Autowired
69     private HistoryService historyService;
70
71     @RequestMapping(method = RequestMethod.GET)
72     public ResponseEntity<FichaAudiometria> get(@PathVariable("taskId") String taskId) {
73         Task task = tareaService.getById(taskId);
74         String executionId = task.getExecutionId();
75         AtencionVariable atencionVariable = (AtencionVariable) runtimeService.getVariable(executionId, "atencion");
76         Long fichaAtencionId = atencionVariable.getFichaAtencionId();
77         FichaAtencion fichaAtencion = fichaAtencionService.getById(fichaAtencionId);
78
79         FichaAudiometria fichaAudiometria = fichaAudiometriaService.get(fichaAtencionId);
80
81         FichaAudiometriaEvaluacion evaluacion;
82         FichaAudiometriaValores valores;
83         FichaAudiometriaResultados resultados;
84
85         if (fichaAudiometria == null) {
86             fichaAudiometria = new FichaAudiometria();
87             evaluacion = new FichaAudiometriaEvaluacion();
88             valores = new FichaAudiometriaValores();
89             resultados = new FichaAudiometriaResultados();
90         } else {
91             evaluacion = evaluacionCtrl.get(fichaAtencionId).getBody() == null ? new FichaAudiometriaEvaluacion()
92                                     : evaluacionCtrl.get(fichaAtencionId).getBody();
93
94             valores = valoresCtrl.get(fichaAtencionId).getBody() == null ? new FichaAudiometriaValores()
95                                     : valoresCtrl.get(fichaAtencionId).getBody();
96             resultados = resultadosCtrl.get(fichaAtencionId).getBody() == null ? new FichaAudiometriaResultados()
97                                     : resultadosCtrl.get(fichaAtencionId).getBody();
98         }
99     }

```

Figura 7.10: Pantalla backend código controlador audiometría

```

13 @PrimaryKeyJoinColumn(name = "ID_PERSONA")
14 @Table(name = "CONTACTO", uniqueConstraints = @UniqueConstraint(columnNames = { "ClienteId", "serial" }) )
15
16 public class Contacto extends Persona {
17
18     private static final long serialVersionUID = 1L;
19
20     @ManyToOne
21     private TipoContacto tipoContacto;
22
23     @ManyToOne
24     @JoinColumn(name = "ClienteId")
25     private Cliente cliente;
26
27     private Long serial;
28
29     public Long getSerial() {
30         return serial;
31     }
32
33     public void setSerial(Long serial) {
34         this.serial = serial;
35     }
36
37     public TipoContacto getTipoContacto() {
38         return tipoContacto;
39     }
40
41     public void setTipoContacto(TipoContacto tipoContacto) {
42         this.tipoContacto = tipoContacto;
43     }
44

```

Figura 7.11: Pantalla backend código entidad contacto

```

822
823             $scope.trazarFondo();
824
825         }
826     });
827 });
828 });
829
830 app.directive('decimal', function ($filter) {
831     var FLOAT_REGEXP_3 = /^[+]?[$]?d+(\.\d*)?$/; //Numbers como: 1123.56
832     var FLOAT_REGEXP_4 = /^[+]?[$]?d+(\,\d*)?$/; //Numbers como: 1123,56
833
834     return {
835         require: 'ngModel',
836         link: function (scope, elm, attrs, ctrl) {
837             ctrl.$parsers.unshift(function (viewValue) {
838                 if (FLOAT_REGEXP_3.test(viewValue)) {
839                     ctrl.$setValidity('float', true);
840                     if (viewValue.split('.')[1] && viewValue.split('.')[1].length >= attrs.decimal) {
841                         ctrl.$setViewValue(parseFloat(viewValue).toFixed(attrs.decimal));
842                         ctrl.$render();
843                         return parseFloat(viewValue).toFixed(attrs.decimal);
844                     } else {
845                         ctrl.$setViewValue(viewValue);
846                         ctrl.$render();
847                         return parseFloat(viewValue).toString();
848                     }
849                 } else if (FLOAT_REGEXP_4.test(viewValue)) {
850                     ctrl.$setValidity('float', true);
851                     return parseFloat(viewValue.replace(',', '.')).toFixed(attrs.decimal);
852                 } else {
853                     ctrl.$setValidity('float', false);
854                     return undefined;
855                 }
856             });
857             ctrl.$formatters.unshift(
858                 function (modelValue) {
859                     if (modelValue === undefined || modelValue === null || modelValue === '') return;
860                     return parseFloat(modelValue);
861                 }
862             );
863         }
864     });
865 });
866
867 app.directive('rangoDecimal', ['$rootScope', '$parse', function ($rootScope, $parse) {
868     return {
869         require: 'ngModel'.

```

Figura 7.12: Pantalla frontend código principal

