Neiva, _____13 de mayo de 2019_____

Señores

CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD SURCOLOMBIANA

Ciudad

El (Los) suscrito(s):

_____Nelson Camilo Quinayas_____, con C.C. No. _____1.082.772.124_____,

_____, con C.C. No. _____,

_____, con C.C. No. _____,

_____, con C.C. No. _____,

autor(es) de la tesis y/o trabajo de grado o _____

titulado____ Diseño e implementación de un sistema de información para la optimización de las pruebas panel metabólico básico: electrolitos, glucosa, nitrógeno de urea y creatinina_____;

presentado y aprobado en el año ___2019____ como requisito para optar al título de

___Ingeniero Electrónico_____;

Autorizo (amos) al CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN de la Universidad Surcolombiana para que con fines académicos, muestre al país y el exterior la producción intelectual de la Universidad Surcolombiana, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios puedan consultar el contenido de este trabajo de grado en los sitios web que administra la Universidad, en bases de datos, repositorio digital, catálogos y en otros sitios web, redes y sistemas de información nacionales e internacionales "open access" y en las redes de información con las cuales tenga convenio la Institución.

- Permita la consulta, la reproducción y préstamo a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato Cd-Rom o digital desde internet, intranet, etc., y en general para cualquier formato conocido o por conocer, dentro de los términos establecidos en la Ley 23 de 1982, Ley 44 de 1993, Decisión Andina 351 de 1993, Decreto 460 de 1995 y demás normas generales sobre la materia.

- Continúo conservando los correspondientes derechos sin modificación o restricción alguna; puesto que de acuerdo con la legislación colombiana aplicable, el presente es un acuerdo jurídico que en ningún caso conlleva la enajenación del derecho de autor y sus conexos.

De conformidad con lo establecido en el artículo 30 de la Ley 23 de 1982 y el artículo 11 de la Decisión Andina 351 de 1993, "Los derechos morales sobre el trabajo son propiedad de los autores" , los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

EL AUTOR/ESTUDIANTE:

Firma: 

EL AUTOR/ESTUDIANTE:

Firma: _____

EL AUTOR/ESTUDIANTE:

Firma: _____

EL AUTOR/ESTUDIANTE:

Firma: _____

TÍTULO COMPLETO DEL TRABAJO: Diseño e implementación de un sistema de información para la optimización de las pruebas panel metabólico básico: electrolitos, glucosa, nitrógeno de urea y creatinina.

AUTOR O AUTORES:

Primero y Segundo Apellido	Primero y Segundo Nombre
Quinayas	Nelson Camilo

DIRECTOR Y CODIRECTOR TESIS:

Primero y Segundo Apellido	Primero y Segundo Nombre
Cortes Cabezas	Albeiro

ASESOR (ES):

Primero y Segundo Apellido	Primero y Segundo Nombre

PARA OPTAR AL TÍTULO DE: Ingeniero Electrónico

FACULTAD: Ingeniería

PROGRAMA O POSGRADO: Ingeniería Electrónica

CIUDAD: Neiva **AÑO DE PRESENTACIÓN: 2019** **NÚMERO DE PÁGINAS: 120**

TIPO DE ILUSTRACIONES (Marcar con una **X**):

Diagramas_**X**__ Fotografías___ Grabaciones en discos___ Ilustraciones en general_**X**__ Grabados___
Láminas___ Litografías___ Mapas___ Música impresa___ Planos___ Retratos___ Sin ilustraciones ___
Tablas o Cuadros_**X**_

SOFTWARE requerido y/o especializado para la lectura del documento:

MATERIAL ANEXO:

PREMIO O DISTINCIÓN *(En caso de ser LAUREADAS o Meritoria)*:

PALABRAS CLAVES EN ESPAÑOL E INGLÉS:

	Español	Inglés		Español	Inglés
1.	Sistema de información	Information system	6.	____FHIR_____	___FHIR_____
2.	_____Spring_____	_____Spring_____	7.	_Servicios web_	_Web service_
3.	____Android_____	____Android_____	8.	_____	_____
4.	__Base de datos__	____Database____	9.	_____	_____
5.	_____Java_____	_____Java_____	10.	_____	_____

RESUMEN DEL CONTENIDO: (Máximo 250 palabras)

Este proyecto desea permitir a las entidades de salud satisfacer la necesidad de gestionar de manera ágil y segura la información del examen panel metabólico básico y la de sus usuarios, a la cual estos pueden acceder mediante el uso de una aplicación web y una aplicación para Android. Dependiendo del tipo de usuarios, tendrán acceso a un rango distinto de acciones disponibles.

Para lograr este objetivo se llevó a cabo el desarrollo e implementación de las aplicaciones; la aplicación web server basada en tecnologías web utilizando Spring Framework el cual facilitó la implementación del proyecto con arquitectura MVC y permitió integrar seguridad y otros conceptos y por otro lado una aplicación móvil para SO Android que sirve de apoyo al funcionamiento de la plataforma web con la elección de las herramientas necesarias para su desarrollo.

Se realizó la codificación de las aplicaciones en donde se implementa la funcionalidad requerida, se establecen los algoritmos para realizar la conexión y envió de datos con el estándar propuesto por HL7 FHIR para el intercambio de información clínica entre sistemas.

A lo largo del documento se presenta el proceso de desarrollo del sistema, exponiendo una descripción de características y requisitos principales, todas las tecnologías utilizadas en el desarrollo de las herramientas,

tanto del lado cliente como del lado del servidor, así como un análisis que recorrerá todas las fases del proyecto desde el diseño hasta las pruebas.

ABSTRACT: (Máximo 250 palabras)

This project wants to allow health entities to satisfy the need to manage in an agile and safe way the basic metabolic panel exam information and that of its users, which they can access through the use of a web application and an application for Android. Depending on the type of users, they will have access to a different range of available actions.

To achieve this goal, the development and implementation of the applications was carried out; the web server application based on web technologies using the Spring Framework, which facilitated the implementation of the MVC architecture project and allowed the integration of security and other concepts and on the other hand a mobile application for Android OS that supports the functioning of the web platform with the choice of the tools necessary for its development.

The codification of the applications where the required functionality is implemented is performed, the algorithms are established to connect and send data with the standard proposed by HL7 FHIR for the exchange of clinical information between systems.

The process of developing the system is presented throughout the document, setting out a description of the main characteristics and requirements, all the technologies used in the development of the tools, both on the client side and on the server side, as well as an analysis that will run through all phases of the project from design to testing.

APROBACION DE LA TESIS

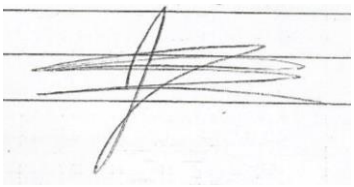
Nombre Presidente Jurado:

Firma:

Nombre Jurado: Martin Diomedes Bravo Obando

Firma: 

Nombre Jurado: Jesús David Quintero Polanco

Firma: 

UNIVERSIDAD
SURCOLOMBIANA

Universidad Surcolombiana
Pregrado Universitario en Ingeniería Electrónica

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE INFORMACIÓN PARA LA OPTIMIZACIÓN DE LAS PRUEBAS PANEL METABÓLICO BÁSICO: ELECTROLITOS, GLUCOSA, NITRÓGENO DE UREA Y CREATININA.

Trabajo Fin de Grado

Presentado por: Quinayas, Nelson Camilo

Director: Cortés Cabezas, Albeiro

Neiva
2018

Resumen

Este proyecto desea permitir a las entidades de salud satisfacer la necesidad de gestionar de manera ágil y segura la información del examen panel metabólico básico y la de sus usuarios, a la cual estos pueden acceder mediante el uso de una aplicación web y una aplicación para Android. Dependiendo del tipo de usuarios, tendrán acceso a un rango distinto de acciones disponibles.

Para lograr este objetivo se llevó a cabo el desarrollo e implementación de las aplicaciones; la aplicación web server basada en tecnologías web utilizando Spring Framework el cual facilitó la implementación del proyecto con arquitectura MVC y permitió integrar seguridad y otros conceptos y por otro lado una aplicación móvil para SO Android que sirve de apoyo al funcionamiento de la plataforma web con la elección de las herramientas necesarias para su desarrollo.

Se realizó la codificación de las aplicaciones en donde se implementa la funcionalidad requerida, se establecen los algoritmos para realizar la conexión y envió de datos con el estándar propuesto por HL7 FHIR para el intercambio de información clínica entre sistemas.

A lo largo del documento se presenta el proceso de desarrollo del sistema, exponiendo una descripción de características y requisitos principales, todas las tecnologías utilizadas en el desarrollo de las herramientas, tanto del lado cliente como del lado del servidor, así como un análisis que recorrerá todas las fases del proyecto desde el diseño hasta las pruebas.

Palabras Clave: Spring Framework, FHIR, Web Service.

Abstract

This project wants to allow health entities to satisfy the need to manage in an agile and safe way the basic metabolic panel exam information and that of its users, which they can access through the use of a web application and an application for Android. Depending on the type of users, they will have access to a different range of available actions.

To achieve this goal, the development and implementation of the applications was carried out; the web server application based on web technologies using the Spring Framework, which facilitated the implementation of the MVC architecture project and allowed the integration of security and other concepts and on the other hand a mobile application for Android OS that supports the functioning of the web platform with the choice of the tools necessary for its development.

The codification of the applications where the required functionality is implemented is performed, the algorithms are established to connect and send data with the standard proposed by HL7 FHIR for the exchange of clinical information between systems.

The process of developing the system is presented throughout the document, setting out a description of the main characteristics and requirements, all the technologies used in the development of the tools, both on the client side and on the server side, as well as an analysis that will run through all phases of the project from design to testing.

Keywords: Spring Framework, FHIR, Web Service.

Índice de contenidos

Índice de figuras

Índice de tablas

1 Introducción

1.1 Justificación

Las herramientas y medios electrónicos a lo largo de su desarrollo han sido empleados por las instituciones públicas, privadas, organizaciones e individuos para almacenar y ordenar información, comunicarse y ofrecer productos y servicios a los usuarios.

Una función principal en las empresas es dar a conocer sus servicios por medio del uso de internet, deben ser competitivas en el ámbito funcional y de acceso de los usuarios a los servicios que ofrece, por tal motivo el diseño e implementación de un módulo para un sistema de información para la optimización de las pruebas panel metabólico básico, ya que en la actualidad se ve la necesidad de poseer un sistema de información en el cual interactúen con sus usuarios y así prestar un servicio necesario para estos, teniendo un procedimiento apropiado al momento de almacenar, organizar y procesar la información adquirida de las pruebas panel metabólico básico de los pacientes para una presentación actualizada, relevante y de fácil acceso para los mismos, capaz de ofrecer un sistema para gestionar la información médica personalizada con acceso desde cualquier lugar y desde cualquier dispositivo tecnológico, tanto para los médicos como para los pacientes, controlando a qué información puede acceder cada uno de estos usuarios del sistema, y cumpliendo con la regulación existente.

Mediante internet se difunde mucha información útil y genera una buena comunicación con sus usuarios. Lo que pretende el diseño e implementación de un módulo para un sistema de información para la optimización de las pruebas panel metabólico básico, es mejorar la accesibilidad a la información clínica de los usuarios, tanto para los pacientes como para los administrativos de la empresa prestadora de salud que lo posea, con solo hacer uso del celular inteligente, Tablet o computador ahorrando tiempo, dinero, y optimizando la agilidad y accesibilidad.

La tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada vez más acusada (Telefonica, 2008).

1.2 Estructura de la memoria

A continuación, se describe brevemente cada uno de los capítulos que componen esta memoria:

- **Introducción**

En este capítulo se realiza la presentación del proyecto, se explica cuál es la motivación por la cual se lleva a cabo, así como los principales objetivos que se desea lograr a través de él.

- **Contexto y estado del arte**
  En este capítulo se pretende investigar, consultar información relacionada al proyecto que permita identificar un marco general de la problemática aquí planteada y las herramientas ofrecidas hasta el momento para su solución.
- **Objetivos y Metodología**
  En este capítulo se plantea los objetivos que se pretenden alcanzar con el desarrollo de este proyecto, es decir, donde se quiere llegar y se describirá la metodología que planteará el cómo se van a lograr estos objetivos.
- **Desarrollo de la contribución**
  En este capítulo se describen las actividades que se llevan a cabo durante el proceso de desarrollo, la identificación de requerimientos, el diseño de la aplicación, la codificación y las pruebas de la propuesta.
- **Conclusiones y trabajo futuro**
  En este capítulo se realiza una valoración del proyecto y de los objetivos alcanzados y se detallan mejoras que pueden llevarse a cabo en un futuro.

2 Contexto y Estado del arte

2.1 Antecedentes

En el proyecto se plantea el desarrollo de un sistema como solución al problema antes descrito, ya que en la actualidad algunas de las entidades manejan un sistema cerrado lo cual indica que solo acceden a la información digital el personal médico y administrativo de estas, además de que su uso solo se puede hacer dentro de la entidad. Por otro lado existen pocas instituciones que manejan estos sistemas abiertos donde los usuario pueden acceder a la información de resultados de exámenes y usuarios; sin embargo este trabajo plantea el desarrollo de una herramienta que permite la administración de información y el primer valor agregado es que la herramienta será totalmente gratuita de libre uso y disponible para todas aquellas organizaciones que requieran de una solución para la administración de resultados de exámenes y usuarios, y abierta a todo el público que tenga relación alguna con el examen panel metabólico básico, ya que en el mercado existen herramientas que cumplen con algunas funcionalidades pero son de pago.

A continuación, como conocimiento general se dan a conocer algunos trabajos y algunas de las soluciones existentes en el mercado y sus características más destacadas:

2.1.1 Baxlab

BAXLAB Software de Gestión para Laboratorio Clínico, es un sistema destinado a organizar, controlar y automatizar la información en los laboratorios clínicos. (Guia Soluciones TIC, 2018)

El Software para Laboratorio Clínico BAXLAB registra la información de los pacientes, a quienes ordenen estudios de laboratorio:

- Trazabilidad de las muestras (Aceptadas o Rechazadas).
- Reportes.
- Validación por Bacteriólogo.
- Consulta e impresión de resultados.
- Descarga de resultados por Entidad, Laboratorios Referenciales, Médico y Pacientes desde la Web del laboratorio.

2.1.2 SysLabs

Es un software de escritorio diseñado para automatizar la gestión administrativa y apoyar las operaciones del Laboratorio Clínico. (SYSLAB Ingeniería, 2018)

Funcionalidades de laboratorio en el SysLabs:

- Recepción de paciente.
- Registro de solicitud de estudios.
- Entrega de comprobante de pago o factura.
- Captura de resultados en el sistema.
- Impresión y/o publicación de resultados vía Internet.
- Registro de entrega de resultados.
- Generación de reportes de control.

2.1.3 Aplicación web para laboratorio clínico del centro de salud #1

Es un trabajo de titulación para la obtención del título de Licenciada en sistemas de información de la universidad de Guayaquil-Ecuador presentado por CINTHYA JAZMÍN LOOR CANTOS en el año 2015 denominado SILAB, que implementa MVC utilizando lenguajes de programación como html, php y MySQL como motor de base de datos. (Repositorio Institucional de la Universidad de Guayaquil, 2015)

Las funcionalidades principales de la aplicación desarrollada son:

- Registro de usuarios.
- Permite conocer resultados, valores normales, significados anormales, valores estadísticos e historial de exámenes clínicos del paciente en base sus parámetros.
- Muestra los resultados en tiempo real utilizando internet como medio de acceso a la información procesada.

2.1.4 Diseño e implementación de un aplicativo móvil para pacientes de urología en la fundación santa fe de Bogotá. Dimic

Es un Proyecto de grado para optar por el título de Ingeniero de sistemas y computación de la universidad de los andes Bogotá-Colombia presentado por Viviana Paola Salcedo Saavedra, Danny Leandro Hurtado Flechas, Gustavo Alberto Ávila Cruz, Alejandro Polania Gutiérrez y Rafael Aranda López King en el año 2015. (Universidad de los Andes repositorio Institucional, 2014)

La solución propuesta consiste en un sistema que permite registrar las actividades de interés para los médicos, en este caso son las ingestas de líquido, las micciones y los escapes de orina presentados por el paciente. En particular la solución debe ser una aplicación para plataformas móviles, que en forma intuitiva ayude al paciente a tomar el registro cuando el evento está sucediendo, sus principales funciones son:

- Registrar los datos principales por un periodo de tiempo específico.
- Consolidar algunos resúmenes de información y métricas para generar el reporte.
- Realizar cuestionarios.
- Permitir seleccionar entre un diario miccional pediátrico y un diario miccional adulto.
- Enviar reportes por correo electrónico a la fundación y/o a un correo en específico.

3 Objetivos y Metodología

3.1 Objetivos

El objetivo del proyecto es diseñar e implementar un sistema de información para la optimización de las pruebas panel metabólico básico: electrolitos, glucosa, nitrógeno de urea y creatinina destinada para usuarios pacientes y personal médico relacionados con dichas pruebas.

Los objetivos específicos del proyecto son los siguientes:

- Identificar los requerimientos del módulo de información.
- Diseñar e implementar la aplicación web server para registro y acceso a la información de las variables de prueba del panel básico metabólico con acceso desde cualquier lugar y desde cualquier dispositivo tecnológico con conexión a internet, tanto para los administrativos como para los pacientes, controlando a qué información puede acceder cada uno de estos.
- Diseñar e implementar una aplicación móvil para la plataforma Android para registro y acceso a la información de la prueba panel metabólico básico, con acceso desde cualquier lugar y dispositivos tecnológicos compatibles y con conexión a internet, tanto para los administrativos como para los pacientes, controlando a qué información puede acceder cada uno de estos.
- Realizar pruebas para validar y verificar el correcto funcionamiento del módulo para el sistema de información.

3.2 Selección de la Metodología

Se propone una metodología de 4 fases que se adapta a las fases del modelo RUP, tomando las actividades necesarias para este proyecto. Se propone una metodología basada en RUP debido a la flexibilidad que tiene este de adaptarse a cada proyecto, haciendo uso de buenas prácticas en el desarrollo de software, como desarrollo iterativo, administración eficiente de requerimientos y prototipos incrementales.

El enfoque metodológico seguido queda resumido en las siguientes fases:

- Especificaciones del sistema. Análisis y requerimientos.

- Desarrollo del sistema. Diseño del sistema y modelado para describir, por un lado, el dominio del problema y los requerimientos y, por otro lado, el dominio de la solución conceptual.
- Implementación. En esta fase se lleva a cabo la programación y codificación que implementan el diseño.
- Transición. Puesta en producción del sistema de información y pruebas del mismo en cuanto al cumplimiento de los requerimientos.

Como lenguaje de modelado se utilizó el Lenguaje Unificado de Modelado (UML), ya que RUP indica cómo utilizarlo efectivamente, por otra parte, UML nos permite indicar claramente los requisitos, arquitecturas y diseños.

3.3 Selección entorno de desarrollo y herramientas

Para el desarrollo del sistema que permita administrar la información de usuarios y resultados del examen panel metabólico básico, se usaron herramientas de desarrollo de software libre. Además, se pensó en el hecho de escoger unas buenas herramientas de desarrollo de software ya que es fundamental para el desarrollo de un proyecto de tal magnitud.

En esta sección se describen las herramientas de desarrollo que se utilizaron para llevar a cabo el desarrollo del proyecto.

3.3.1 Spring Tool Suite

Para el desarrollo de la aplicación web server se hace uso del IDE Spring Tool Suite en su versión 3.8.1 que nos ofrece facilidades a la hora de crear un proyecto Web dinámico basado en Spring MVC.

Spring Tool Suite proporciona un entorno ready-to-use para implementar, depurar, ejecutar y desplegar las aplicaciones, Spring soporta el despliegue de aplicaciones tanto en servidores locales, virtuales y en la nube. Es de libre acceso para el desarrollo y uso en operaciones internas sin límite de tiempo, completamente de código abierto y licencia bajo los términos de la Licencia Pública Eclipse.

Además, integra como servidor web para el despliegue de la aplicación web, por sencillez y compatibilidad de uso con Spring Tool Suite, Pivotal TC Server, en su versión Developer Edition 3.1 como servidor web de prueba.

Pivotal TC Server, trabaja como un contenedor de servlets y JSPs, al igual que Tomcat. Implementa las especificaciones de los servlets y de JSP de Sun Microsystems, como contenedor de servlets utiliza Catalina.

Otra herramienta que incluye es Maven que utiliza un Project Object Model (POM) donde se incluyen las dependencias de otros módulos, componentes externos y el orden de construcción de los elementos. Realiza también la compilación del código y su empaquetado, comprueba que el código es correcto, hasta que se despliega la aplicación, pasando por la ejecución de pruebas y generación de informes y documentación.

3.3.2 Android Studio

Para el desarrollo de la aplicación móvil se usaron herramientas oficiales para el desarrollo de aplicaciones Android, tales como el IDE Android Studio en su versión 2.3, éste cuenta con las herramientas necesarias para el diseño y desarrollo de aplicaciones en el sistema operativo Android, ya que en este software es posible elaborar en conjunto el diseño estético de la aplicación y los protocolos de comunicación con el servidor.

3.3.3 MySQL Workbench

Para el desarrollo de la base de datos se utilizó MySQL Workbench en su versión 6.3 CE que es una herramienta visual de diseño de bases de datos que integra desarrollo de software, Administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL.