```

1  'use strict';
2  app.controller('EspirometriaCtrl', function ($scope, $modalInstance, $alert, Flash, $state,
3   ImageService, TareaService, tarea, data, formUrl, taskClaimed, EspirometriaPDFService, $wi
4   $uibModal, $http) {
5   $scope.data = angular.copy(data);
6   // Obtener datos de triaje (peso, talla, etc)
7   $scope.datosTriage = data.datosTriage;
8   // borrar datos de triaje
9   delete data.datosTriage;
10  $scope.orden = angular.copy(data.orden);
11  usSpinnerService.stop('spinner-1');
12  $scope.formUrl = formUrl || '';
13  $scope.titulo = 'Ficha Espirometria';
14  $scope.tarea = tarea;
15  $scope.ordenAtencion = data;
16  $scope.empresa = data.empresa;
17  $scope.fichaAtencion = data.fichaAtencion;
18  $scope.paciente = data.fichaAtencion.paciente;
19
20  var atencion = data.atencion;
21
22  $scope.espirometria = angular.copy(data);
23
24  $scope.usuario = $window.user.perfil.toLowerCase();
25
26  $scope.espirometria.imagen = $scope.espirometria.imagen === null ? {} : $scope.espirom
27  $scope.peso = 0;
28
29  if (!angular.equals($scope.espirometria.imagen, {})) {
30    EspirometriaPDFService.get(data.fichaAtencion.id).then(function (resp) {
31      $scope.espirometria.imagen = resp.data;
32      var blob = base64toBlob($scope.espirometria.imagen.base64.replace("data:application/
33      var fileURL = URL.createObjectURL(blob);
34      $scope.pdfUrl = fileURL;
35
36      $scope.peso = blob.size / 1024 / 1024;
37    });
}

```

Figura 7.13: Pantalla frontend código controlador espirometría