3.3.4 Postman

Postman es una herramienta para el testing de APIs disponible en forma de extensión para Google Chrome que dispone de interfaz gráfica, con lo que es muy intuitiva para su uso. Nos permite construir y gestionar de una forma cómoda nuestras peticiones a servicios web mediante el uso HTTP (GET, POST, PUT, PATCH, etc.). Una vez instalada hay que definir la petición a realizar e incluir los datos que queremos enviar (solo para PUT o POST) y la URL a la que deseamos realizar la petición.

Posteriormente a la petición, muestra la respuesta de la API en el formato estándar de intercambio de la misma ya que es compatible con una gran cantidad de formatos de intercambio de datos, como puede ser XML o JSON.

3.3.5 Google Chrome

Google Chrome es un navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones (frameworks) de código abierto, se utilizó este navegador web en su versión 66.0 para la depuración de la presentación de la interfaz web gráfica de usuario y su conexión con el servidor web.

3.3.6 Creately

Creately es una de las herramientas que se encuentra en la red para elaborar mapas conceptuales y todo tipo de diagramas. Se utilizo para crear los diagramas de casos de uso, de secuencia y de clases.

4 Desarrollo de la contribución

4.1 Requerimientos y Análisis

4.1.1 Identificación de requerimientos

En esta sección se tiene como propósito definir las especificaciones funcionales y no funcionales para el desarrollo de un sistema que permitirá gestionar la información de usuarios y resultados del examen panel metabólico básico. Este sistema será utilizado por los usuarios pertenecientes a la entidad prestadora de servicios de salud que lo posea.

Además, describir la funcionalidad del usuario a lo largo de ella. Los requerimientos funcionales y no funcionales de las tablas siguientes se recopilaron al inicio por parte del desarrollador dependiendo de las historias de usuario.

4.1.1.1 Requerimientos funcionales

Un requisito funcional (RF) define una función del sistema de software o sus componentes. A continuación, se presentan los requerimientos deducidos de las historias de usuarios, en la tabla 1 se muestran.

Tabla 1. Requerimientos funcionales del sistema

Id	Nombre	Función	Prioridad
RF01	Gestión de sesión	Implementación de un módulo para validación de usuarios y permitir el acceso a la intranet del sistema.	Alta
RF02	Gestión de usuarios	Administración de información de usuarios por parte de un usuario de tipo administrador para el registro, actualización, consulta y borrado de los mismos.	Alta
RF03	Gestión de resultados	Administración de información de exámenes por parte de usuarios asignados con roles que se encargan de autorizar exámenes, registrar resultados, así como la actualización o eliminación de los mismos.	Alta

RF04	Gestión de búsqueda	Implementación de un buscador de usuarios por id para el administrador.	Media
RF05	Descarga reporte	Descarga un archivo con extensión PDF del resultado seleccionado	Media
RF06	Envió notificación	Envía una notificación manual por parte del médico indicando que el resultado está listo.	Media
RF07	Generar gráficos	El paciente puede ver una gráfica con todos los resultados de la misma variable, indicando los valores de referencia para que pueda realizar una comparación de datos.	Media

(Elaboración propia)

4.1.1.2 Requerimientos no funcionales

Se definen requerimientos no funcionales (RNF) como las características que el sistema debe tener para asegurar la calidad de los servicios prestados por el mismo. A continuación, se muestran.

- **Desempeño**
  - El sistema deberá garantizar el correcto funcionamiento del mismo a los diferentes roles de usuario. Con base en esto la información almacenada en base de datos podrá ser registrada, consultada y actualizada permanente y simultáneamente.
- **Disponibilidad**
  - El sistema deberá permanecer disponible el 100% o muy cercano.
- **Escalabilidad**
  - La arquitectura posibilitará la incorporación de nuevas funcionalidades y módulos flexiblemente sin procedimientos drásticos para el desarrollador.
- **Facilidad de uso**
  - Para la familiarización del usuario con el software se requiere una interfaz gráfica ligera e intuitiva sumada a una correcta emisión de avisos de error y advertencia.
- **Hardware**
  - Debido a que es una aplicación Web se requiere mínimo procesador.
  - La aplicación móvil requiere mínimo procesador.

- Procesamiento para servidores de aplicaciones y bases de datos se definirá a medida que los usuarios hagan uso del sistema.
- Capacidad memoria RAM servidor, mínima 1Gb.
- Capacidad almacenamiento en servidor, mínima 2 Gb.
- Para clientes con equipos de escritorio, portátiles y dispositivos móviles.

- **Software**
  - Sistemas operativos de escritorio cualquier sistema operativo que soporte un navegador web como Google Chrome, Firefox o Microsoft Edge.
  - Sistema operativo Android 4+ y aplicación para cliente móvil.
  - Motor de bases de datos MySQL.
  - Lenguaje de programación JAVA.
- **Comunicaciones**
  - La arquitectura Cliente Servidor de tres capas, siendo estas la capa aplicación, servidor de aplicaciones y servidor de base de datos.
- **Performance**
  - Garantiza un tiempo de acceso no mayor a 5 segundos.

4.1.1.3 Definición de actores y roles del sistema

Los actores corresponden a las personas que interactúan con el sistema software y se especializan en roles según sus permisos sobre el mismo.

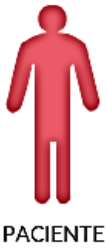
Como resultado y según los requerimientos analizados a partir de la lista de exigencias, se presenta a continuación la descripción de los actores y roles participantes en el sistema:

ADMINISTRADOR

- ✓ Administrador: Persona encargada de la gestión de usuarios del sistema.
- ✓ Responsabilidades: Ingreso seguro al sistema, registro de usuarios, modificación de información de usuarios y eliminación de usuarios.

- ✓ Médico: Persona encargada de la autorización de exámenes a los usuarios del sistema.
- ✓ Responsabilidades: Ingreso seguro al sistema, autorización de examen, realizar observaciones y eliminación de exámenes.

- ✓ Laboratorista: Persona encargada del registro de resultados.
- ✓ Responsabilidades: Ingreso seguro al sistema, registrar resultados de exámenes, realizar observaciones y editar resultados.



- ✓ Paciente: Persona que tiene relación con el examen.
- ✓ Responsabilidades: Ingreso seguro al sistema, consultar resultados de exámenes, descarga de reportes pdf y ver gráficas.

4.1.2 Análisis

4.1.2.1 Alcance del sistema

En este punto se delimita el sistema de información, se indica qué procesos pertenecen al ámbito del Sistema de Información y se identifican las entidades externas al sistema que aportan o reciben información.

La siguiente tabla, muestra el alcance definido para el sistema Web y la Aplicación móvil.

Tabla 2. Alcance Aplicaciones del sistema

Función	Aplicación web	Aplicación Android
Gestión de inicio de sesión	Si	Si
Registro de usuarios	Si	No

Actualización de usuarios	Si	No
Consulta de usuarios	Si	Si (Pacientes)
Eliminación de usuarios	Si	No
Autorización de examen	Si	Si
Registro de resultado examen	Si	Si
Consulta de examen	Si	Si
Actualización resultado examen	Si	No
Eliminación de examen	Si	No
Generación de reporte pdf	Si	No
Generación de gráficas	Si	Si

(Elaboración propia)

4.1.2.2 Viabilidad técnica y económica

En lo referente a la viabilidad técnica, cabe destacar que, desde el momento de la asignación de este proyecto, ya se dispone de los requisitos para elaborar este tipo de proyectos, a excepción de algunos conocimientos de codificación en los lenguajes de programación que se utilizan, en los cuales se trabajó arduamente.

Para la viabilidad técnica se presentan las restricciones en hardware y software con el propósito de desarrollo de la solución planteada, así como su disponibilidad. Las restricciones técnicas identificadas son las siguientes:

a) Disponibilidad del equipo computador/servidor para alojar la base de datos para pruebas durante el desarrollo.

b) Disponibilidad del equipo computador/servidor para su utilización como servidor de pruebas durante el desarrollo de aplicaciones Web.

c) Disponibilidad del equipo computador para el trabajo de análisis, diseño, construcción y pruebas.

d) Herramientas de libre distribución para el modelamiento UML y construcción de la base de datos del proyecto.

e) Herramientas IDE para la construcción de la interfaz gráfica y codificación de las funcionalidades de las aplicaciones.

f) Sistema administrador de base de datos con capacidad para soportar múltiples conexiones.

g) Disponibilidad de un servidor Web JAVA para pruebas de producción.

El proyecto es técnicamente viable porque el tesista cuenta con todos los requisitos citados. Con el propósito de aprovechar los recursos existentes al máximo, se toman las siguientes medidas:

1) Los puntos a, b, y c quedan cubiertos empleando un computador portátil con procesador Intel Pentium 4 y memoria RAM de 4GB, que se adaptan a las exigencias del servidor de base de datos, sistema operativo y para las fases de análisis, diseño, desarrollo y pruebas por parte del tesista. Esto obedece estrictamente a razones de simplificación de recursos, en entornos de trabajo reales sí se exige una separación entre servidores y de mejores características.

2) Para el punto d existe una herramienta online Createlty sujeta a las exigencias técnicas propias de la documentación con RUP y además son de uso gratis con limitaciones. Por otro lado, para la implementación de la base de datos se hace uso de MySQL Workbench 6.3 CE. Para el punto e, se integran herramientas IDE Spring Tool Suite Versión 3.8, IDE Android Studio Versión 2.3 y para el punto f se hace uso del administrador de base de datos MySQL 5.7.

3) Para el requisito del punto g, se hace uso de un servidor web que despliega aplicaciones JAVA y servidor de base de datos para realización de pruebas de producción que ofrece webhosting.com en su versión de prueba por 14 días y pruebas en localhost con Tomcat 8.5 y Pivotal TC Server 3.1.

En cuanto a la viabilidad económica, tomando como punto de partida los ítems técnicos citados para la implementación, se considera que en el proyecto:

1) Los requisitos a nivel de hardware, se excluyen asumiendo su disponibilidad por parte del tesista.

2) Las herramientas para el modelamiento UML y de la base de datos son libres de costo.

3) El IDE Spring Tool Suite, IDE Android Studio, MySQL Workbench y MySQL, se encuentran a disposición desde Internet y libre de costo.

4.1.2.3 Análisis Costo – beneficio

En este análisis se presentan las razones y criterios tomados como justificación para el desarrollo del proyecto y la inversión económica, así como el grado de contribución esperado en los procesos de gestión de información clínica en las entidades de salud tras su implantación.

Una vez expuestos los detalles del costo en el proyecto, se concluye que no hay una gran inversión en hardware y software gracias al empleo de herramientas de código libre de uso gratuito y debido a la disponibilidad del hardware por parte del tesista. La inversión se reserva para la cobertura en costos de logística del proyecto.

Por otra parte, conviene precisar las ventajas y beneficios ofrecidos por la solución. El propósito es optimizar los procesos de gestión de información de usuarios y de resultados del examen panel metabólico básico, lo cual conlleva a disminuir el tiempo en la entrega de resultados y la insatisfacción de los pacientes debido a que en ocasiones los lleva a demasiada pérdida de tiempo al realizar este proceso.

Contribuirá con la centralización de toda la información en medios digitales a disposición de los usuarios relacionados con el examen desde dispositivos tecnológicos con conexión a Internet. Las funcionalidades mencionadas cuentan con las medidas en seguridad que debe poseer sistema web estándar.

Tras el análisis efectuado, la solución presenta beneficios a futuro para todos los usuarios relacionados con el examen panel metabólico básico y a un costo razonable en inversión en una institución pública, ya que solo se necesita el servidor web en la nube para alojar la aplicación. Es viable dadas las ventajas y mejoras ofrecidas en la automatización de los procesos de gestión de la información clínica.

4.1.2.4 Identificación de las necesidades del usuario

Como principales necesidades establecidas por los usuarios de una entidad de salud se obtuvieron los siguientes alcances:

- Disponibilidad de información del resultado panel metabólico básico tanto dentro como fuera de la entidad de salud y en cualquier momento y para cualquier usuario del sistema que tenga relación con el examen.

- Acceso a tiempo completo de la información de resultados y de usuario.
- Presentación de estadísticas y generación de reportes físicos.

Estas necesidades indicadas quedan cubiertas por los requerimientos del sistema dada la similitud entre las expectativas de usuarios con las funcionalidades del nuevo sistema. A partir de la lista de exigencias y habiendo identificado las necesidades de los usuarios es factible construir los diagramas de casos de uso.

La figura 1 muestra los principales módulos de la aplicación a desarrollar, mediante diagramas de casos de uso.

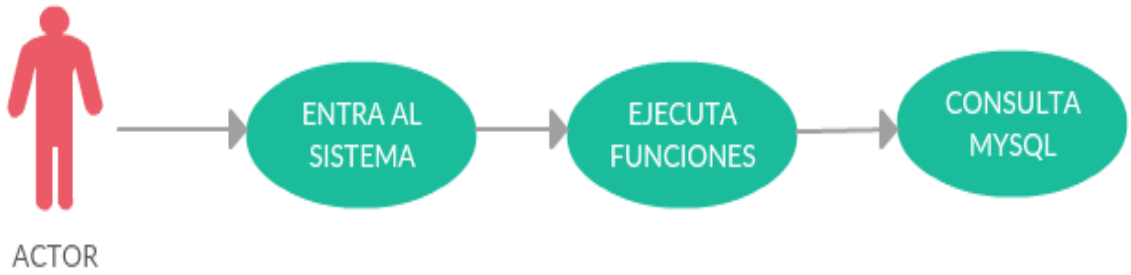


Figura 1. Casos de uso general del sistema.

Como se observa para el caso de uso general del sistema, existen cuatro tipos de usuarios (administrador, medico, laboratorista y paciente) actores principales del sistema y que pueden iniciar o terminar alguna sesión dentro de la aplicación, accediendo a los servicios que esta ofrece dependiendo del tipo de usuario.

Tabla 3. Eventos caso de uso general del sistema

Análisis caso de uso general del Sistema Eventos	
Acción del actor	Respuesta del sistema
1. Esta acción inicia cuando una persona hace uso de la aplicación como un usuario en el sistema.	2. Solicita usuario y contraseña de acceso al sistema.
3. Introduce usuario y contraseña de acceso y envía.	4. Verifica datos y re direcciona de acuerdo al tipo de usuario a la sección correspondiente.

5. El actor identificado ejecuta funciones del sistema.	6. Procesa la función ejecutada.
7. Actor cierra sesión.	8. Cierra sesión.

(Elaboración propia)

4.1.2.4.1 Diagramas casos de uso aplicación web

A continuación, se muestra los diagramas de casos de uso de los actores de la web del sistema y muestra las actividades que cada uno puede realizar en él.

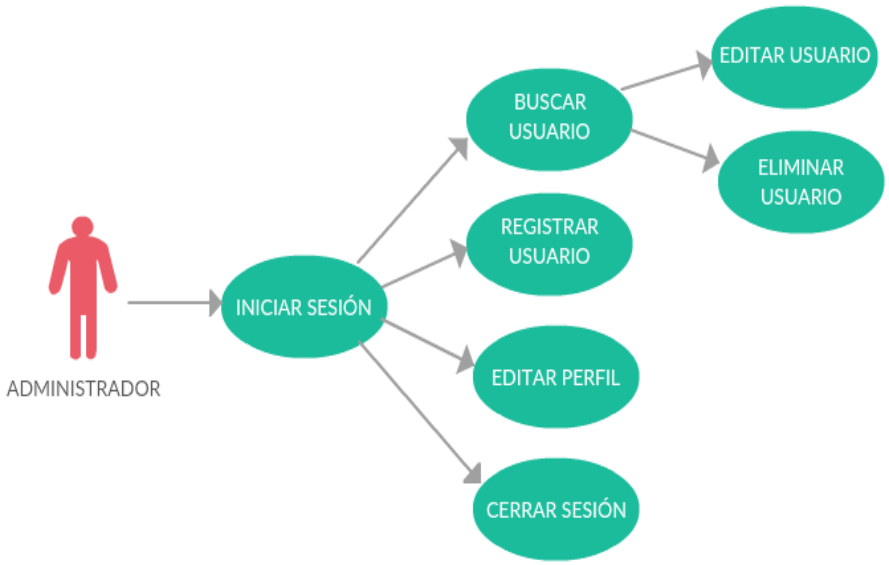


Figura 2. Diagrama casos de uso administrador.

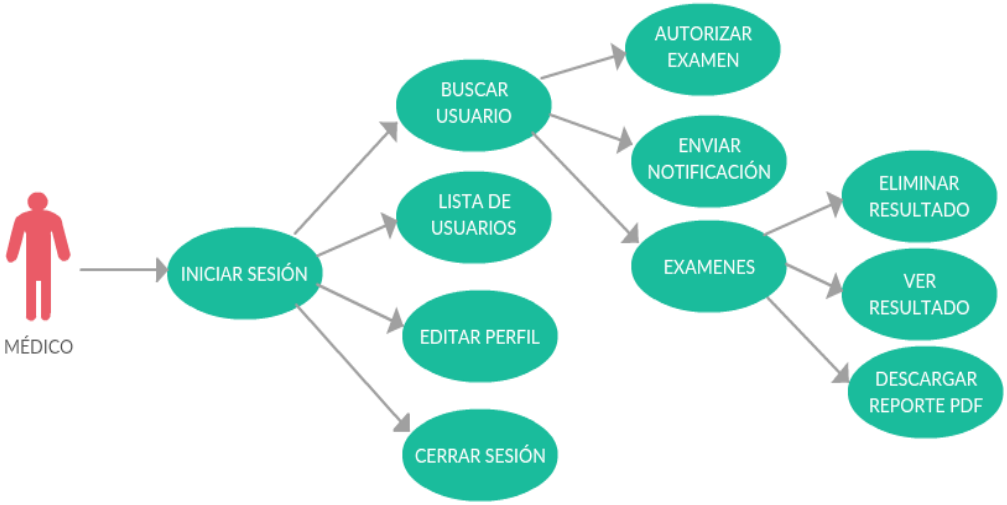


Figura 3. Diagrama casos de uso Médico.

Figura 4. Diagrama casos de uso laboratorista.

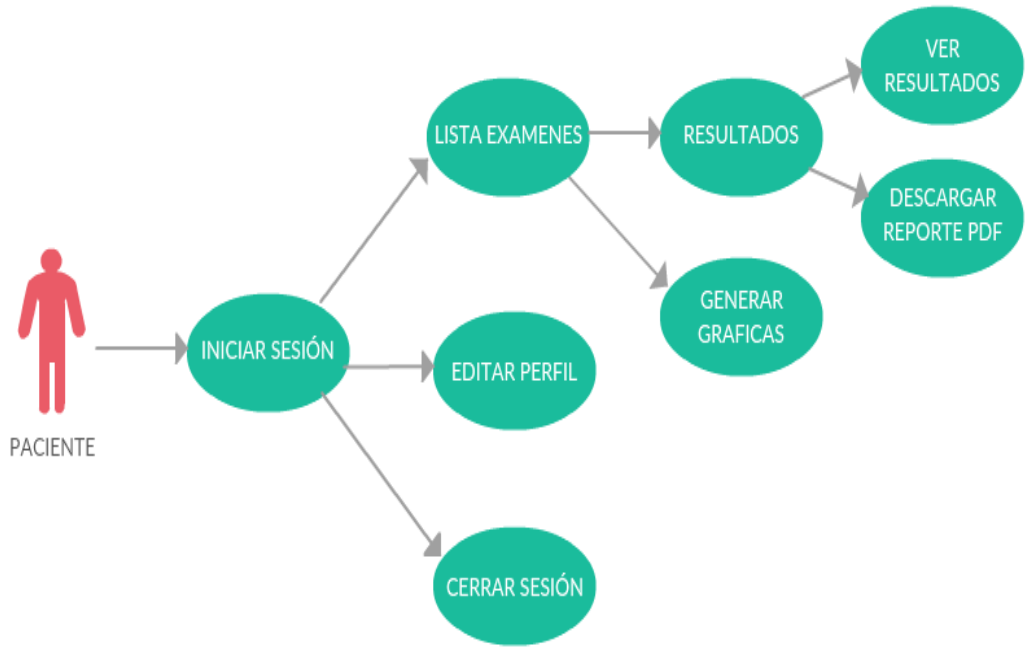


Figura 5. Diagrama casos de uso paciente.

4.1.2.4.2 Diagramas de casos de uso aplicación móvil

A continuación, se muestra los diagramas de casos de uso de los actores de la aplicación móvil del sistema y muestra las actividades que cada uno puede realizar en él.

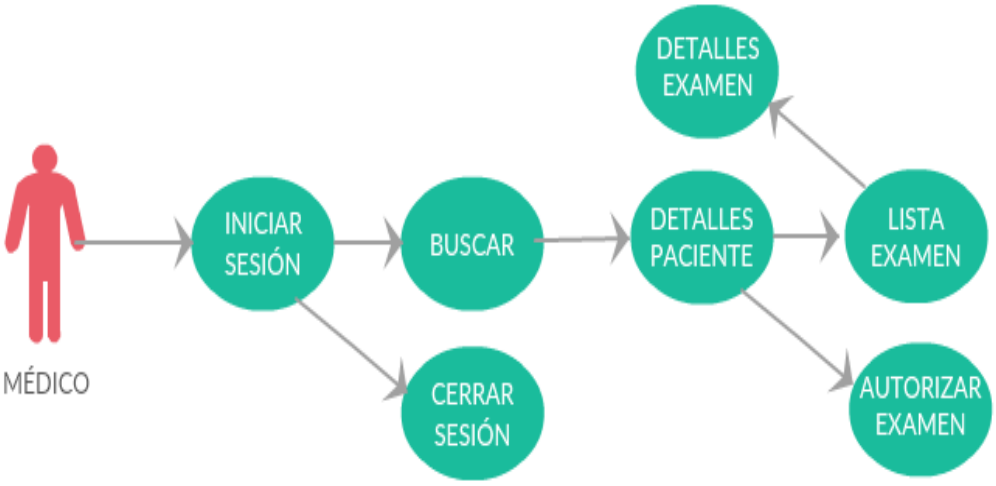


Figura 6. Diagrama casos de uso médico.