```

1  'use strict'
2  app.service('EspirometriaPDFService', function ($http) {
3
4    this.get = function (atencionId) {
5      return $http({
6        method: 'GET',
7        url: '/api/atenciones/' + atencionId + '/espirometria'
8      });
9    }
10   this.save = function (atencionId, archivo) {
11     if (archivo.id !== undefined) {
12       return this.modify(atencionId, archivo);
13     };
14     return $http({
15       method: 'POST',
16       url: '/api/atenciones/' + atencionId + '/espirometria',
17       data: JSON.stringify(archivo),
18       headers: {
19         'Accept': 'application/json',
20         'Content-Type': 'application/json'
21       }
22     });
23   }
24
25   this.modify = function (atencionId, archivo) {
26     return $http({
27       method: 'PUT',
28       url: '/api/atenciones/' + atencionId + '/espirometria',
29       data: JSON.stringify(archivo),
30       headers: {
31         'Accept': 'application/json',
32       }
33     });
}

```

Figura 7.14: Pantalla frontend código comunicación con el backend

## AUDIOMETRIA

**DATOS PERSONALES**

NOMBRES:	Juan	APELLIDOS:	Perez Garcia
EDAD: 24 años	FECHA DE NACIMIENTO: 1993-04-24	SEXO: (M)	DNI: 73528958

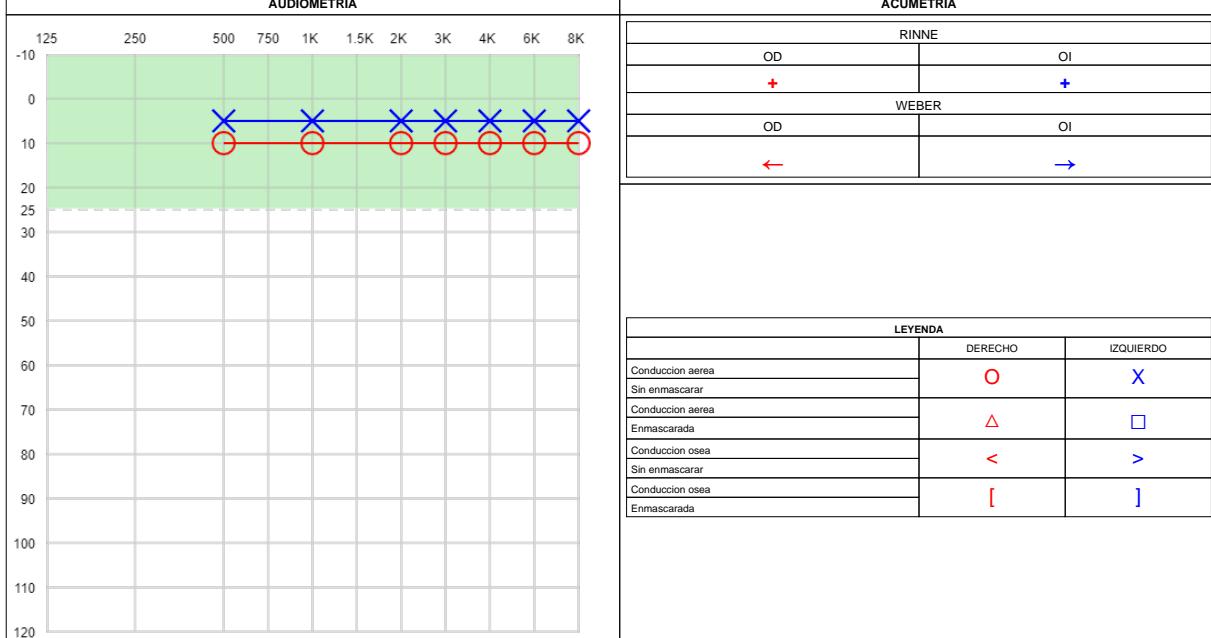
OCUPACION ACTUAL: administrador

**FECHA DEL EXAMEN: 2017-12-22**

REQUISITOS PARA PASAR AUDIOMETRIA	SI	NO	ANTECEDENTES RELACIONADOS	SI	NO
He tenido el cambio de altitud geografica mayor a 1000 msnm en las ultimas 24 horas		X	Consumo de tabaco		X
Se ha expuesto a ruido las ultimas 16 horas		X	Antecedentes de exposicion o ruidos en el trabajo		X
Presenta algun proceso infeccioso o inflamatorio del tracto respiratorio alto		X	Servicio militar y/o practica de tiro		X
Descanso 8 horas a mas	X		Hobbies con exposicion a ruido		X
Ha consumido alcohol las ultimas 24 horas		X	Exposicion laboral a quimicos		X
Usa medicamentos ototoxicos		X	Infecciones de oido		X
TIENE QUE REGRESAR (FECHA)			<b>SINTOMATOLOGIA ACTUAL</b>		
PROTECCION DE AREAS DE RUIDO			Disminucion de la audicion		X
Tapones	X		Otalgia		X
Orejeras	X		Acufenos		X
Exposicion al ruido		X	Vertigo		X
Tiempo de exposicion al ruido por dia			Secrecion otica		X
Años trabajando con ruido			Infeccion de oido		X

**AUDIOMETRIA ACTUAL:**

CONDUCTO AUDITIVO	OD	Normal		OI	Normal	
TIMPANOS	OD	Normal		OI	Normal	

**CONCLUSIONES (Segun klokoff): KLOCKOFF MODIFICADO - Otras Hipoacusias**

DETERIORO AUDITIVO			RECOMENDACIONES
PARCIAL	OIDO DERECHO: 0.0%	GLOBAL: 0,00% (% DE PRIORIDADES DE BIAURAL)	

## RADIOGRAFIA TORAX OIT

LECTURA DE RADIOGRAFIA DE TORAX UTILIZANDO LA CLASIFICACION INTERNACIONAL DE LA OIT DE RADIOGRAFIAS DE NEUMOCONIOSIS

Nº DE PLACA:	ME00000376						DNI: 73528958			Fecha del Examen: 2017-12-22							
Nombres y Apellidos:	Juan, Perez Garcia						Edad: 24 años			Sexo: M							
Empresa:	JULSA						Fecha de Lectura: 2017-11-22										
I. Calidad Radiografica	1.	Buena	X	Causas	1.	Sobreexposición		5.	Escapulas								
	2.	Aceptable			2.	Subexposición		6.	Artefacto								
	3.	Baja Calidad			3.	Posición Centrado		7.	Otros								
	4.	Inaceptable			4.	Inspiración Insuficiente		8.	-								
	Comentario sobre defectos Técnicos																
II. ANORMALIDADES PARENQUIMATOSAS (si NO hay anormalidades parenquimatosas pase a III. A. Pleurales.)										SI	X						
(2.1. Zonas Afectadas: Marque TODAS las zonas afectadas)			2.2 Profusión: (Opacidades pequeñas, es de 12 puntos, consulte las radiografías estandar marque la subcategoria de profusión)				2.3. Forma y Tamaño: (Consulte las radiografías estandar; Se requiere dos simbolos; Marque un primario y un secundario)				2.4. Opacidades Grandes: (Marque 0 si no hay ninguna o marque A, B y C)						