Figura 7. Diagrama casos de uso laboratorista.

Figura 8. Diagrama de casos de uso paciente.

4.1.2.4.3 Descripción de los casos de uso del sistema

A continuación, se muestra los detalles de cada uno de los casos de uso del sistema.

Tabla 4. Detalles caso de uso Iniciar Sesión

Ítem	Descripción
Caso de uso N.º 1	Iniciar sesión
Propósito	Identifica al usuario mediante número de documento y una contraseña.
Actores	Administrador, Personal Médico, Laboratorista y Pacientes
Precondiciones	Usuarios deben estar registrados.
Postcondiciones	Se crea un objeto Usuario el cual contiene los datos de la base.

(Elaboración propia)

De la tabla 4, el usuario introduce su número de documento y su contraseña, ambos son validados por el servidor de aplicación tras consultar la base de datos. Si son correctos se direcciona a la página principal con el menú correspondiente según cual sea el tipo de usuario administrador, personal médico, laboratorista o paciente.

Tabla 5. Detalles caso de uso crear usuario

Ítem	Descripción
Caso de uso N.º 2	Crear usuario
Propósito	Crea un nuevo usuario y lo guarda en la base de datos.
Actores	Administrador
Precondiciones	No debe existir en la base de datos un usuario con el mismo número de documento.
Postcondiciones	Se habrá insertado un nuevo usuario en la base de datos exitosamente.

(Elaboración propia)

De la tabla 5, el administrador deberá introducir los datos del nuevo usuario en un formulario que variará ligeramente según el tipo de usuario que desee. El id de usuario será igual al número de documento del mismo.

Tabla 6. Detalles caso de uso eliminar usuario

Ítem	Descripción
Caso de uso N.º 3	Eliminar usuario
Propósito	Elimina un usuario de la base de datos.
Actores	Administrador
Precondiciones	El usuario debe existir en la base de datos.
Postcondiciones	El usuario habrá sido eliminado de la base de datos exitosamente.

(Elaboración propia)

De la tabla 6, el administrador deberá buscar el usuario que desea borrar. Finalmente, el usuario quedará eliminado de la base de datos y toda la información asociada a él.

Tabla 7. Detalles caso de uso editar usuario

Ítem	Descripción
Caso de uso N.º 4	Editar información de usuario
Propósito	Edita la información de un usuario de la base de datos.
Actores	Administrador, Personal, Laboratorista o Paciente
Precondiciones	El usuario debe existir en la base de datos.
Postcondiciones	La información de usuario habrá sido editada de la base de datos exitosamente.

(Elaboración propia)

De la tabla 7, el administrador busca por número de documento el usuario que desee modificar la información. El administrador es el único que tiene acceso a modificar información de todos los usuarios. El usuario de tipo Personal, Laboratorista o Paciente solo podrán modificar su

información personal a excepción del tipo y número de documento. Una vez realizado los cambios podrán guardar las modificaciones en la base de datos exitosamente.

Tabla 8. Detalles caso de uso buscar

Ítem	Descripción
Caso de uso N.º 5	Buscar
Propósito	Consultará la base de datos y mostrará la información actual del usuario.
Actores	Administrador, Personal o Laboratorista
Precondiciones	El usuario debe existir en la base de datos.
Postcondiciones	La información del usuario se mostrará exitosamente.

(Elaboración propia)

De la tabla 8, el Administrador ingresa el número de documento del usuario a buscar y el sistema le devuelve la información personal de este. El personal médico ingresa número de documento del usuario y el sistema le devuelve la información personal y lista de resultados y exámenes registrados de este.

Tabla 9.detalles caso de uso recuperar contraseña

Ítem	Descripción
Caso de uso N.º 6	Recuperar Contraseña
Propósito	Envía la contraseña del usuario al correo electrónico registrado.
Actores	Administrador, Personal, Laboratorista o Paciente
Precondiciones	El usuario debe existir en la base de datos.
Postcondiciones	La contraseña del usuario habrá sido enviada al correo electrónico registrado exitosamente.

(Elaboración propia)

De la tabla 9, el usuario ingresa su número de documento y espera un mensaje de que ya fue enviada la contraseña al correo.

Tabla 10. detalles caso de uso autorizar examen

Ítem	Descripción
Caso de uso N.º 7	Autorizar Examen
Propósito	Registra un examen al usuario deseado en la base de datos.
Actores	Personal Medico
Precondiciones	Tipo de usuario personal medico
Postcondiciones	El examen se habrá registrado correctamente en la base de datos.

(Elaboración propia)

De la tabla 10, el usuario personal médico una vez realiza la búsqueda del paciente puede autorizar un examen.

Tabla 11. detalles caso de uso eliminar resultado

Ítem	Descripción
Caso de uso N.º 8	Eliminar Resultado
Propósito	Elimina de la base de datos un registro de resultado de examen.
Actores	Personal Médico
Precondiciones	Tipo de usuario personal médico
Postcondiciones	El examen se habrá eliminado correctamente en la base de datos.

(Elaboración propia)

De la tabla 11, el Personal Médico podrá eliminar un registro de un examen de un paciente.

Tabla 12. detalles caso de uso ver resultado

Ítem	Descripción
Caso de uso N.º 9	Ver resultado

Propósito	Consulta de la base de datos un registro de resultado de examen.
Actores	Personal Médico, Laboratorista o Paciente.
Precondiciones	Resultado debe existir en la base de datos
Postcondiciones	La información del resultado se mostrará correctamente de la base de datos.

(Elaboración propia)

De la tabla 12, los usuarios Personal Médico, Laboratorista o Paciente podrán visualizar los detalles del resultado del examen realizado.

Tabla 13. Detalles caso de uso registrar resultado

Ítem	Descripción
Caso de uso N.º 10	Registrar Resultado
Propósito	Registra la información de un resultado en la base de datos.
Actores	Laboratorista
Precondiciones	Tipo usuario Laboratorista
Postcondiciones	El resultado se habrá registrado correctamente en la base de datos.

(Elaboración propia)

De la tabla 13, el Laboratorista al ingresar al sistema obtendrá una lista de los exámenes autorizados por parte del personal médico, a lo cual selecciona cualquiera e ingresa el resultado y las observaciones para luego registrarlo en la base de datos.

Tabla 14. Detalles caso de uso editar resultado

Ítem	Descripción
Caso de uso N.º 11	Editar Resultado
Propósito	Edita la información de un resultado de la base de datos.

Actores	Personal Médico o Laboratorista
Precondiciones	Resultado debe existir en la base de datos.
Postcondiciones	El resultado se habrá editado correctamente en la base de datos.

(Elaboración propia)

De la tabla 14, los usuarios laboratorista podrán editar el resultado del examen al igual que las observaciones hechas por este. El usuario personal médico podrá editar las observaciones hechas por este.

Tabla 15. Detalles caso de uso ver gráficas

Ítem	Descripción
Caso de uso N.º 12	Ver gráficas
Propósito	Generar una gráfica de los valores de los resultados por variable.
Actores	Paciente
Precondiciones	Debe existir valores para los resultados.
Postcondiciones	La gráfica se generará correctamente.

(Elaboración propia)

De la tabla 15, el usuario paciente podrá seleccionar una variable del examen Panel Metabólico Básico para ver una gráfica de manera que facilite su comprensión, comparación y análisis.

Tabla 16. Detalles caso de uso generar reporte pdf

Ítem	Descripción
Caso de uso N.º 13	Generar PDF
Propósito	Generar un archivo .pdf del resultado seleccionado.
Actores	Paciente o Personal Médico.
Precondiciones	Debe existir el resultado registrado en la base de datos.

Postcondiciones	El archivo .pdf se generará correctamente y se descargará.

(Elaboración propia)

De la tabla 16, los usuarios Paciente o Personal Médico podrán generar un reporte en PDF del resultado seleccionado para descargar en su dispositivo.

4.1.2.5 Definición del sistema

La definición del sistema se presenta involucrando a las entidades principales en el modelamiento del escenario de negocio. Esto favorecerá el establecimiento y definición de la arquitectura final junto con las clases de diseño necesarias para su construcción. La solución cubre los requerimientos mediante los paquetes representados en el diagrama de paquetes de la figura 9.

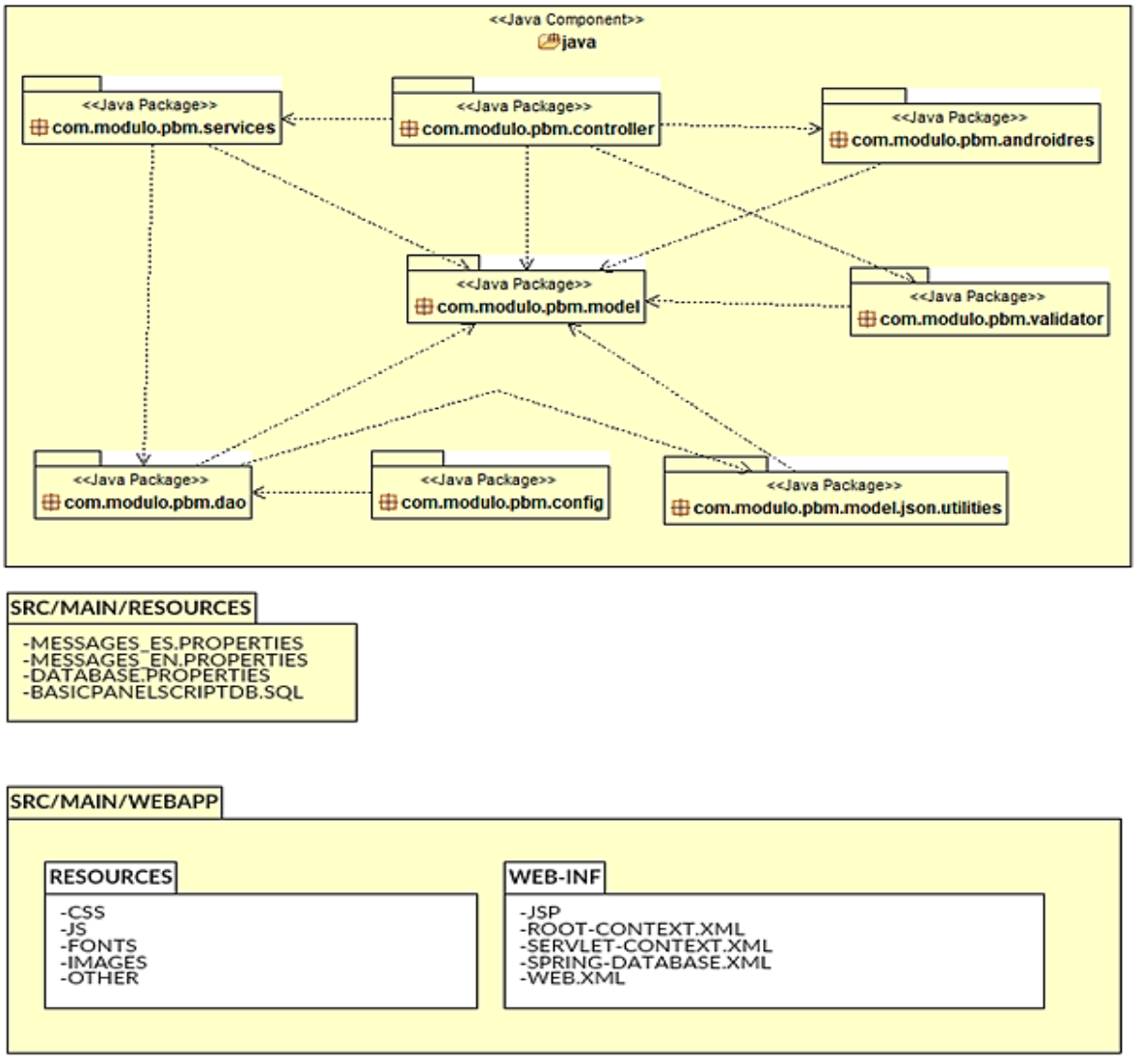


Figura 9. Diagrama de paquetes.

El diagrama de paquetes mostrado contiene los siguientes elementos:

- **resources**: Contiene los ficheros de configuración y recursos necesarios para dar estilo y funcionalidad a la página web.
- **WEB-INF**: Contiene paquetes con archivos de tipo jsp, utilizados para renderizar las distintas vistas, configuraciones de seguridad de Spring y configuración de conexión a base de datos.
- **utilities**: Contiene componentes con funciones de carácter estático.
- **validator**: Contiene componentes que permiten validar formularios de forma avanzada una vez recibida una respuesta que requiere dicha validación.
- **services**: Contiene componentes que se encargan de realizar acciones solicitadas por los controladores, procesando la información necesaria de los componentes.
- **DAO**: Componentes que siguen el patrón Data Access Object, consultas a base de datos.
- **model**: Contiene componentes que representan las entidades del sistema.
- **controller**: Contiene las clases de los controladores.
- **Androidres**: Contiene clases de los recursos Web Services para las solicitudes realizadas por el cliente Android.

4.2 Diseño

4.2.1 Arquitectura del sistema

Los clientes se comunican con la aplicación del módulo de información vía Web, mediante una arquitectura cliente/servidor. El servidor proporciona los servicios a los usuarios y la lógica de negocios. Estos a su vez realizan la validación correspondiente de la solicitud, para procesarla con el correspondiente servicio y permitir establecer conexión con el manejador de base de datos para realizar operaciones de inserción, consulta, edición y eliminado de información.

En la Figura 10, se muestra la arquitectura que tiene el módulo del sistema de información. En la capa de presentación se aprecia el cliente que realiza las peticiones HTTP al servidor,

para posteriormente poder recuperar o registrar información a la base de datos. Se crean servicios Web en el mismo servidor con la finalidad de darle la funcionalidad a la aplicación móvil para Android.

Figura 10. Arquitectura del sistema.

4.2.2 Diseño de la arquitectura de la solución

Para la implementación del sistema se elige el modelo en capas que es una técnica en programación que tiene por objetivo separar las diferentes partes de la aplicación, con el objetivo de mejorar su mantenimiento y sus funciones. La ventaja principal de este estilo es que el desarrollo se puede llevar a cabo en varios niveles y, en caso de que se desee realizar algún cambio, solo afectará al nivel requerido sin tener que revisar entre el código fuente de otros módulos. Además, debido a su diseño es altamente escalable ante la incorporación de nuevos módulos y funcionalidades a futuro.

Las tres capas más representativas son las siguientes:

- Capa de procesamiento de datos. Esta recibe la información enviada por la capa de aplicaciones, la cual verifica y ejecuta todas las funciones que pueden representar los datos en el sistema y su persistencia.

- Capa de almacenamiento de datos. Es donde residen los datos. Está formada por el gestor de base de datos MySQL que realiza todo el almacenamiento de los datos del sistema.

- Capa de aplicaciones. En esta se encuentran las aplicaciones que se usan en los dispositivos finales por los usuarios de cualquier rol, estas se conectan con la capa de procesamiento de datos puesto que esta entrega los datos en un objeto JSON para

luego ser presentados al usuario final y está conformada por la aplicación web principal de exámenes, en la cual los usuarios con roles asignados pueden administrar la información de resultados y usuarios desde un dispositivo con navegador web y por otra parte la aplicación móvil en la cual los usuarios podrán autorizar y registrar exámenes y ver la información de resultados y de usuarios.

4.2.3 Vista de Despliegue

A continuación, la figura 11 gráfica la representación de las relaciones entre los nodos físicos y su localización junto con los componentes en hardware y software.

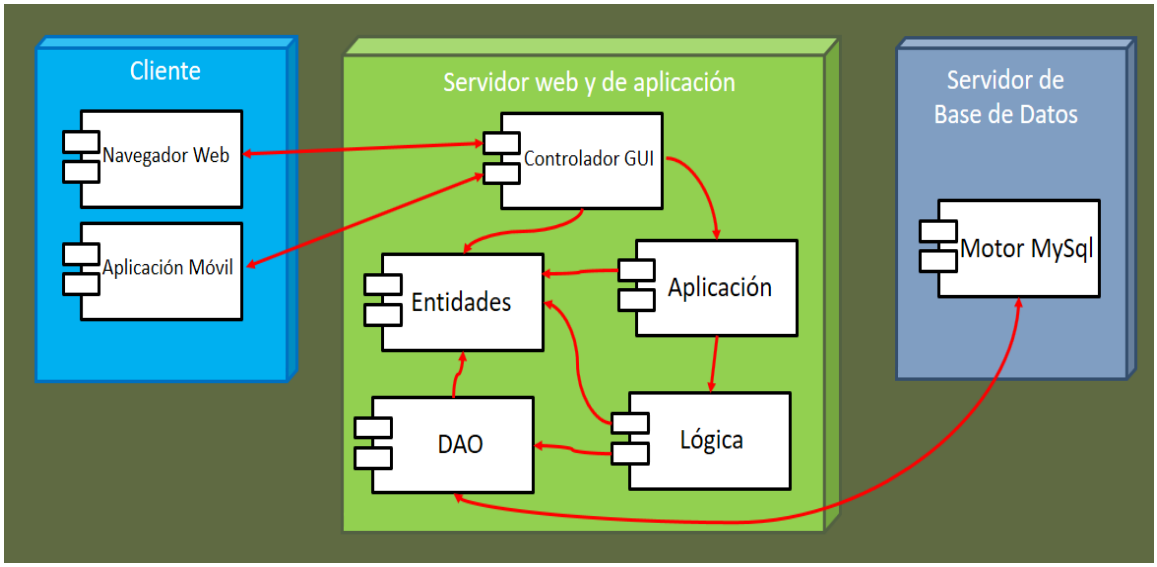


Figura 11. Diagrama de despliegue.

Los nodos indicados en la figura se describen a continuación.

- Cliente: Este nodo representa al navegador Web de la máquina cliente y la Aplicación móvil, desde el cual se realiza las peticiones al sistema.
- Servidor Web y de Aplicación: En este nodo residen los archivos del código fuente con la lógica de negocio estructurada en capas.
- Servidor de Base de datos: Este nodo contiene el sistema administrador de base de datos. Interactúa con el nodo del servidor Web en su capa de acceso a datos (DAO).

4.2.4 Diagramas de clases

Las clases de diseño representan a las entidades de negocio identificadas, con sus atributos y tipos de datos utilizados. Además, representan a las clases cuyos métodos más importantes tienen a cargo la implementación de la lógica de negocio, así como las operaciones de lectura y escritura en la base de datos.

A continuación, se muestra el diagrama de clases, que representan los objetos fundamentales del sistema, con los que el usuario va a interaccionar para obtener el resultado deseado. Reflejan la estructura estática del sistema.

4.2.4.1 Diagramas de clases Aplicación Web server

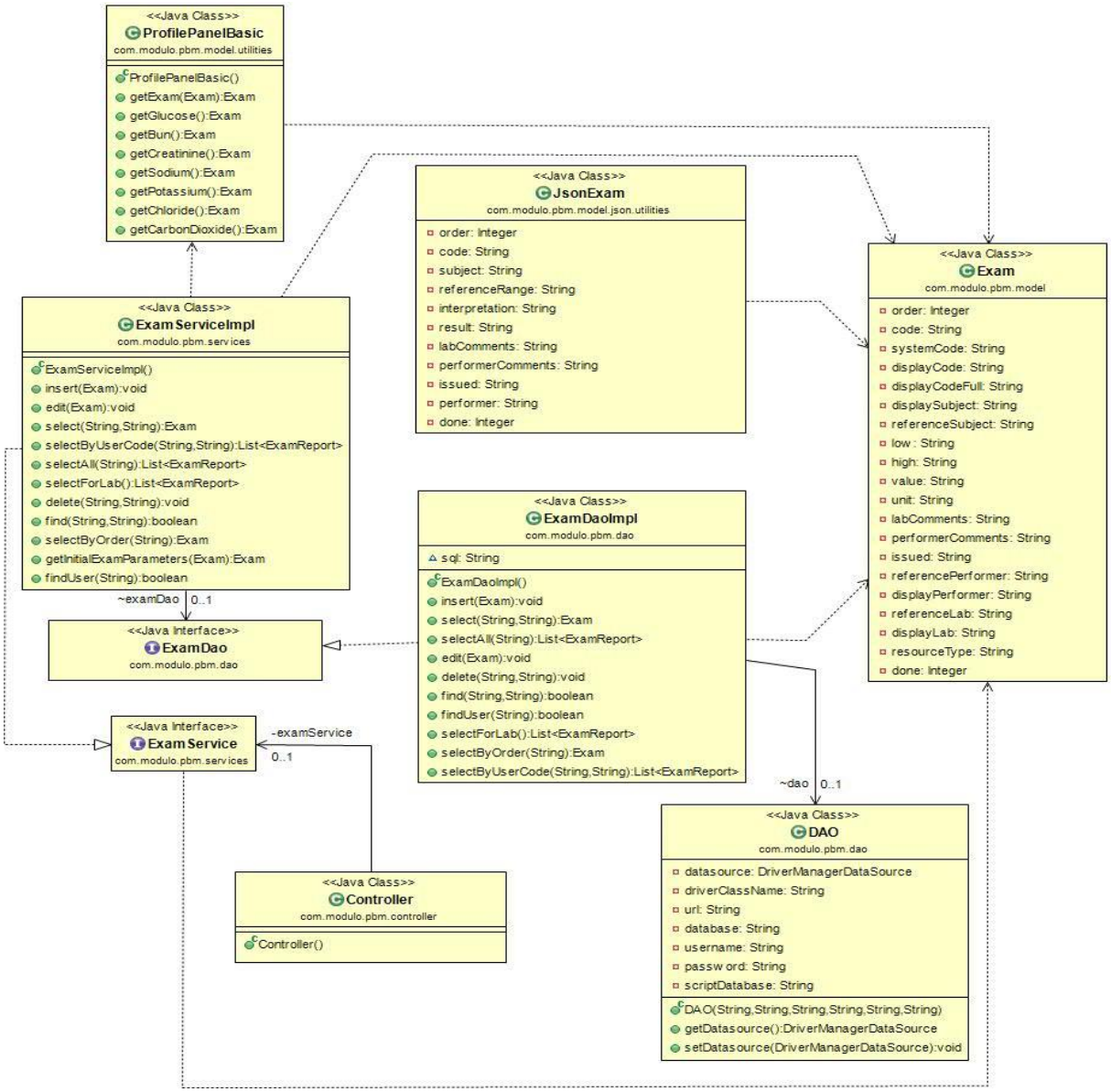


Figura 12. Diagrama de clases gestión de información de exámenes.

Las clases de diseño en la gestión de información de exámenes muestran la relación entre las clases de entidades, Dao y servicios. La interacción entre estas clases es imprescindible para las funcionalidades de autorización y consulta de resultados de exámenes.

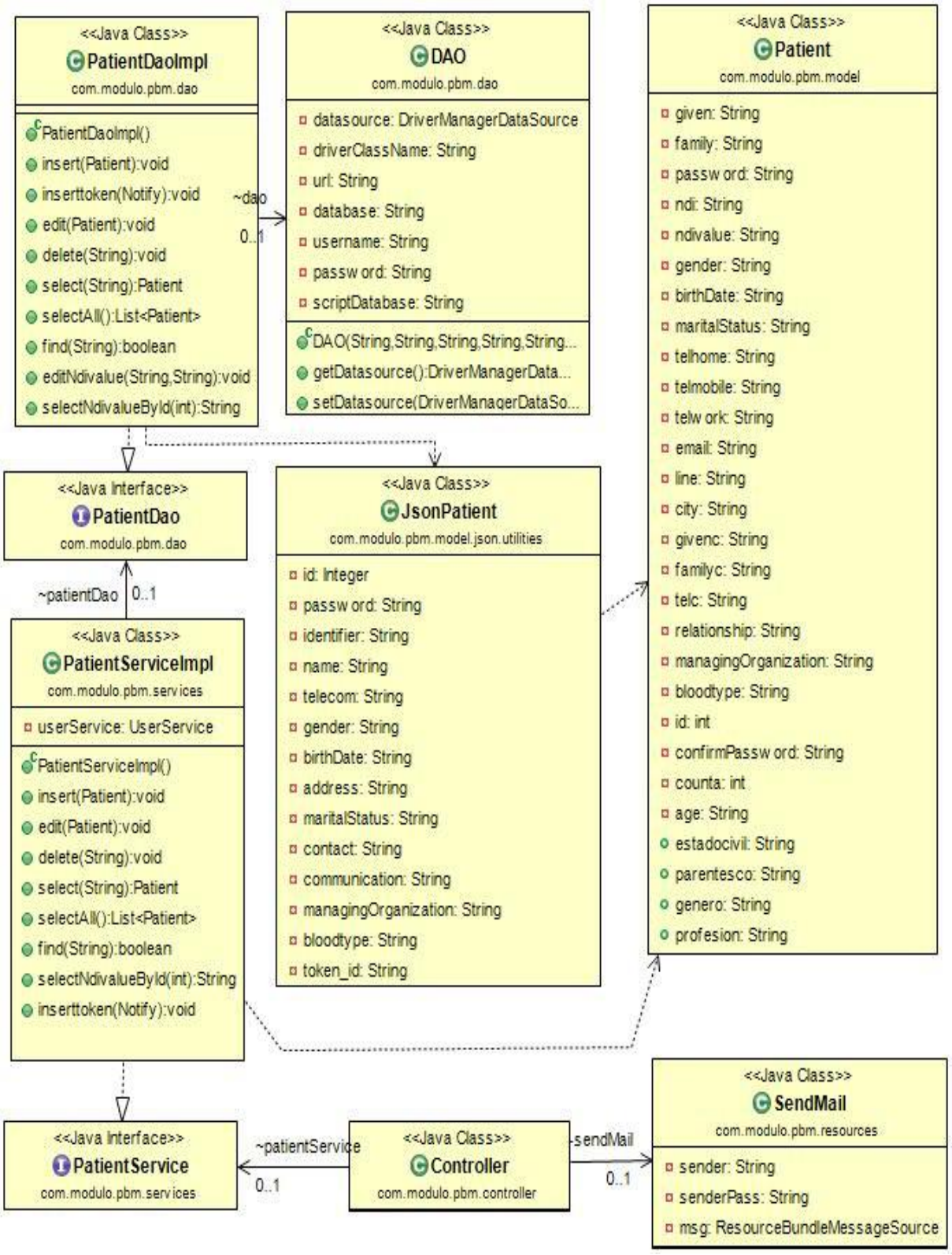


Figura 13. Diagrama de clases gestión de información de usuarios.

Las clases de diseño en la gestión de información de usuarios muestran la relación entre las clases de entidad, Dao y servicios. La interacción entre estas clases es imprescindible para las funcionalidades de registro, consulta, modificación y eliminación de registros de usuarios.

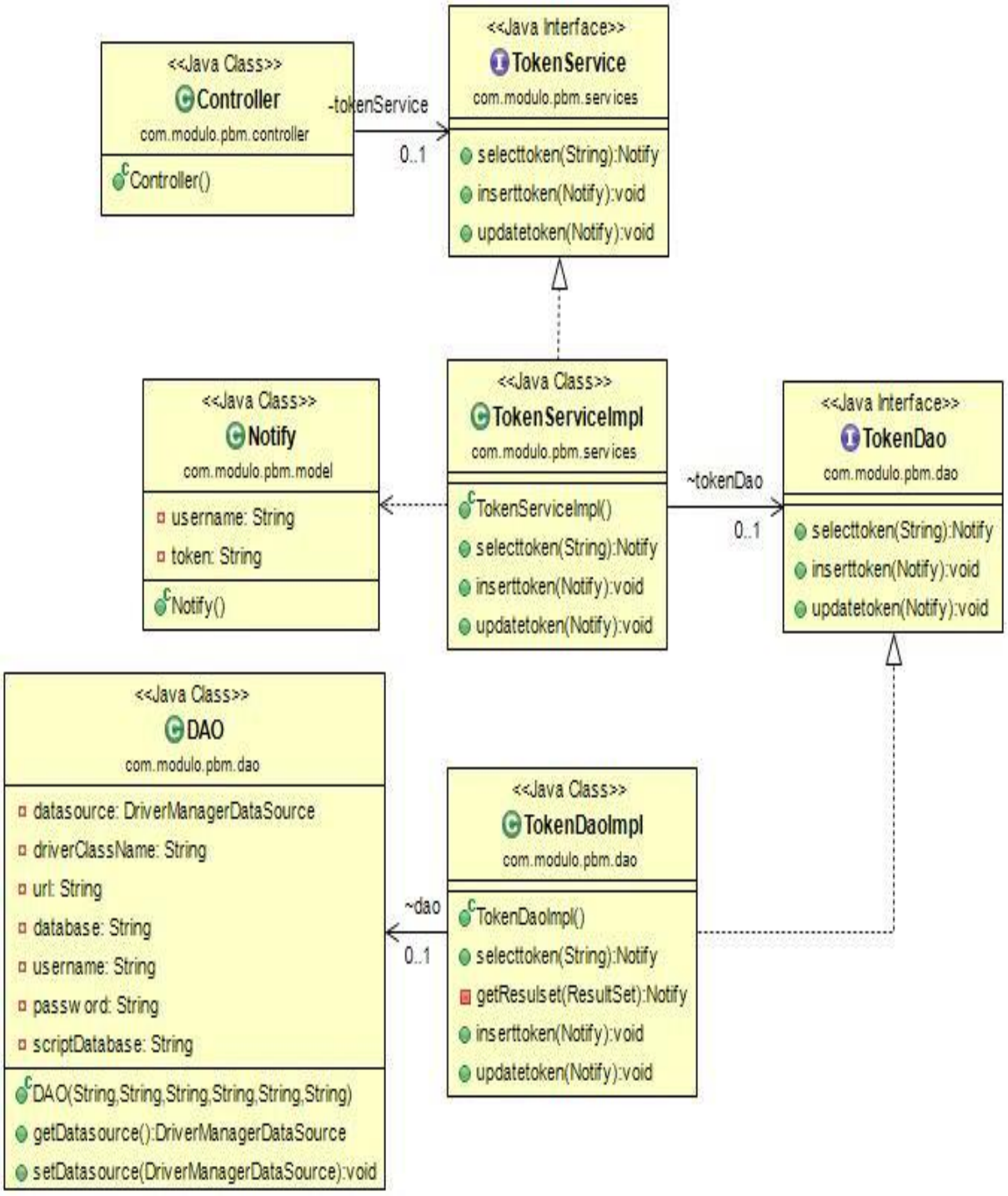


Figura 14. Diagrama de clases gestión Token.

Las clases de diseño en la gestión de Token muestran la relación entre las clases de entidad, Dao y servicios. La interacción entre estas clases es imprescindible para las funcionalidades de registro, consulta y modificación de Tokens.

4.2.4.2 Diagramas de clases Aplicación Android

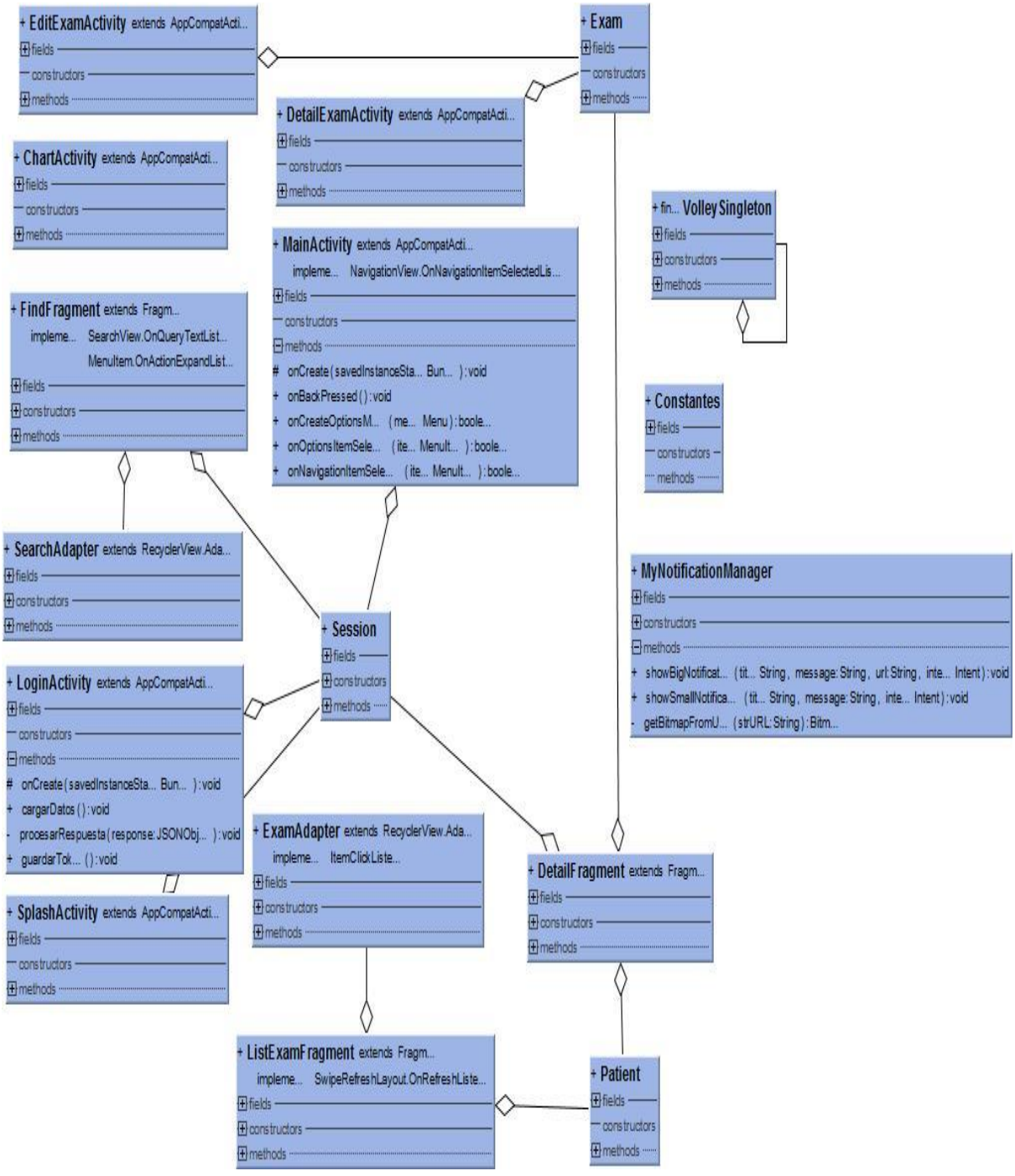


Figura 15. Diagrama de clases principales.

El diagrama que se muestra en la figura anterior está directamente relacionado con las clases que se utilizaron en la aplicación móvil, son creadas de acuerdo a la necesidad de nuestro proyecto. Se muestran las clases más importantes con sus respectivos atributos y como están relacionados unas con otras.

4.2.5 Diagramas de secuencia

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre estos para llevar a cabo la funcionalidad respectiva. El siguiente diagrama de secuencia muestra el flujo en orden de eventos que surgen entre la interacción del actor y el sistema principal.

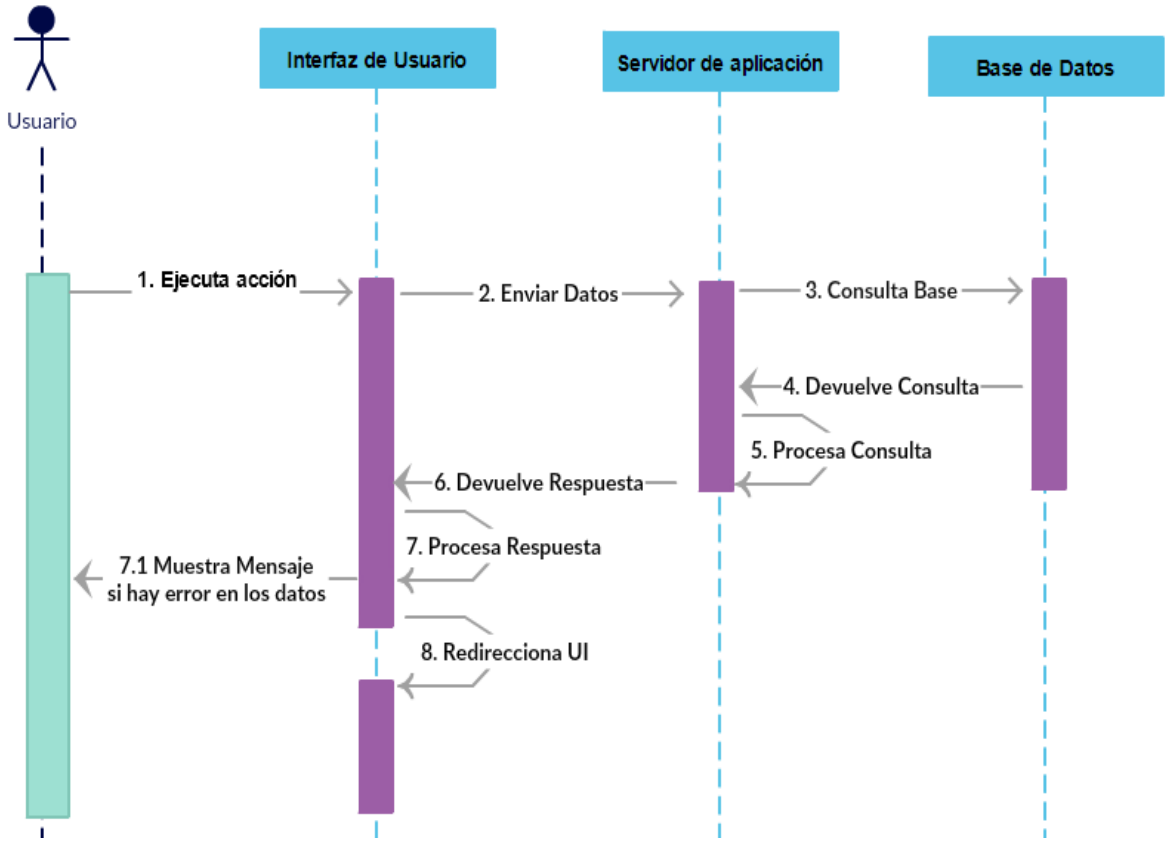


Figura 16. Diagrama de secuencia general del sistema.

Descripción general del diagrama de Secuencia de sistema general:

- Se presenta la interfaz principal al usuario.
- El usuario proporciona al sistema su usuario y contraseña y presiona el botón enviar.

- El sistema ejecuta los servicios correspondientes, para validar su existencia en la base de datos y el tipo de usuario.
- El sistema procesa la respuesta de la consulta y redirecciona a la sección correspondiente, según el tipo de usuario.

4.2.5.1 Diagramas de secuencia aplicación web

Luego de identificar que conceptos del modelo de dominio intervienen en los casos de uso, es necesario ahora indicar cuales son los mensajes que estas instancias deben intercambiar para completar la funcionalidad. Estos intercambios los presentamos en diagramas UML de secuencia.

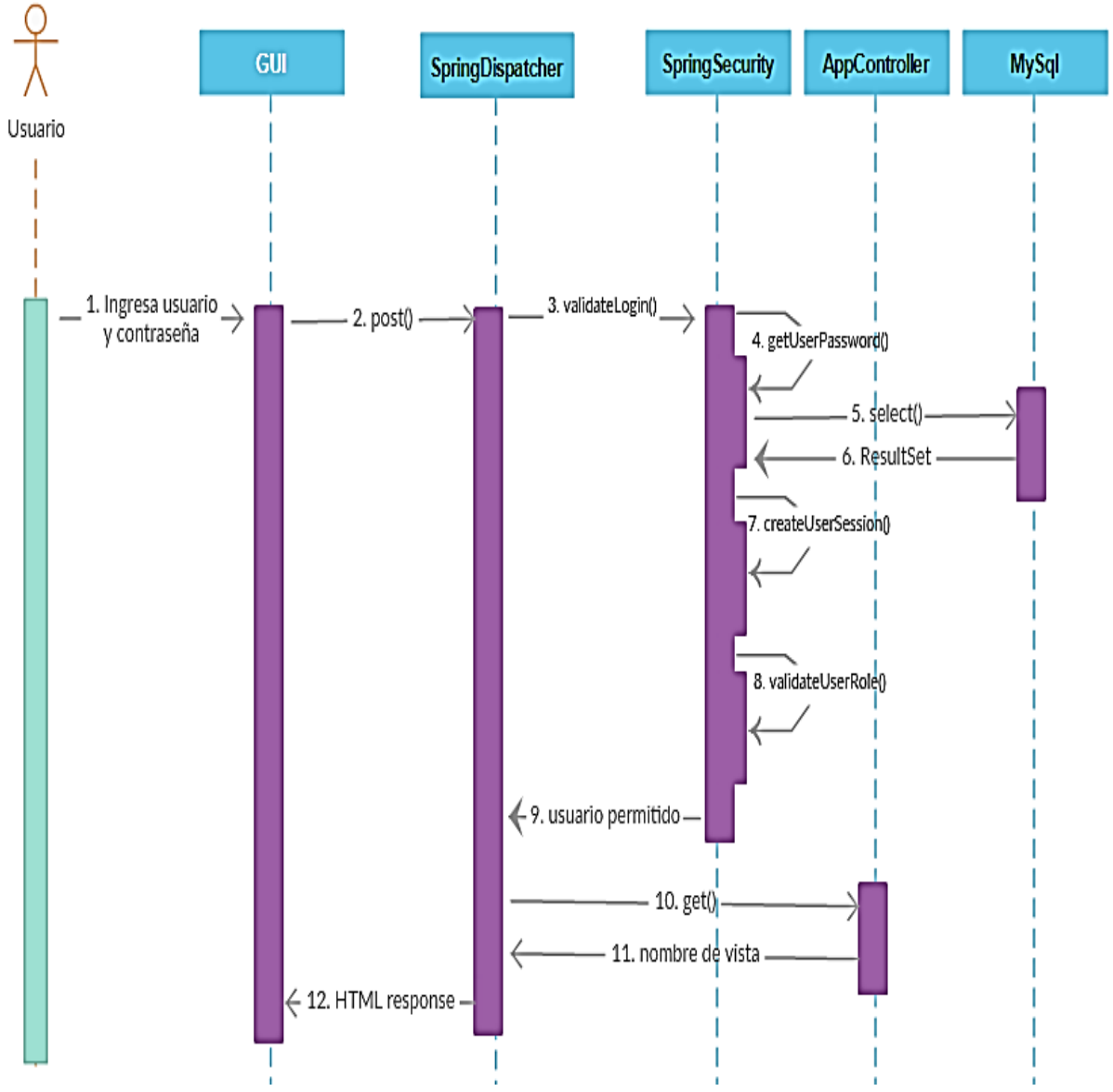


Figura 17. Diagrama de secuencia iniciar sesión.