	Der.	Izq.	0/-	0/0	0/1	Primaria		Secundaria		0							
Superior			1/0	1/1	1/2	p	s	p	s	A							
Medio			2/1	2/2	2/3	q	t	q	t	B							
Inferior			3/1	3/2	3/3	r	u	r	u	C							
III. ANORMALIDADES PLEURALES (Si no hay anormalidades pase a simblos)										SI	X						
3.1. Placas Plaurales (0= Ninguna, D=Hemitorax derecho, I=Hemitorax Izquierdo)																	
Sitio Marque las casillas adecuadas			Clasificación			Extensión (Pared Toráxica; Combinada para placas de perfil y de frente)				Ancho (opcional) (Ancho mínimo exigido 3 mm)							
						1	< 1/4 de la pared lateral del tórax				a	De 3 a 5 mm					
						2	Entre 1/4 y 1/2 de la pared lateral del tórax				b	De 5 a 10 mm					
						3	> 1/2 de la pared lateral del tórax				c	De a 10 mm					
Pared Toraxica de perfil	0	D	I	0	D	I	O	D	O	I	D	I					
De frente	0	D	I	0	D	I	1	2	3	1	2	3					
Diaphragma	0	D	I	0	D	I	a	b	c	a	b	c					
Otros(s)	0	D	I	0	D	I	Obtencion del Angulo Costofrenico						O	D	I		
3.2. Engrosamineto Difuso de la Pleura (0= Ninguna, D=Hemitorax derecho, I=Hemitorax Izquierdo)																	
Pared Toraxica				Calsificación			Extensión				Ancho						
De perfil	O	D	I	O	D	I	O	D	O	I	D	I					
De frente	O	D	I	O	D	I	1	2	3	1	2	3					
IV. SIMBOLOS												SI	X				
aa	at	ax	bu	ca	cg	cn	co	cp	cv	di	ef	em	es	od			
fr	hi	ho	id	ih	kl	me	pa	pb	pi	px	ra	rp	tb				

CONCLUSIÓN: habiendo realizado la lectura de la radiografía de tórax ME00000376 con metodología OIT, utilizando el set completo de Radiografía OIT como patrón, se concluye:

Descripción de conclusión : conclusionessss

# Bibliografía

---

- [Çalışkan and Sevindik, 2015] Çalışkan, M. and Sevindik, K. (2015). *Beginning Spring*. Wrox, 1 edition.
- [Bhaumik, 2015] Bhaumik, S. (2015). *Bootstrap Essentials*. Packt Publishing, 1 edition.
- [Branas, 2014] Branas, R. (2014). *AngularJS Essentials*. Packt Publishing, 1 edition.
- [Bruedgge and Dutoit, 2002] Bruedgge, B. and Dutoit, A. H. (2002). *Ingeniería del software orientado a objetos*. Prentice Hall, 1 edition.
- [Cantelon and Harter, 2014] Cantelon, M. and Harter, M. (2014). *Node.js In Action*. Manning, 1 edition.
- [Chandermani, 2015] Chandermani (2015). *AngularJS by Example*. Packt Publishing, 1 edition.
- [Freeman, 2014] Freeman, A. (2014). *Pro AngularJS*. Apress, 1 edition.
- [Gutierrez, 2014] Gutierrez, F. (2014). *Introducing Spring Framework: A Primer*. Apress, 1 edition.
- [Hahn, 2016] Hahn, E. (2016). *Express in Action*. Manning, 1 edition.
- [Haviv, 2014] Haviv, A. (2014). *MEAN Web Development*. Packt Publishing, 1 edition.
- [Pilone and Miles, 2008] Pilone, D. and Miles, R. (2008). *Head first Software development*. O'Reilly Media, 1 edition.
- [Pressman, 2006] Pressman, R. (2006). *Ingeniería del software*. Mc Graw Hill, 6 edition.
- [Scott, 2016] Scott, E. (2016). *SPA Design and Architecture*. Manning, 1 edition.
- [Shenoy and Sossou, 2014] Shenoy, A. and Sossou, U. (2014). *Learning Bootstrap*. Packt Publishing, 1 edition.
- [Somerville, 2011] Somerville, I. (2011). *Ingeniería de software*. Addison-Wesley, 9 edition.
- [Varanasi and Belida, 2015] Varanasi, B. and Belida, S. (2015). *Spring REST*. Apress, 1 edition.
- [Walls, 2016] Walls, C. (2016). *Spring Boot in Action*. Manning, 1 edition.

## BIBLIOGRAFÍA

---

[Whitten and Bentley, 2008] Whitten, J. L. and Bentley, L. D. (2008). *Análisis de sistemas, diseño y métodos*. Mc Graw Hill, 9 edition.

[Wilson, 2013] Wilson, J. (2013). *Node.js the Right Way*. The Pragmatic Programmers, 1 edition.