Como se aprecia en la figura 17, los datos de usuario se envían en una petición post, si el usuario es correcto Spring Security guardará sus datos en el contexto de Spring como usuario que ha iniciado sesión y mediante el controlador retornara datos a las aplicaciones.

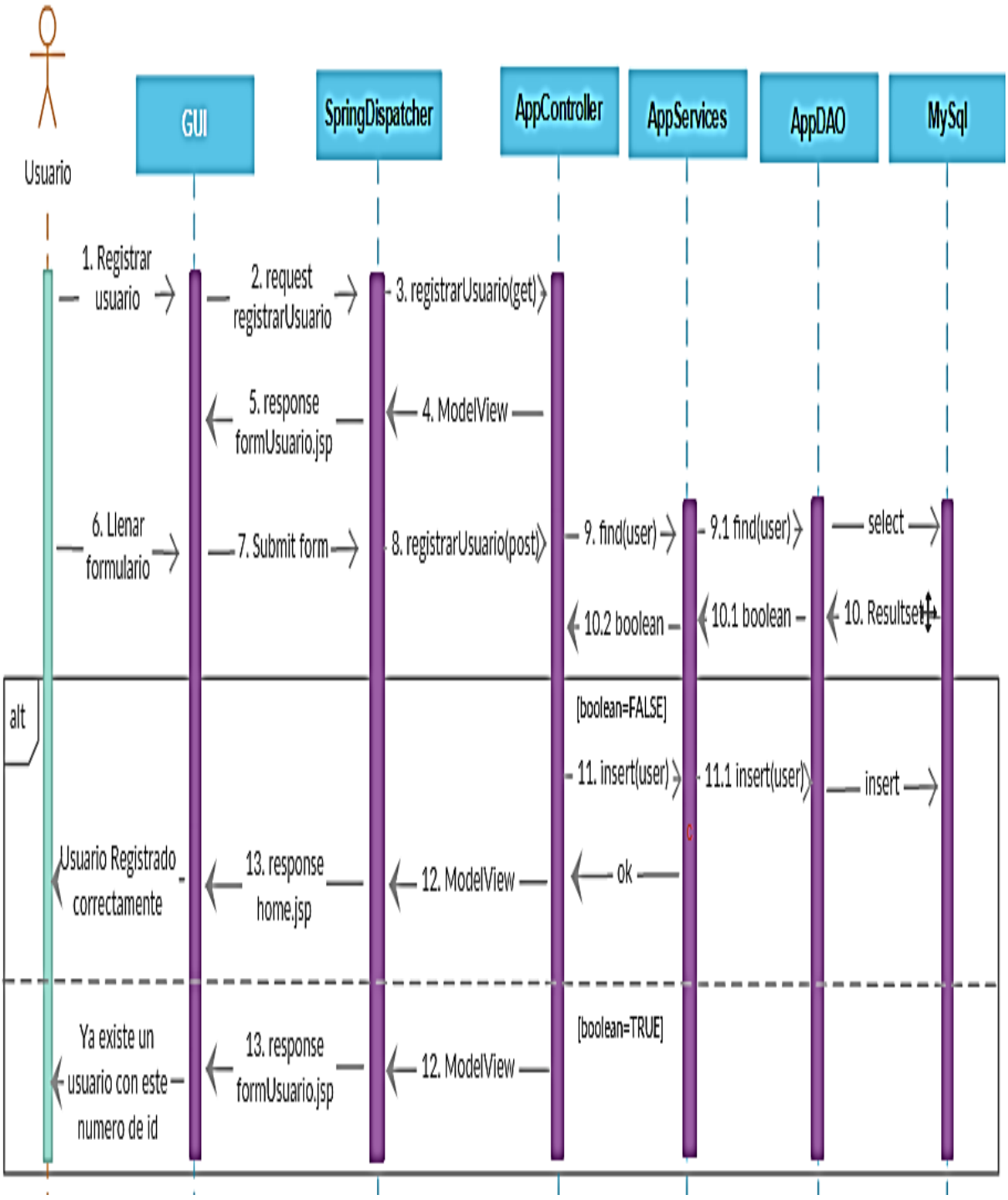


Figura 18. Diagrama de secuencia de registro de un usuario.

En la figura 18, se muestra el proceso de registro de usuarios desde la web del sistema. Primero el administrador solicita el formulario para completar los campos con la información

del usuario a registrar, una vez lleno los campos lo envía y el servidor hace uso de los servicios encargados para realizar un registro exitoso en la base de datos.

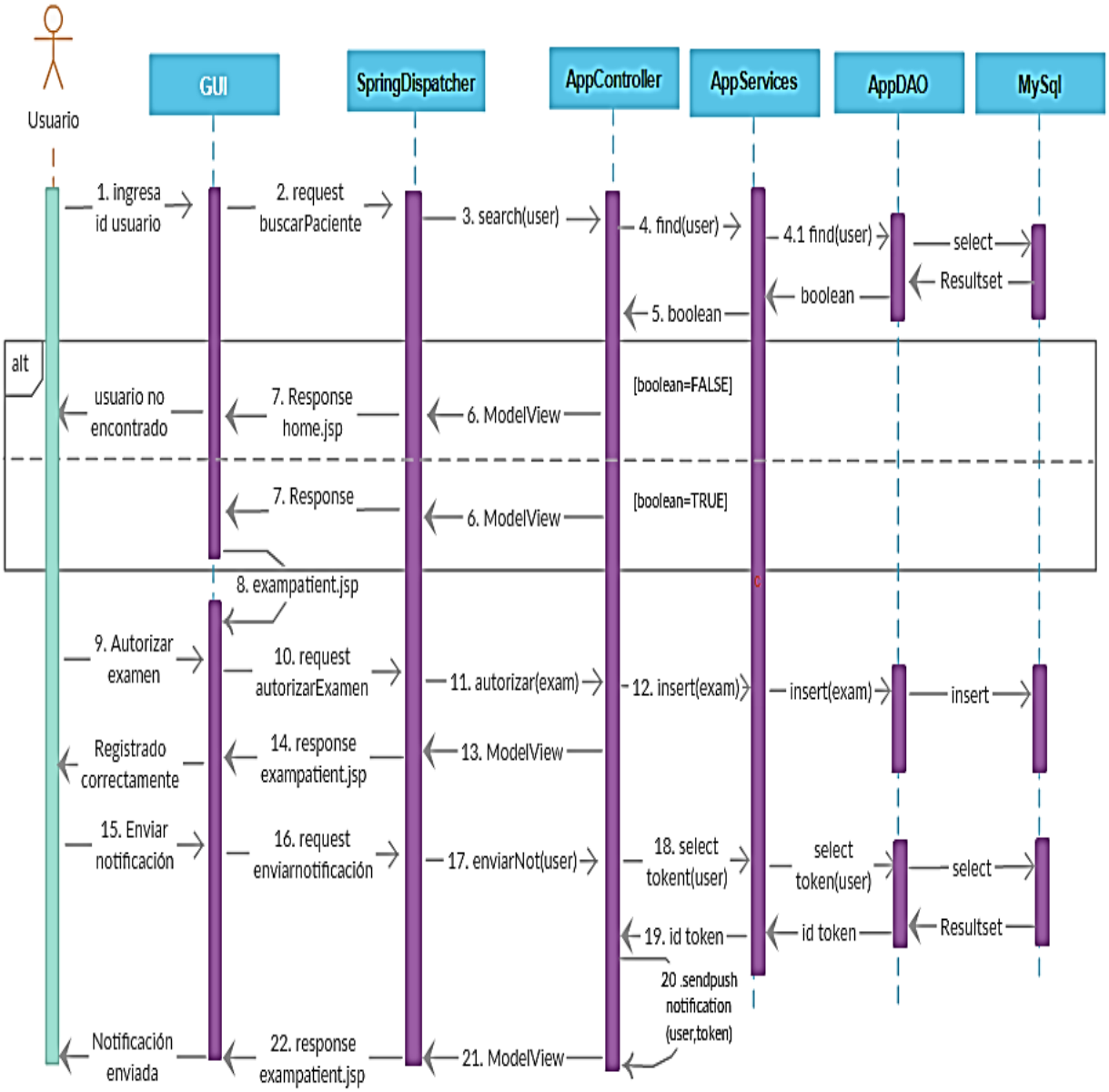


Figura 19. Diagrama de secuencia búsqueda, autorización y envió de notificación.

En la figura 19, se muestra el flujo de eventos en la búsqueda de pacientes para luego autorizar un examen y enviar notificaciones por parte del usuario tipo médico.

A continuación, en la figura 20, se muestra el flujo de eventos que ejecuta un usuario tipo laboratorista al ingresar al sistema, entre estos esta cargar la lista de exámenes autorizados, luego mostrando el formulario de examen para luego llenar los campos correspondientes con el resultado obtenido en el laboratorio y por último enviándolo para registrarlo en la base de datos.

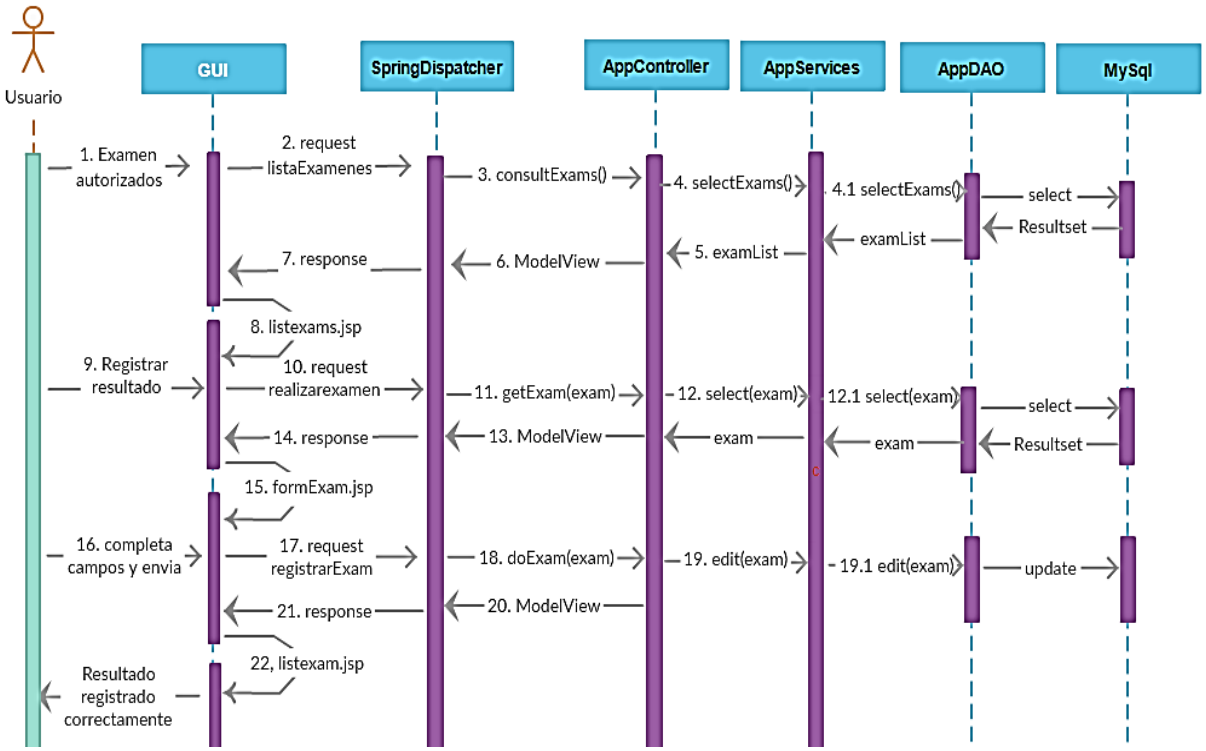


Figura 20. Diagrama de secuencia ver lista de exámenes autorizados y registrar resultado.

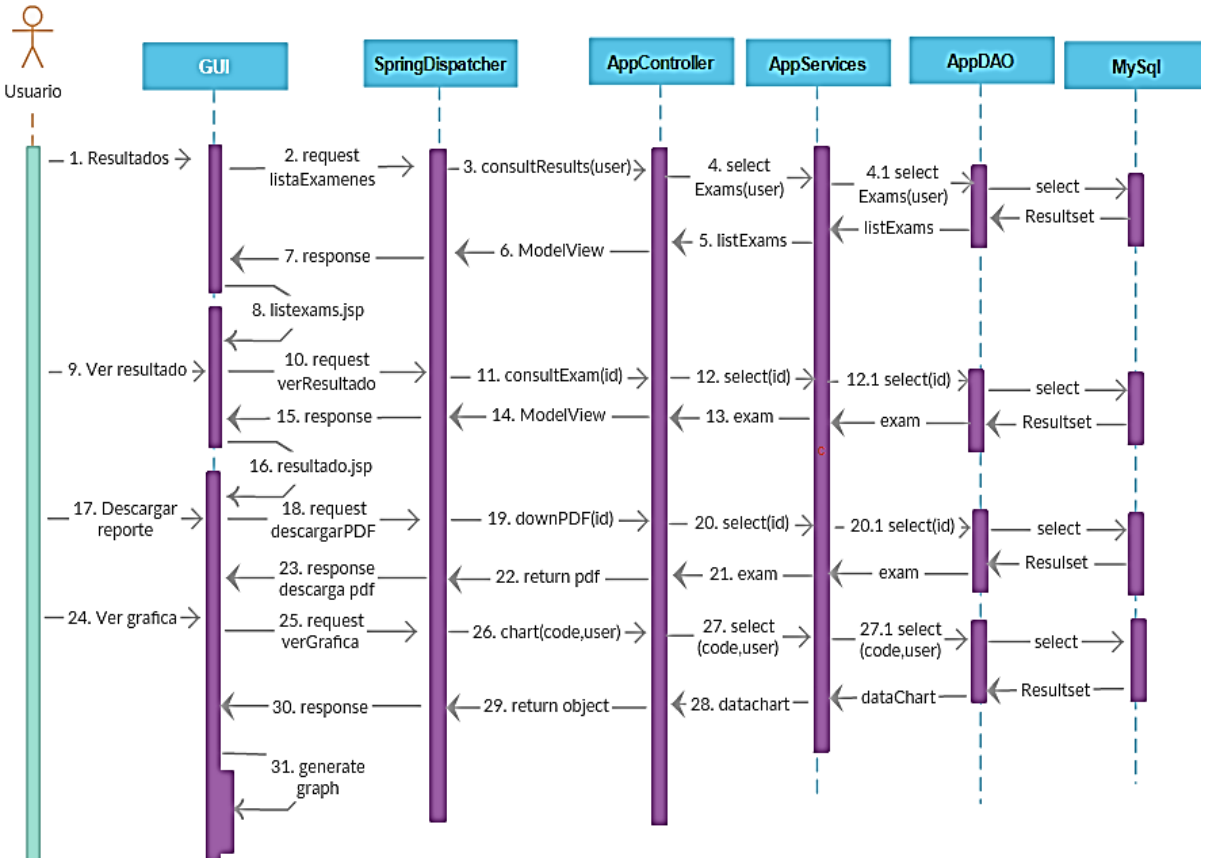


Figura 21. Diagrama de secuencia cargar lista de resultados, ver resultado, descargar PDF y ver gráfica.

En la figura anterior se muestra el flujo de eventos que ejecuta el usuario de tipo paciente al ingresar al sistema.

Como se pudo apreciar en los diagramas de secuencia el actor mediante la interface de usuario realiza las peticiones al servidor las cuales pasan por el DispatcherServlet el cual es el encargado de encontrar el controlador requerido, el controlador toma la solicitud y llama a los métodos de servicio apropiados en función del método GET o POST utilizado. Una vez el controlador realiza las operaciones retorna un ModelView donde va la información requerida y el nombre de la vista al DispatcherServlet, luego pasa los datos del modelo a la vista que finalmente se procesa en el navegador.

4.2.5.2 Diagramas de secuencia aplicación móvil

A continuación, se presenta los diagramas de secuencia para flujos de eventos del cliente Android. En la figura 22 se observa el flujo de eventos para el inicio de sesión.

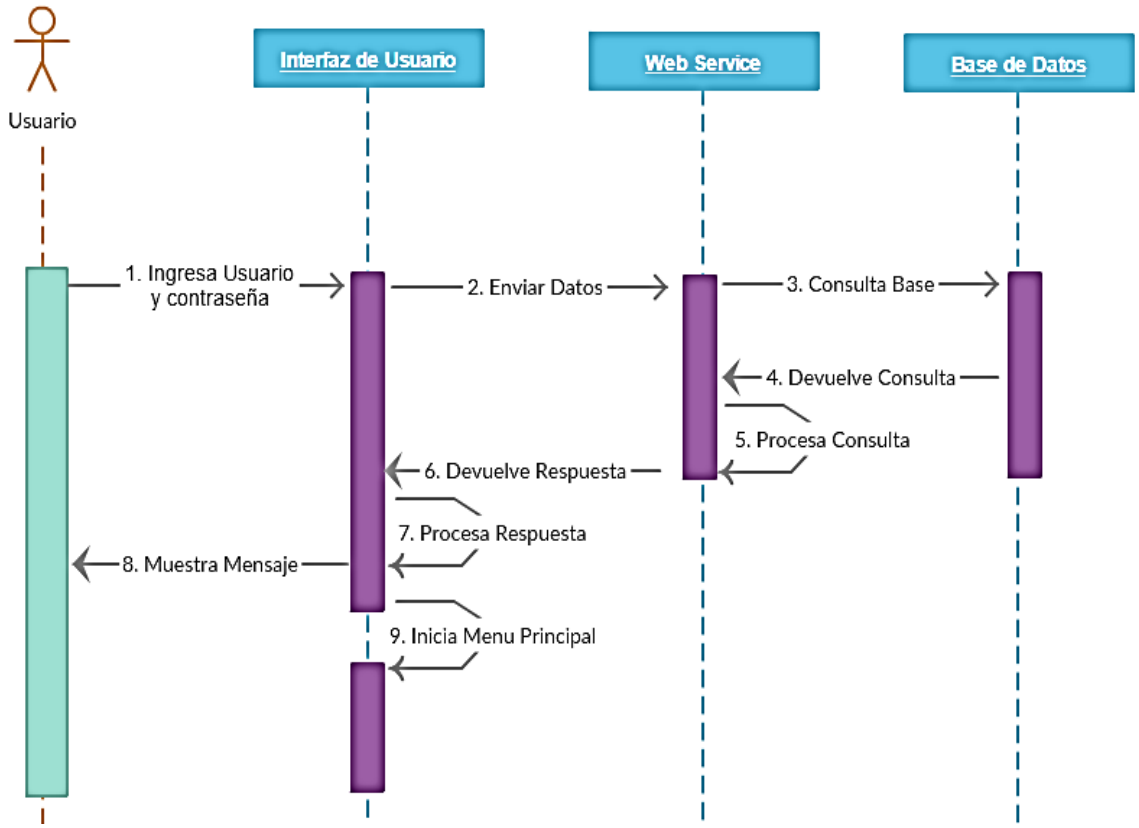


Figura 22. Diagrama de Secuencia Iniciar Sesión.

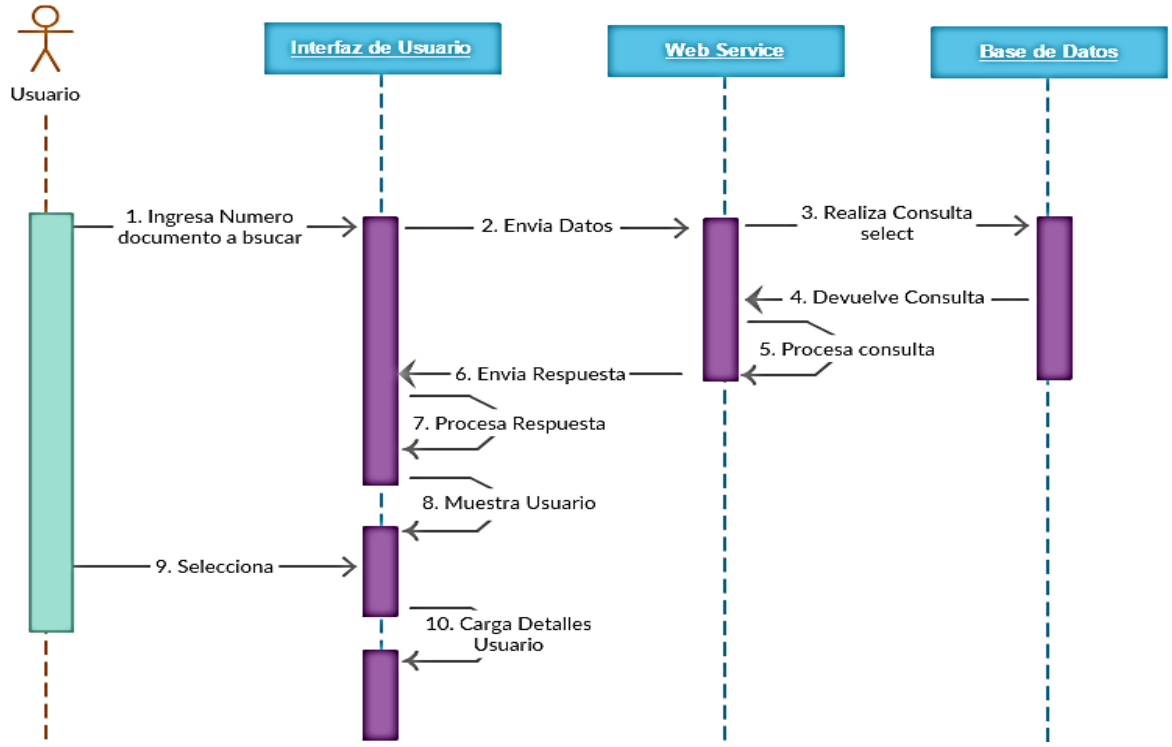


Figura 23. Diagrama de Secuencia Búsqueda de Paciente.

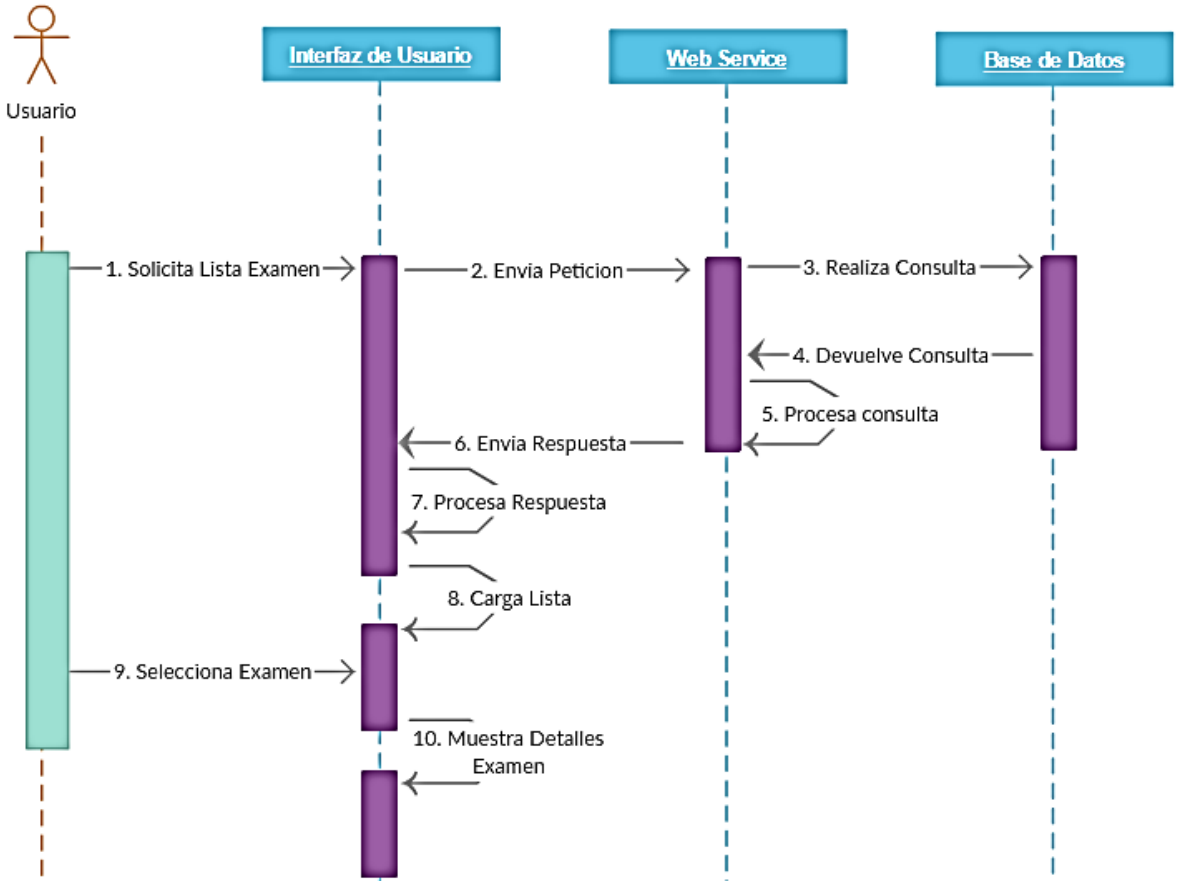


Figura 24. Diagrama de Secuencia Consulta de resultados Examen.

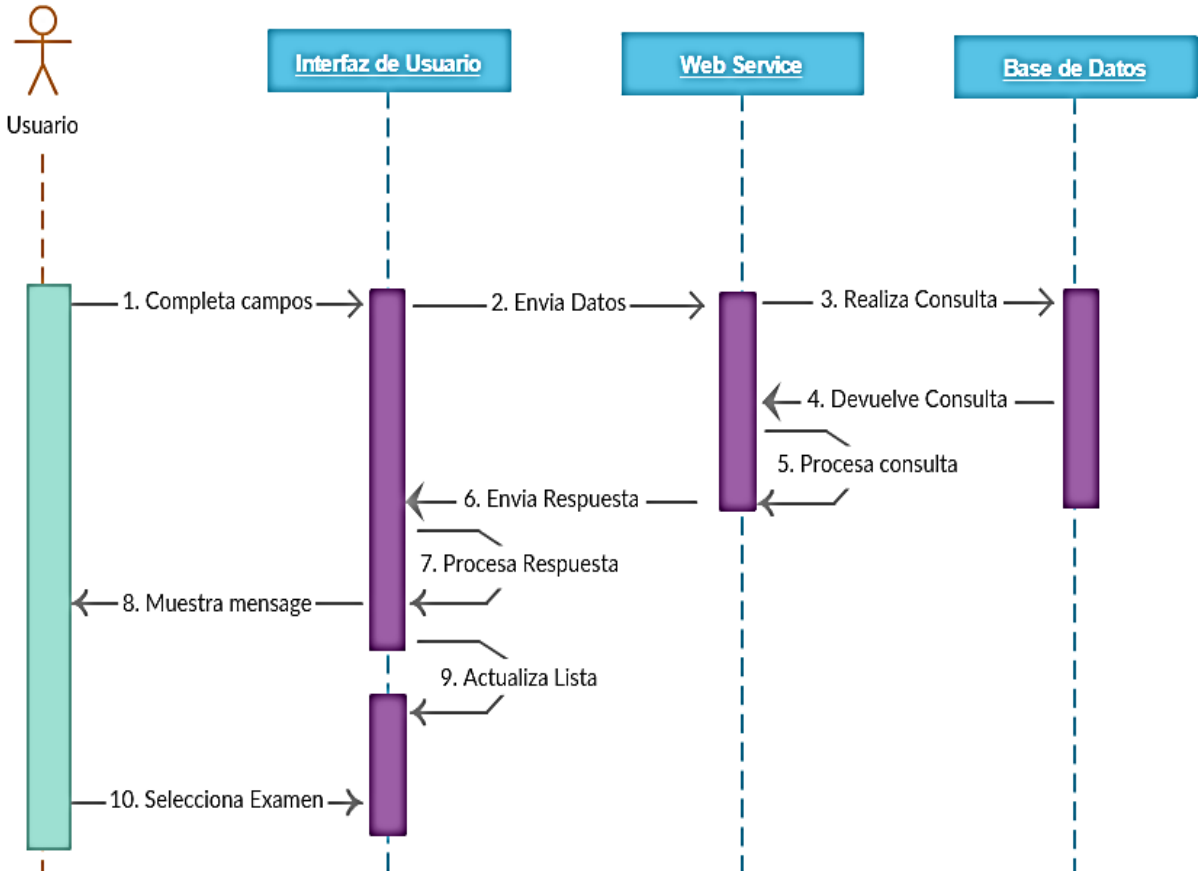


Figura 25. Diagrama de Secuencia Registro de resultados.

En los diagramas de secuencia para la aplicación móvil el actor mediante la interface de usuario realiza las peticiones al web service el cual se encarga de ejecutar los métodos de servicio apropiados. Una vez el controlador realiza las operaciones retorna un objeto JSON donde va la información requerida para finalmente procesarla en la aplicación móvil.

4.2.6 Diagrama de base de datos

Para el almacenamiento de información se ha utilizado una base de datos MySQL 5.7. Ésta alojara toda la información relativa a los usuarios y resultados del examen panel metabólico básico. Para el seguimiento y control de ésta se ha utilizado MySQL Workbench 6.3 CE.

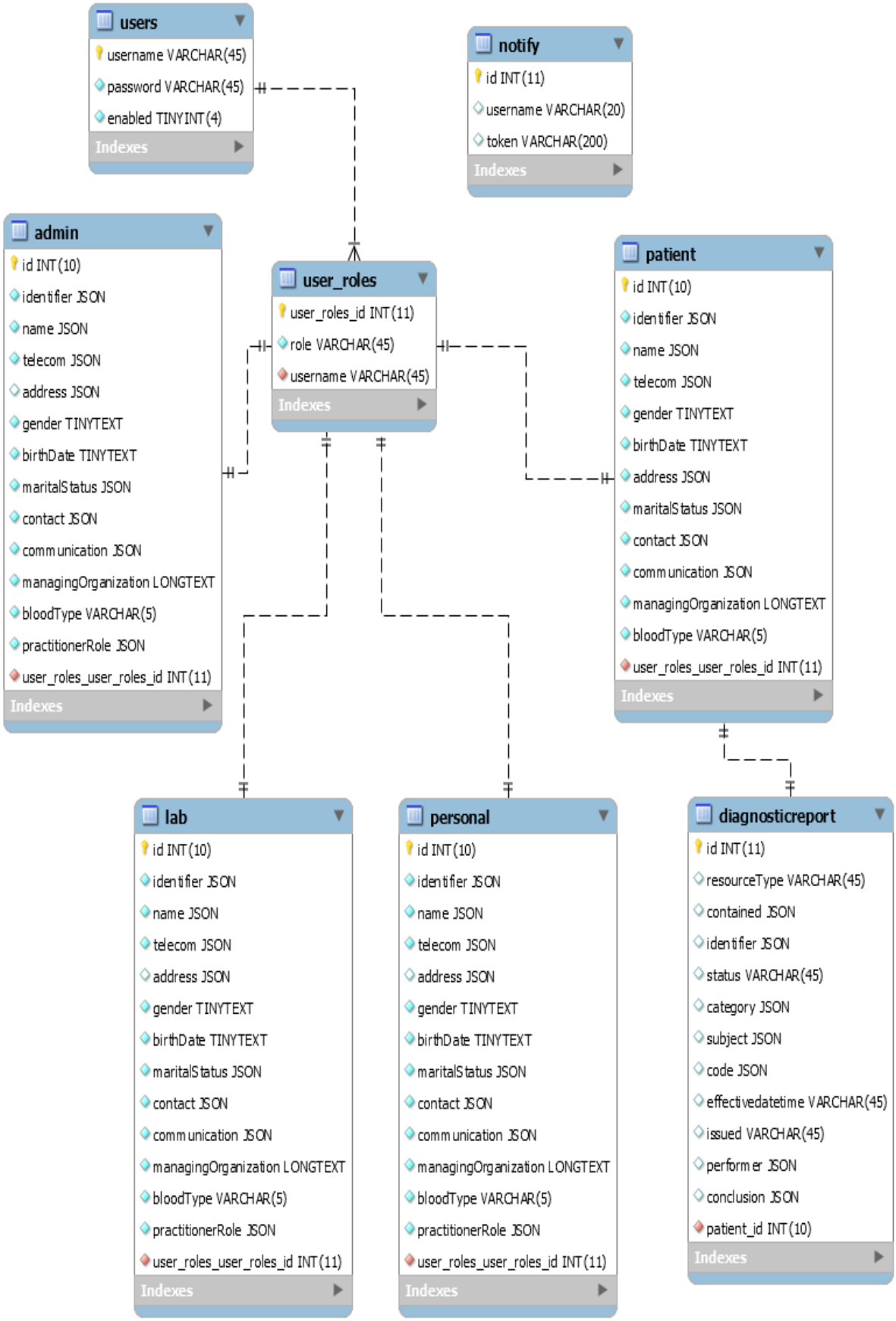


Figura 26 Diagrama de la base de datos del sistema.

4.3 Implementación

En esta sección se presenta cómo se ha llevado a cabo la implementación de la aplicación, que procesos se han seguido y cómo se ha llevado a cabo el proyecto. El desarrollo del proyecto se ha realizado con un computador portátil TOSHIBA, sistema operativo Windows 10 con un entorno de desarrollo Spring Tool Suite versión 3.8, Java 1.8, Android Studio 2.3 y servidor de pruebas Pivotal TC server versión 3.1.

4.3.1 Aplicación web server

El proyecto se ha desarrollado con Spring MVC 4. A continuación, se describe cómo se ha llevado a cabo la implementación de la aplicación web server.

4.3.1.1 Maven

Maven es una herramienta para crear y gestionar todo tipo de proyectos Java. Se utilizó para generar el nuevo proyecto. Una vez generado hay que configurar el fichero POM (project object model), un XML donde se deben añadir todas las dependencias del proyecto utilizando las versiones adecuadas. Una vez validado, Maven descargará todas las librerías JAR necesarias. Hecho esto, ya tenemos una base con la que empezar a trabajar.

4.3.1.2 Modelos

Para la implementación del modelo de datos del sistema se tiene en cuenta el estándar FHIR HL7, entonces lo primero es identificar qué información se va a ofrecer al usuario. Esta tarea consta de dos etapas. En primer lugar, es necesario estudiar el sistema, sus funcionalidades e interacción con el usuario, y extraer los objetos de información.

En segundo lugar, ya con los objetos de información identificados, es necesario estudiar la base de datos del sistema, así como su estructura, y modelar estos objetos, identificando los campos que han de tener y su relación con las tablas de la base de datos del sistema.

4.3.1.2.1 Identificación de los objetos de información del sistema

Como hemos introducido, es necesario hacer un estudio de la funcionalidad del sistema analizando cómo interactúan pacientes y profesionales con la aplicación. En primer lugar, el sistema establece los tipos de usuario Administrador, Personal Médico, Laboratorista y Paciente.

Los pacientes del sistema son personas que tendrán relación alguna con las variables de las pruebas del examen Panel Metabólico Básico.

Por su parte, los profesionales médicos, que forman parte de una organización sanitaria, también juegan un papel fundamental en el sistema. Se pretende que, a través del sistema, el profesional pueda conocer toda la información de sus pacientes.

Una vez hecho todo esto, podemos identificar una serie de objetos de información en el sistema:

- Administrador
- Paciente
- Laboratorista
- Personal Médico
- Examen
- Usuarios
- Roles de Usuarios
- Notificación

Un punto a tener en cuenta, es que estos objetos no solo tienen que existir, sino tener sentido fuera del contexto del sistema. Es por ello que se descartan algunos objetos para el modelado, pues la información que contiene solo tiene sentido dentro del sistema.

Una vez identificados los objetos de información del sistema, podemos pasar a identificar los recursos FHIR que utilizará nuestro servicio para enviar la información. Habrá que dar forma a la información contenida en estos objetos ya definidos, de acuerdo a lo que el estándar propone.

Primero se hace un estudio de los recursos que ofrece el estándar viendo cuáles son los apropiados para el sistema, para así relacionarlos con los atributos de nuestros objetos.

4.3.1.2.2 Identificación de los recursos FHIR

Para la identificación de los Recursos FHIR que se van a implementar, se realizó un estudio de la lista de recursos que ofrece la especificación.

Tabla 17. Recursos FHIR identificados

Recurso FHIR	Objeto del sistema	Objeto de información
Patient	Patient	Paciente
Practitioner	Personal	Personal sanitario
DiagnosticReport	DiagnosticReport	Reporte examen
Observation	Exam	Registro medida de variable

(Elaboración propia)

Recurso Patient

Este recurso contiene la información relativa a los pacientes que participan en alguna actividad relacionada con la salud, sus atributos se centran en la información necesaria para apoyar los procedimientos clínicos que tienen lugar en torno al paciente.

Para este proyecto, contiene la información relativa al Paciente.

Tabla 18. Recurso Patient

Atributo	Tipo de dato	Cardinalidad	Descripción
identifier	Identifier	0..*	Identificador del paciente
name	HumanName	0..*	Nombres y apellidos del paciente
telecom	ContactPoint	0..*	Teléfono/correo electrónico del paciente
gender	Code	0..1	Sexo del paciente
birthDate	date	0..1	Fecha de nacimiento del paciente
address	Address	0..*	Dirección de residencia del paciente
maritalStatus	CodeableConcept	0..1	Estado civil del paciente
contact	BackBoneElement	0..*	Información de contacto personal

comunication	BackBoneElement	0..*	Idiomas que se pueden usar para comunicarse con el paciente sobre su salud
bloodType	Code	0..1	Tipo de sangre

(https://www.hl7.org/fhir/)

Recurso Practitioner

Este recurso incluye a todas las personas que están involucradas en el proceso de cuidado de la salud de un paciente, como parte de sus actividades y responsabilidades formales. Entre otros roles, contempla ser utilizado para el personal médico, personal de enfermería y laboratoristas.

Para este proyecto, contendrá la información de Administrador, Laboratorista y Personal Médico.

Tabla 19. Recurso Practitioner

Atributo	Tipo de dato	Cardinalidad	Descripción
identifier	Identifier	0..*	Identificador del profesional
name	HumanName	0..*	Nombres y apellidos del profesional
telecom	ContactPoint	0..*	Teléfono/correo electrónico del profesional
gender	Code	0..1	Sexo del profesional
birthDate	date	0..1	Fecha de nacimiento del profesional
address	Address	0..*	Dirección de residencia del profesional
maritalStatus	CodeableConcept	0..1	Estado civil del profesional
contact	BackBoneElement	0..*	Información de contacto personal

comunication	BackBoneElement	0..*	Idiomas que se pueden usar para comunicarse con el profesional
bloodType	Code	0..1	Tipo de sangre
practitionerRole	BackBoneElement	0..*	Cargo o profesión del personal sanitario

(https://www.hl7.org/fhir/)

Recurso Observation

Dentro de este recurso se engloban todos los resultados obtenidos de las actividades realizadas con o sobre el paciente. Entre ellos, encontramos medidas de signos vitales, datos de laboratorio o imágenes médicas.

En nuestro caso, contiene la información de los objetos de examen para registro de medida de variable.

Tabla 20. Recurso Observation

Atributo	Tipo de dato	Cardinalidad	Descripción
code	CodeableConcept	0..*	Tipo de observación
subject	Reference	0..*	Nombres del paciente
referenceRange	BackBoneElement	0..*	Proporciona datos para la interpretación
interpretation	CodeableConcept	0..1	Alto, bajo, normal etc.
valueQuantity	Quantity	0..1	Valor del resultado
labComments	string	0..*	Comentarios sobre el resultado
performerComments	string	0..1	Comentarios sobre el resultado
issued	date	0..*	Fecha de emisión
performer	Reference	0..*	Nombre profesional sanitario encargados del resultado

done	Code	0..1	Estado

(https://www.hl7.org/fhir/)

4.3.1.2.3 Modelado de los objetos de información del sistema

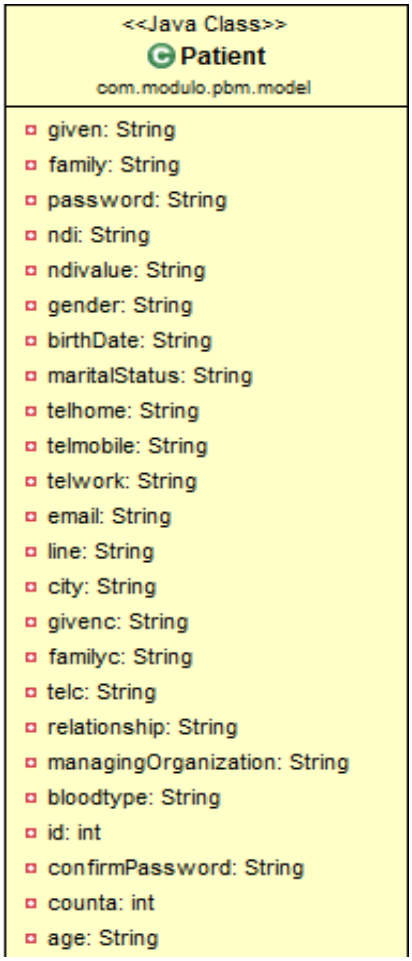
Una vez identificados los objetos hay que modelarlos, indicando los atributos que lo forman e identificándolos con campos de la base de datos de donde recogeremos la información. En estos objetos es donde se guardará la información obtenida de las consultas a la base de datos para posteriormente, modelarla en los recursos FHIR. Por ello, como métodos únicamente tendrán Getters y Setters, para obtener o modificar, respectivamente, estos atributos con la información.

Objeto Paciente

En la Tabla se muestra el objeto que contiene la información referente a un usuario de tipo paciente. Se identifican y describen sus atributos y dónde está almacenado cada elemento de información en la base de datos. También se añade la representación del objeto modelado.

Tabla 21. Objeto Paciente

Atributos	Relación BD	Descripción	Modelo
given	name	Nombre paciente	
family	name	Apellido paciente	
ndi	identifier	Tipo documento	
ndivalue	identifier	Número de documento	
gender	gender	Género paciente	
birthDate	birthDate	Fecha de Nacimiento	
maritalStatus	maritalStatus	Estado Civil	
telhome	telecom	Número teléfono casa	
telmobile	telecom	Número teléfono celular	

telwork	telecom	Número teléfono oficina	
email	telecom	Correo electrónico	
line	address	Dirección residencia	
city	address	Ciudad	
givenc	contact	Nombre contacto	
familyc	contact	Apellido contacto	
telc	contact	Nro. teléfono contacto	
relationShip	contact	Parentesco	
Managing Organization	Managing Organization	Entidad prestadora de salud	
bloodType	bloodType	Tipo de sangre	

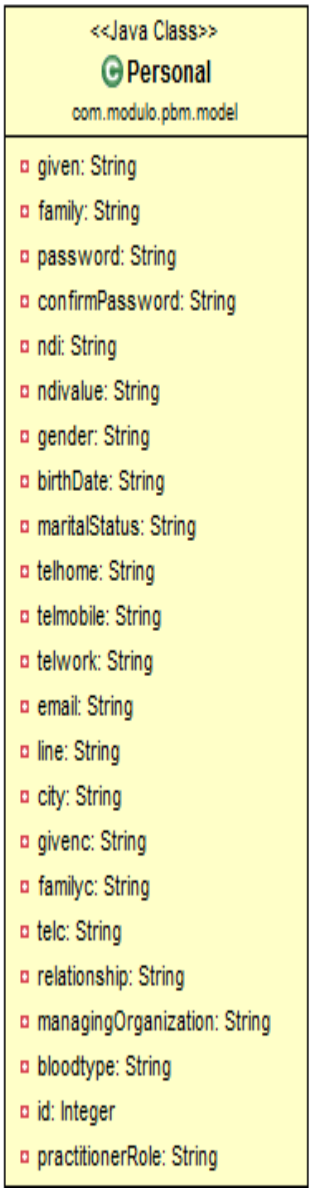
(Elaboración propia)

Objeto Personal Médico, Laboratorista y Administrador

Objeto que contiene la información almacenada en base de datos sobre un usuario del tipo Personal Médico, Laboratorista y Administrador. Se emplea una clase para cada tipo de usuario y se hace el análisis a una ya que la información a extraer es la misma.

Tabla 22. Objeto Personal Sanitario

Atributos	Relación BD	Descripción	Modelo
given	name	Nombre profesional	
family	name	Apellido profesional	
ndi	identifier	Tipo documento	

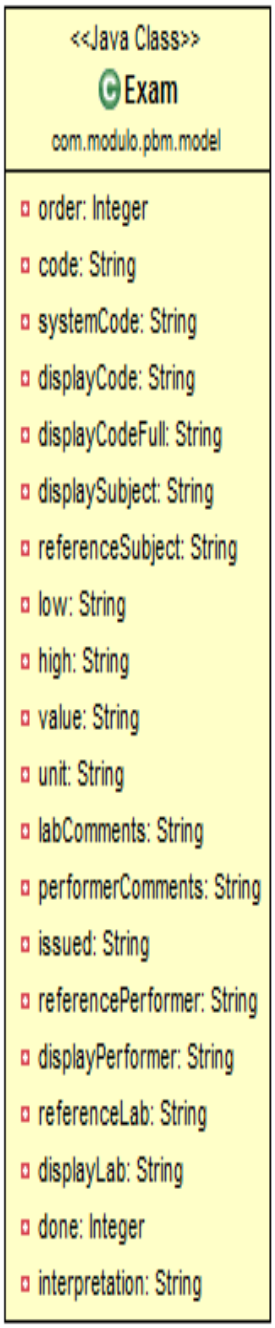
ndivalue	identifier	Número de documento	
gender	gender	Género profesional	
birthDate	birthDate	Fecha de Nacimiento	
maritalStatus	maritalStatus	Estado Civil	
telhome	telecom	Número teléfono casa	
telmobile	telecom	Número teléfono celular	
telwork	telecom	Número teléfono oficina	
email	telecom	Correo electrónico	
line	address	Dirección residencia	
city	address	Ciudad	
givenc	contact	Nombre contacto	
familyc	contact	Apellido contacto	
telc	contact	Número teléfono contacto	
relationShip	contact	Parentesco	
Managing Organization	Managing Organization	Entidad prestadora salud	
practitionerRole	practitionerRole	Profesión	
bloodType	bloodType	Tipo de sangre	

(Elaboración propia)

Objeto Examen

Objeto que contiene la información de un resultado del examen Panel Metabólico Básico.

Tabla 23. Objeto Examen

Atributos	Relación BD	Descripción	Modelo
order	id	Id en tabla	
code	code	Código examen	
systemCode	code	Referencia de recursos del examen	
displayCode	code	Representación definida por el sistema	
displaySubject	subject	Nombre paciente	
referenceSubject	subject	Referencia paciente	
low	referenceRange	Limite bajo	
high	referenceRange	Limite alto	
value	valueQuantity	Resultado	
unit	valueQuantity	Unidades de medida	
labComents	labComments	Observaciones del laboratorista	
Performer Comments	Performer Comments	Observaciones del médico	
issued	issued	fecha	
referencePerformer	performer	Referencia Médico encargado	
displayPerformer	performer	Nombre Médico encargado	
referenceLab	performer	Referencia laboratorista encargado	

displayLab	performer	Nombre laboratorista encargado	
done	done	Estado	
interpretation	interpretation	Interpretación de resultado	

(Elaboración propia)

4.3.1.2.4 Implementación de los recursos FHIR

Una vez identificados y modelados los recursos que se utilizaran, hay que definir el formato de los recursos.

Para establecer el formato definido por el estándar FHIR, se seleccionó JSON debido a que es un formato para el intercambio de datos y de fácil uso en java. Una de las mayores ventajas que tiene el uso de JSON es que puede ser leído por cualquier lenguaje de programación. Por lo tanto, puede ser usado para el intercambio de información entre distintas tecnologías.

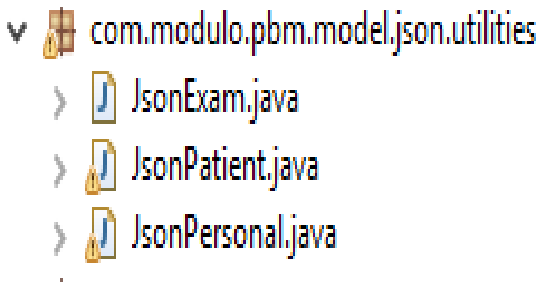


Figura 27. Paquete que contiene las clases que generan los objetos JSON.

Una vez identificados los recursos del objeto se procede a codificar los objetos JSON de cada recurso.

Para codificar la información en formato JSON se crea la clase JsonPatient.java la cual se encargará de crear los objetos JSON que representan los elementos y sus atributos de los recursos patient.

A continuación, se detallará las implementaciones de algunos de los elementos para entender cómo funciona.

```
public void setIdentifier(String value, String display) {

    JsonNodeFactory factory = JsonNodeFactory.instance;
    ObjectNode identifier = factory.objectNode();
    ArrayNode arrayNode = factory.arrayNode();

    ObjectNode objectNode = factory.objectNode();
    objectNode.put("use", "official");
    objectNode.put("value", value);
    objectNode.put("system", "urn:oid:2.16.840.1.113883.2.17.6.3");
    objectNode.put("display", display);

    arrayNode.add(objectNode);

    identifier.set("identifier", arrayNode);
    this.identifier = identifier.toString();
}
```

Figura 28. Codificación JSON elemento identifier del recurso Patient.

De la figura 28, la clase JsonNodeFactory especifica métodos para obtener acceso a instancias de nodo, así como a la implementación básica de los métodos. El ArrayNode es una clase de nodo que representa matrices mapeadas del contenido de Json. ObjecNode, nodo que se asigna a estructuras de objetos JSON en contenido JSON. A continuación, en la figura 29 se muestra la salida JSON de la implementación del elemento identifier y debajo de esta sus nodos.

{"identifier": [{"use": "official", "value": "6", "system": "urn:oid:2.16.840.1.113883.2.17.6.3", "display": "CC"}]}

```
object {
  "identifier": array [
    object {
      "use": string "official",
      "value": string "6",
      "system": string "urn:oid:2.16.840.1.113883.2.17.6.3",
      "display": string "CC"
    }
  ]
}
```

Figura 29. Salida JSON elemento identifier y sus nodos.

```
public void setCode(String code, String system, String display) {
    JsonNodeFactory factory = JsonNodeFactory.instance;

    ArrayNode arrayCode = factory.arrayNode();
    arrayCode.add(new ObjectNode(new JsonNodeFactory(false))
            .put("code", code)
            .put("system", system)
            .put("display", display));

    ObjectNode coding = factory.objectNode();
    coding.set("coding", arrayCode);

    ObjectNode code1 = factory.objectNode();
    code1.set("code", coding);
    this.code = code1.toString();
}
```

Figura 30. Codificación JSON del elemento code del recurso Exam.

```
public void setPractitionerRole(String managingOrganization, String code) {

    String display = "";
    switch (code) {
    case "doctor":
        display = "Doctor";
        break;
    case "nurse":
        display = "Nurse";
        break;
    case "pharmacist":
        display = "Pharmacist";
        break;
    case "researcher":
        display = "Researcher";
        break;
    case "teacher":
        display = "Teacher/educator";
        break;
    case "ict":
        display = "ICT professional";
        break;
    }
    JsonNodeFactory factory = JsonNodeFactory.instance;

    ArrayNode arrayCode = factory.arrayNode();
    arrayCode.add(new ObjectNode(new JsonNodeFactory(false)).put("code", code)
            .put("system", "http://snomed.info/sct").put("display", display));

    ObjectNode role = factory.objectNode();
    role.set("coding", arrayCode);

    ObjectNode communication1 = factory.objectNode();
    communication1.set("role", role);
    communication1.put("managingOrganization", managingOrganization);
    ObjectNode practitionerRole = factory.objectNode();
    practitionerRole.set("practitionerRole", new ArrayNode(new JsonNodeFactory(false)).add(communication1));
    this.practitionerRole = practitionerRole.toString();
}
```

Figura 31. Codificación JSON del elemento practitionerRole del recurso Personal.

Tras analizar los recursos FHIR, se observa que principalmente se utilizan dos tipos de datos codificados, uno para códigos propios de FHIR y otro para códigos con terminologías externas:

- **Códigos propios de FHIR**, se trata de códigos cerrados propios de este estándar. Comprenden diferentes vocablos que describen una variable concreta dentro de un recurso determinado. Existen tantas codificaciones como variables que se codifican. Por ejemplo, dentro del elemento identifier, tenemos la variable use, que puede adoptar como valor usual, official, temp o secondary.

- **Códigos con terminologías externas**, estos presentan equivalencias con otros códigos y terminologías externas a HL7, para el proyecto se hace uso de LOINC. Este tipo permite que las variables ligadas a estos códigos presenten equivalencias directas entre conceptos con otros estándares. Gracias a esto se obtiene mayor grado de interoperabilidad semántica.

Variables que describen los elementos de un recurso:

- **Use**, propósito del recurso

- **System**, Establece el espacio de nombre para el valor, es decir, una URL que describe un conjunto de valores que son únicos.

- **Type**, Un tipo codificado para el identificador que se puede usar para determinar qué identificador usar para un propósito específico.

- **Value**, La parte del identificador típicamente relevante para el usuario y que es única dentro del contexto del sistema.

- **Code**, Describe lo que se observó, en ocasiones esto se llama "nombre" de la observación.

- **Coding**, Código definido por un sistema de terminología

- **Display**, Una representación del significado del código en el sistema, siguiendo las reglas del sistema.

4.3.1.3 Servicios

Contiene componentes que se encargan de realizar acciones (Agregar, Consultar, Actualizar y Eliminar) solicitadas por los controladores, procesando la información necesaria de los componentes.

Por ejemplo, si el Laboratorista va a registrar un resultado, el servicio que se ejecutará será el de edit(), en este caso que se encuentra en la clase ExamService.java el cual contiene toda la lógica necesaria para invocar métodos de la capa de persistencia y realizar el registro adecuadamente. A continuación, la figura muestra los servicios para la gestión de resultados de exámenes panel metabólico básico.

```
package com.modulo.pbm.services;

import java.util.List;
import com.modulo.pbm.model.Exam;
import com.modulo.pbm.model.ExamReport;

public interface ExamService {

    public void insert(Exam exam);

    public void edit(Exam exam);

    public boolean find(String ndivalue, String order);

    public Exam select(String ndivalue, String order);

    public List<ExamReport> selectByUserCode(String username, String code);
    public List<ExamReport> selectAll(String ndivalue); //  <---
    public List<ExamReport> selectForLab();//  <---

    public Exam selectByOrder(String order);

    public void delete(String ndivalue, String order);

    Exam getInitialExamParameters(Exam exam);

    public boolean findUser(String username);
}
```

Figura 32. Servicios gestión de resultados de exámenes.

4.3.1.4 Persistencia

Se crean las clases DAO y en ellas se incluyen las operaciones sobre la base de datos usando JDBC y se definen las funciones para las operaciones solicitadas.

En la figura se muestra la implementación de los métodos para cada una de las acciones (Registrar, Consultar, Actualizar y Eliminar).

```
@Override
public void insert(Exam exam) {

    JsonReportExam jsonDataExam = new JsonReportExam(exam);
    System.out.println(jsonDataExam.getContained());
    String SQL_INSERT = "INSERT INTO diagnosticreport (resourceType, contained, identifier, "
            + "status,category, subject,code, effectivedatetime,issued,performer,conclusion) values ('"
            + jsonDataExam.getResourceType() + "', '" + jsonDataExam.getContained() + "', '"
            + jsonDataExam.getIdentifier() + "','" + jsonDataExam.getStatus() + "', '" + jsonDataExam.getCategory()
            + "', '" + jsonDataExam.getSubject() + "', '" + jsonDataExam.getCode() + "', '"
            + jsonDataExam.getEffectiveDateTime() + "','" + jsonDataExam.getIssued() + "', '"
            + jsonDataExam.getPerformer() + "','" + jsonDataExam.getConclusion() + "') ";

    try {
        dao.executeUpdate(SQL_INSERT);
    } catch (SQLException e) {
        e.printStackTrace();
    }

}
```

Figura 33. Método insert() para registrar un examen.

```
@Override
public Exam select(String ndivalue, String order) {

    String sqls = sql + " FROM diagnosticreport WHERE subject->'$.reference'='" + ndivalue + "' AND id = " + order;

    Exam exam = new Exam();
    try {
        Connection connection = dao.getDatasource().getConnection();
        ResultSet resulset = connection.prepareStatement(sqls).executeQuery();
        while (resulset.next()) {
            exam = getResulset(resulset);
        }
        connection.close();

    } catch (SQLException e) {
        e.printStackTrace();
    }

    return exam;

}
```

Figura 34. Método select() para seleccionar un examen por usuario e id.

```
@Override
public void edit(Exam exam) {

    JsonReportExam jsonDataExam = new JsonReportExam(exam);

    String SQL_UPDATE = "UPDATE diagnosticreport SET " + setData(jsonDataExam) + " WHERE id = " + exam.getOrder();

    try {
        dao.executeUpdate(SQL_UPDATE);
    } catch (SQLException e) {
        e.printStackTrace();
    }

}

@Override
public void delete(String ndivalue, String order) {
    String SQL_DELETE = "DELETE FROM diagnosticreport WHERE subject->'$.reference'='" + ndivalue + "' AND id = '"
            + order + "'";
    try {
        dao.executeUpdate(SQL_DELETE);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

Figura 35. Métodos edit() y delete(), editar y eliminar un examen.

El proceso de cada método al ser invocado es el siguiente, primero se establece una conexión con la base de datos, para obtener una conexión se hace uso del método getConnection de DriverManager y se le pasa los parámetros de la base de datos, el usuario y la contraseña. Una vez obtenida la conexión ya se puede hacer la consulta a la base de datos ejecutando una consulta de tipo PreparedStatement. Después de hacer las consultas que se necesite se cierra la conexión para liberar los recursos.

4.3.1.5 Controladores

Con los controladores se gestiona las peticiones redirigidas por el Servlet de Spring, y se retorna la respuesta al cliente usuario. En la siguiente figura se muestra como gestiona un controlador una petición. Una función para obtener el listado de exámenes pertenecientes a un paciente y la información de usuario, donde se invocan a los servicios correspondientes para llevar a cabo las operaciones.

```
@RequestMapping(value = "/personal/{ndivalue}/paciente", method = RequestMethod.GET)
public String consultPatient(@PathVariable("ndivalue") String ndivalue, Model model) {

    logger.debug("Agregar/Ver examenex paciente --> {}", ndivalue);
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    UserDetails userDetail = (UserDetails) auth.getPrincipal();
    Personal personal = personalService.select(userDetail.getUsername());
    model.addAttribute("name", personal.getGiven() + " " + personal.getFamily());

    model.addAttribute("examForm", new Exam());
    model.addAttribute("panelbasicList", new DefaultModel().panelbasicList());
    model.addAttribute("patient", patientService.select(ndivalue));
    model.addAttribute("exams", examService.selectAll(ndivalue));

    model.addAttribute("action", "paciente");
    model.addAttribute("roleType", "ROLE_PERSONAL");

    return "personal/exampatient";
}
```

Figura 36. Controlador información de usuario y exámenes.

```
@RequestMapping(value = "/personal/autorizarExamen", method = RequestMethod.POST)
public String autorizeExam(@ModelAttribute("examForm") Exam exam, BindingResult result, Model model,
        final RedirectAttributes redirectAttributes) {

    // User details session
    Authentication auth = SecurityContextHolder.getContext().getAuthentication();
    UserDetails userDetail = (UserDetails) auth.getPrincipal();

    Personal personal = personalService.select(userDetail.getUsername());
    exam.setDisplayPerformer(personal.getGiven() + " " + personal.getFamily());
    exam.setReferencePerformer(personal.getNdivalue());
    Date date = new Date();
    DateFormat hourdateFormat = new SimpleDateFormat("HH:mm:ss dd/MM/yyyy");
    exam.setIssued(hourdateFormat.format(date));

    exam.setStatus("Registered");
    examService.insert(exam);
    redirectAttributes.addFlashAttribute("css", "success");
    redirectAttributes.addFlashAttribute("msg", "Examen Registrado correctamente");

    return "redirect:/personal/" + exam.getReferenceSubject() + "/paciente";
}
```

Figura 37. Controlador autorizar examen.

En la figura 37 se muestra el controlador que realiza la autorización de un examen a un paciente seleccionado.

La anotación @Controller define el método como un controlador. @RequestMapping, con el atributo **value** que indica que */personal/autorizarExamen* es la URL que se le asigna al

método del controlador y el atributo **method** define el método de servicio para manejar la solicitud HTTP que es POST. Una vez ejecutada la lógica del método, este nos retorna un String, el cual se interpreta como el nombre de la vista a mostrar tras invocar el controlador.

Si el Servlet no encuentra un controlador adecuado para gestionar la petición, retornará un error 404 de Recurso no encontrado.

4.3.1.6 Autenticación y autorización

En cualquier tipo de aplicación se hace necesaria una gestión de la seguridad y de acceso a los recursos disponibles. En el sistema se establecen niveles para el acceso a los recursos, que corresponden con los roles de los usuarios: el Administrador, que tendrá acceso a los recursos disponibles en cuya ruta se incluya el patrón ***/admin***; el Médico, que tendrá acceso a los recursos disponibles en cuya ruta se incluya el patrón ***/personal***; el Laboratorista que tendrá acceso a los recursos disponibles en cuya ruta se incluya el patrón ***/lab*** y por último el Paciente que tendrá acceso a los recursos disponibles en cuya ruta se incluya el patrón ***/patient***.

Para ello se define los roles de usuario y cada usuario podrá tener un rol. Para llevar a cabo esto se crea una tabla users con dos campos de username y password y tiene una relación con la tabla user_roles la cual almacena el usuario y su rol. Estos roles son ROLE_ADMIN, ROLE_PERSONAL, ROLE_LAB y ROLE_PATIENT.

En el archivo de configuración de Spring Security root-context.xml se configura AuthenticationManager como se muestra en la figura 38.

```
<authentication-manager>
    <authentication-provider>
        <jdbc-user-service data-source-ref="dataSource"
            users-by-username-query="select username,password, enabled from users where username=?"
            authorities-by-username-query="select username, role from user_roles where username =?  " />
    </authentication-provider>
</authentication-manager>
```

Figura 38. Configuración AuthenticationManager.

En la figura se puede observar dos querys que se usan:

- Para autenticar: select username, password, enabled from users where username=?
- Para autorizar: select username, role from user_roles where username =?

Mediante el http introducimos los patrones url a controlar con los permisos asociados por AuthenticationManager, en la figura 39 se muestra la configuración de acceso a los recursos por rol.

```
<http auto-config="true" use-expressions="true">

    <intercept-url pattern="/admins/**" access="hasRole('ROLE_ADMIN')" />
    <intercept-url pattern="/temp/**" access="hasRole('ROLE_TEMPORAL')" />
    <intercept-url pattern="/patient/**" access="hasRole('ROLE_PATIENT')" />
    <intercept-url pattern="/personal/**" access="hasRole('ROLE_PERSONAL')" />
    <intercept-url pattern="/lab/**" access="hasRole('ROLE_LAB')" />
    <!-- access denied page -->

    <access-denied-handler error-page="/403" />
    <form-login login-page="/login" default-target-url="/"
        authentication-failure-url="/login?error" username-parameter="username"
        password-parameter="password" />
    <logout logout-success-url="/login?logout" />
    <!-- enable csrf protection request-matcher-ref="csrfMatcher" -->
    <csrf />

    <session-management>
        <concurrency-control max-sessions="1"
            expired-url="/login" />
    </session-management>

</http>
```

Figura 39. Configuración de acceso a los recursos.

4.3.1.7 JDBC

Se utilizó JDBC, el conector JDBC de MySQL para poder conectarnos a la base de datos (DB) y así poder realizar todas las operaciones requeridas por los usuarios, relacionadas con datos de los mismos. Para implementar esta funcionalidad importante, en el sistema se crea un archivo database.properties en la carpeta resources del proyecto, el cual contiene las propiedades de conexión a la DB como se muestra en la figura 40.

```
driverClassName = com.mysql.jdbc.Driver
url = jdbc:mysql://localhost/
database = basicpanel
username = root
password = root
scriptDatabase = /BasicPanelScriptDB.sql
```

Figura 40. Propiedades base de datos.

Una vez listas las propiedades de conexión se crea la configuración del conector JDBC, para ello se crea el archivo spring-database.xml donde se define el bean de origen de propiedades y el bean de origen de datos figura 41.

```
<bean
    class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
    <property name="location" value="classpath:database.properties" />
</bean>


<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${driverClassName}" />
    <property name="url" value="${url}${database}" />
    <property name="username" value="${username}" />
    <property name="password" value="${password}" />
</bean>
```

Figura 41. Configuración JDBC.

4.3.1.8 Internacionalización

En muchas ocasiones es necesario tener nuestra aplicación web en varios idiomas. Para ello lo primero que necesitamos en un bean que implemente la interface MessageSource para localizar el origen de los archivos que contienen el texto en varios idiomas. Estos archivos tendrán la extensión .properties y un sufijo que indicará el idioma. A continuación, se muestran los nombres de los archivos para español e inglés:

- Para español messages_es.properties.
- Para ingles messages_en.properties.

Lo único que tenemos que hacer es crear un fichero de texto por cada idioma que vayamos a usar y ubicarlo en el paquete /src/main/resources donde se encuentran los recursos.

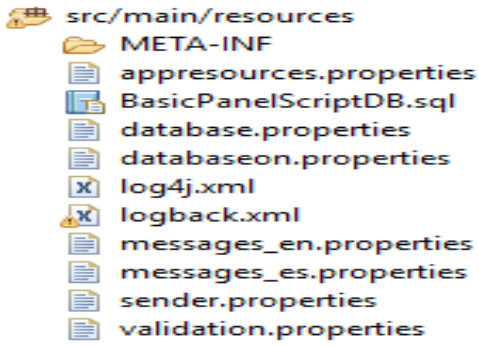


Figura 42. Paquete recursos.

Cada fichero de texto, contendrá claves y valores:

Tabla 24. Claves y valores, español e ingles

messages_es.properties	messages_en.properties
...	...
exam.glucosa = GLUCOSA	exam.glucosa = GLUCOSE
exam.bun = NITROGENO UREICO	exam.bun = UREIC NITROGEN
exam.creatinina = CREATININA	exam.creatinina = CREATININE
exam.sodio = SODIO	exam.sodio = SODIUM
exam.potasio = POTASIO	exam.potasio = POTASSIUM
exam.calcio = CALCIO	exam.calcio = CALCIUM
exam.cloro = CLORO	exam.cloro = CHLORIDE
exam.co2 = DIOXIDO DE CARBONO	exam.co2 = CARBON DIOXID
...	...

(Elaboración propia)

Ahora para utilizar el valor de las claves, en los JSP tenemos la opción de hacerlo mediante los tags de Spring.

```
<h1><spring:message code="exam.glucosa" /></h1>
```

Si no se especifica un LocaleResolver, se utiliza por defecto AcceptHeaderLocaleResolver, el cual devuelve el valor del idioma establecido en las preferencias del navegador y viene reflejado en las cabeceras de las peticiones HTTP.

Para que el usuario pueda realizar el cambio de idioma cuando desee, para darle esta característica es necesario utilizar el bean LocaleChangeInterceptor y añadirlo a la lista de interceptores en la configuración como se muestra a continuación.

```
@Bean
public LocaleChangeInterceptor localeChangeInterceptor() {
 LocaleChangeInterceptor localeChangeInterceptor = new LocaleChangeInterceptor();
        localeChangeInterceptor.setParamName("lang");
        return localeChangeInterceptor;
    }

 @Override

    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(localeChangeInterceptor());
```

```
    }
```

LocaleChangeInterceptor este intercepta la petición HTTP y cambia el idioma actual si la URL contiene el parámetro lang=idioma donde idioma es el código del lenguaje deseado: es o en, el nombre del parámetro lo establecemos con el método setParamName("lang"), el cual se llama lang. A continuación, se muestran la implementación de los links para hacer el cambio de idioma.

```
<c:if test="${pageContext.response.locale.language == 'es'}">
<a href="?lang=en"><i class="fa fa-language fa-2x text-warning" aria-hidden="true"></i></a>
</c:if>

<c:if test="${pageContext.response.locale.language == 'en'}">
<a href="?lang=es"><i class="fa fa-language fa-2x text-success" aria-hidden="true"></i></a>
</c:if>
```

4.3.1.9 Interfaz de Usuario

En esta sección se comenta como se realizaron algunos de los elementos mas importantes que componen las interfaces.

4.3.1.9.1 Material Design for Bootstrap 4 (MDB4)

Los estilos de los elementos que componen las interfaces se realizan mediante el uso de MDB4, estos elementos se definen mediante el uso de etiquetas html.

Teniendo en cuenta lo mencionado anteriormente, se añadirá el siguiente código en cada archivo JSP para poder comenzar a trabajar con MDB4.

```
<! -----CSS files----->

<link href="<c:url value='/resources/material/css/bootstrap.min.css' />"
        rel="stylesheet"></link>
<link href="<c:url value='/resources/material/css/mdb.min.css' />"
        rel="stylesheet"></link>
<link href="<c:url value='/resources/material/css/style.css' />"
        rel="stylesheet"></link>

<! -----JS files----->

<script src="<c:url value='/resources/material/js/bootstrap.min.js' /> "></script>
<script src="<c:url value='/resources/material/js/mdb.min.js' /> "></script>
```

A continuación, se presentarán las clases más usadas en la aplicación.

Container, la clase "container" consiste en que el elemento del DOM en el que se aplica dicha clase tenga un tamaño proporcional a la pantalla del dispositivo que está haciendo uso de la aplicación web. La clase "container" distingue cuatro rangos de screen mediante el uso de las reglas @import de CSS, menos de 768px (donde el tamaño del "container" será del 100% del ancho de la pantalla), entre 769 y 991 px (donde el tamaño será de 769px), entre 992 y 1199px (donde el tamaño será de 992px) y más de 1200px (donde el tamaño será de 1200px).

Col y Row, la clase "col" es la base del sistema de columnas de Bootstrap utilizado para optimizar una página web a la hora de realizar un "responsive design". Este sistema básicamente divide el elemento del DOM en el que se esté trabajando en doce partes, y además tiene en cuenta los rangos de pantalla en los que se encuentra.

```
<div class="row">
<div class="col-md-4">
<!--Main column-->
</div>
</div>
```

Hidden/visible, Las clases "hidden" y "visible" se utilizan para hacer que un elemento sea mostrado o no en la vista que se especifique.

Btn, esta clase es aplicable a los botones (buttons en HTML) y links (a en HTML), y se encarga de darles un formato con borde redondeado y propiedades para "active" y "hover" de manera que cambien al pasar el puntero o cuando el usuario haga clic en ellos. Además, esta clase se combinará con otras dos clases, una para la combinación de colores del botón (btn-danger, btn-warning, btn-info, btn-success, btn-primary etc), y otra para el tamaño del botón (btn-xs, btn-sm, btn-md o btn-lg).

```
<! — botón Registrar-->
<button type="submit" class="btn btn-success btn-sm">REGISTRAR</button>
<! — link con diseño botón realizer observaciones-->
<a href="${exampacienteUrl}" title="Realizar Observaciones" class="btn-floating btn-warning btn-
sm"></a>
```

Text-center, pull-left y pull-right, estas clases sirven para alinear elementos en la pantalla; con la clase "text-center" se podrá centrar texto en un elemento, con la clase "pull-left" se podrá alinear a la izquierda, y con la clase "pull-right" se podrá alinear a la derecha.

Card, es un contenedor de contenido flexible y extensible. Incluye opciones para encabezados y pies de página, colores de fondo y otras opciones de visualización.

mt, mb, mr, ml, son clases para configurar la margen entre contenidos top, bottom, right y left.

Navbar, esta clase es solo aplicable a las barras de navegación (nav en HTML) se encarga de darles un estilo más llamativo y propiedades para que sean responsivas.

```
<nav class="navbar  navbar-expand-lg navbar-dark green ">
</nav>
```

4.3.1.9.2 Font Awesome

Para empezar a usarlo se agrega el siguiente código en las páginas JSP donde se hará uso de iconos.

```
<link href="<c:url value="/resources/fontawesome/css/fontawesomeall.min.css"/>"
rel="stylesheet" />
```

Una vez referenciado el framework se puede usar en cualquier lugar con la etiqueta . a continuación, se muestra el código donde se hace el uso de iconos en un CARD y un link con diseño botón.

```
<div class="text-center mt-3"><i class="fas fa-user fa-5x light-green-text"></i></div>

<a href="#" title="Eliminar Resultado" class="btn-floating btn-danger btn-sm"><i class="fas fa-
trash"></i></a>
```

4.3.1.9.3 DataTables

Para empezar a usarlo se agrega el siguiente código en las páginas JSP donde se hará uso de tablas.

```
<link rel="stylesheet" href="<c:url
value='/resources/datatable/css/dataTables.bootstrap4.min.css' />"></link>

<script src="<c:url value='/resources/datatable/js/dataTables.bootstrap4.min.js' /> "></script>
<script src="<c:url value='/resources/datatable/js/jquery.dataTable.min.js' /> "></script>
```

Para su implementación se llama la función que inicializa la tabla.

```
<script type="text/javascript">
```

```
$(document).ready(function() {
$('#TablaExam').DataTable();
});
</script>
```

A continuación, se muestra el código de las clases que dan el diseño a la tabla.

```
<table id="TablaExam" class="table table-striped table-bordered dt-responsive
nowrap" cellspacing="0" width="100%"></table>
```

4.3.1.9.4 Toasttr

Es una librería que se usa en el proyecto para notificaciones las cuales se encargan de mostrar mensajes de error, advertencia o éxito de las peticiones del cliente al servidor. Para empezar a usarla se agrega el siguiente código en las páginas JSP donde se hará uso de notificaciones.

```
<link href="<c:url value='/resources/toasttr/toastr.css' />" rel="stylesheet"></link>
<script src="<c:url value='/resources/toasttr/toastr.js' />"></script>
```

Para su Implementación se llama la función toastr(), a la cual se le pasan dos parámetros provenientes del servidor, el css el cual indica que tipo de notificación será de error, éxito o advertencia y el msg el cual contiene el mensaje. Dentro de esta función se establecen las características de animación, posición y duración de la notificación.

```
<script>
        window.onload = function() {
                toastr["${css}"]("", "${msg}",{
                                        "closeButton" : false,
                                        "debug" : false,
                                        "newestOnTop" : false,
                                        "progressBar" : false,
                                        "rtl" : false,
                                        "positionClass" : "toast-bottom-right",
                                        "preventDuplicates" : false,
                                        "onclick" : null,
                                        "showDuration" : 700,
                                        "hideDuration" : 700,
                                        "timeOut" : 5000,
                                        "extendedTimeOut" : 1000,
                                        "showEasing" : "swing",
                                        "hideEasing" : "linear",
                                        "showMethod": "slideDown",
                                        "hideMethod": "slideUp"
                                                })
```

```
                    }
</script>
```

4.3.2 Aplicación Android

4.3.2.1 Interfaz de Usuario (UI)

Android ofrece una variedad de componentes de IU previamente compilados, como objetos de diseño estructurados y controles de IU que permiten compilar la interfaz gráfica de usuario para la aplicación. las interfaces en Android se realizan mediante ficheros con etiquetas xml que definen elementos a los cuales se darán funcionalidades mediante el uso de código.

A continuación, se muestra algunas de las implementaciones de los elementos que componen la aplicación.

TextInputLayout, la validación requiere un formulario que contenga los controles necesarios para que el usuario especifique la información para la correcta autenticación. Para ello se crea un layout al cual se le añaden elementos TextInputLayout los cuales se encargarán de recibir la información que ingresa el usuario, como el número de documento y la contraseña. Además de darle un estilo elegante con etiquetas flotantes.

```
<! -- Username-->
<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_marginBottom="8dp"
   android:layout_marginTop="8dp">

   <EditText
      android:id="@+id/input_nodocumento"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:hint="Nro Documento"
      android:inputType="text|number" />
</android.support.design.widget.TextInputLayout>

<!-- password -->
<android.support.design.widget.TextInputLayout
   android:layout_width="match_parent"
   android:layout_height="wrap_content"

   <EditText
      android:id="@+id/input_password"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
```

```
    android:hint="Contraseña"
    android:inputType="textPassword" />
</android.support.design.widget.TextInputLayout>
```

AppCompatButton, para que el usuario interactúe con el contenido se crean botones para realizar acciones al pulsar sobre ellos. Agregarlos a la aplicación es sencillo.

```
<!-- button -->
<android.support.v7.widget.AppCompatButton
  android:id="@+id/login"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="INICIAR SESIÓN" />
```

TextView, corresponde a una etiqueta de texto simple, que sirve para mostrar un texto al usuario. Su atributo más importante es android:text, cuyo valor indica el texto a mostrar por pantalla. También pueden ser de interés los atributos android:textColor y android:textSize.

```
<TextView
    android:id ="@+id/tv_valor_glucosa "
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text=" "/>
```

LinearLayout, layout que organiza sus componentes en una única fila o una única columna, según éste sea horizontal o vertical, respectivamente. Para establecer la orientación podemos utilizar o bien el atributo android:orientation en la definición del layout en el archivo XML (tomando como valor horizontal o vertical, siendo el primero el valor por defecto).

```
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="vertical">

  <include
    android:id="@+id/include"
    layout="@layout/card_view_info"                                    />

  <include
    android:id="@+id/includes"
    layout="@layout/card_view_results"                                 />

</LinearLayout>
```

TableLayout, se trata de un layout que organiza sus elementos como una rejilla dividida en filas y columnas. Se compone de un conjunto de elementos de tipo TableRow que representan a las diferentes filas de la tabla. Cada fila a su vez se compone de un conjunto de vistas.

```
<TableLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:orientation="horizontal"
  android:layout_margin="16dp"
  android:weightSum="1">
  <TableRow
    android:id="@+id/tr_head"
    android:background="@drawable/table_row_bg"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:backgroundTint="#bdbdbd"
    android:padding="5dp">
    <TextView
      android:layout_width="0dp"
      android:layout_height="wrap_content"
      android:layout_weight="0.25"
      android:padding="5dp"
      android:text="PANEL METABOLICO BASICO"/>
  </TableRow>
</TableLayout>
```

RelativeLayout, este layout es muy flexible; permite indicar la posición de un elemento en función de otros y de los bordes de la pantalla.

```
<RelativeLayout
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:background="?attr/selectableItemBackground"
  android:clickable="true"
  android:padding="@dimen/activity_horizontal_margin">

  <ImageView
    android:id="@+id/ivContactItem1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_marginRight="@dimen/activity_horizontal_margin"
    android:src="@drawable/ic_id" />

  <TextView
    android:id="@+id/tv_ndivalue"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:layout_marginLeft="@dimen/activity_horizontal_margin"
    android:layout_toRightOf="@id/ivContactItem1"
    android:textAppearance="@style/TextAppearance.AppCompat.Medium"
    android:textColor="#212121"                                          />
/>

</RelativeLayout>
```

DrawerLayout, panel lateral de navegación donde se muestran las opciones principales de la aplicación y está en el borde izquierdo de la pantalla oculto, aparece cuando el usuario desliza el dedo o cuando el usuario toca el ícono menú de la app en la barra de acciones.

```
<android.support.v4.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/drawer_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:fitsSystemWindows="true"
  tools:openDrawer="start">

  <include
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />

  <android.support.design.widget.NavigationView
    android:id="@+id/nav_view"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:fitsSystemWindows="true"
    app:headerLayout="@layout/nav_header_main"
    app:menu="@menu/activity_main_drawer" />
</android.support.v4.widget.DrawerLayout>
```

Menu, para definir el menú, se crea un archivo activity_main_drawer.xml y se añade el menú con los elementos:

- El <menu> define un menú, que es un contenedor para elementos del menú, además puede tener uno o más elementos <item> y <group>.
- El <item> representa un único elemento en el menú. Este elemento puede contener un elemento <menu> anidado para crear un submenú.
- El <group> contenedor para elementos <item> para asignarles estado o visibilidad.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <group
    android:id="@+id/menu_patient"
    android:checkableBehavior="single"
    android:visible="false">
    <item
      android:id="@+id/nav_list_exam"
      android:icon="@drawable/ic_results"
      android:title="Mis Resultados" />
    <item
      android:id="@+id/nav_chart"
      android:icon="@drawable/ic_chart128"
      android:title="Gráficas" />
  </group>
  <group
    android:id="@+id/menu_personal"
    android:checkableBehavior="single"
    android:visible="false">
    <item
      android:id="@+id/nav_search"
      android:icon="@drawable/ic_results"
      android:title="Buscar" />
  </group>
  <group
    android:id="@+id/menu_lab"
    android:checkableBehavior="single"
    android:visible="false">
    <item
      android:id="@+id/nav_lab_no_done"
      android:icon="@drawable/ic_results"
      android:title="Laboratorios" />
  </group>
  <item android:title="Información">
    <menu>
      <item
        android:id="@+id/nav_about_pmb"
        android:icon="@drawable/ic_info128"
        android:title="Info PMB" />
      <item
        android:id="@+id/nav_about_app"
        android:icon="@drawable/ic_menu_send"
        android:title="Info App" />
    </menu>
  </item>
</menu>
```

SearchView, se crea un archivo de menú search_main.xml que contiene un elemento responsable de mostrar un botón de búsqueda. Cuando el usuario oprime este botón, éste se expande y muestra un campo de texto que permite que el usuario ingrese el número de

documento del paciente a buscar. El asistente responsable de esta tarea es SearchView además hay que definir otros atributos como app:showAsAction con el valor collapseActionView permite que el SearchView se expanda cuando se selecciona el botón.

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  tools:context=".SearchActivity">

  <item
    android:id="@+id/action_search"
    android:icon="@android:drawable/ic_menu_search"
    app:showAsAction="ifRoom|collapseActionView"
    app:actionViewClass="android.support.v7.widget.SearchView"
    android:title="Search"
    android:iconifiedByDefault="true"
    android:showAsAction="ifRoom|collapseActionView"
    android:actionViewClass="android.widget.SearchView"
    tools:ignore="AppCompatResource" />
</menu>
```

FloatingActionButton, botón flotante para su implementación se usa una etiqueta <android.support.design.widget.FloatingActionButton>, a la cual se le añade todos los atributos para el diseño del botón.

```
<android.support.design.widget.FloatingActionButton
  android:id="@+id/fab_view"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="center_vertical|left"
  android:layout_margin="@dimen/fab_margin"
  android:src="@drawable/ic_view"
  app:layout_anchor="@+id/app_bar_layout"
  app:layout_anchorGravity="bottom|end" />
```

ReciclerView, sirve para mostrar grandes conjuntos de datos que se pueden desplazar de manera muy eficiente. Para ello se crea el archivo fragment_list_exam.xml al cual se le añade el ReciclerView donde se van a mostrar la lista de resultados de un paciente.

```
<android.support.v7.widget.RecyclerView
  android:id="@+id/reciclador"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:padding="0dp"
  android:scrollbars="vertical" />
```

Cada ítem de la lista que se carga en el **ReciclerView** presenta detalles básicos de cada examen como el id, el nombre del examen y la fecha de autorización que están dentro de un **CardView** que permite mostrar información dentro de tarjetas que tienen una apariencia uniforme que en su diseño pueden tener sombras y esquinas redondeadas para esto se coloca los elementos entre las etiquetas <android.support.v7.widget.CardView>.

```
<android.support.v7.widget.CardView
   xmlns:card_view="http://schemas.android.com/apk/res-auto"
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/card_view"
   android:layout_width="match_parent"
   android:layout_height="80dp"

   android:foreground="?android:attr/selectableItemBackground">
</android.support.v7.widget.CardView>
```

Para la representación gráfica de los datos de cada variable del examen Panel Metabólico Básico se hace uso de MpAndroidChart Library, con la cual se puede dibujar un gráfico de líneas en la Aplicación Android.

Para lograr esto se crea el archivo activity_chart.xml y se le añade en el diseño de la actividad la etiqueta <com.github.mikephil.charting.charts.LineChart/>.

```
<com.github.mikephil.charting.charts.LineChart
   android:id="@+id/chart"
   android:layout_width="match_parent"
   android:layout_height="match_parent" />
```

4.3.2.2 Web Service

Antes de desarrollar la aplicación Android primero se implementó el Web Service, esto se hizo en Spring con las funcionalidades necesarias para realizar operaciones sobre la base de datos en Mysql a través de peticiones GET y POST.

Para llevar a cabo el proceso se crean clases de representación de recursos y se comienza pensando en las interacciones del servicio. En esta sección se hablará de algunos servicios GET y POST ya que todos manejan la misma dinámica.

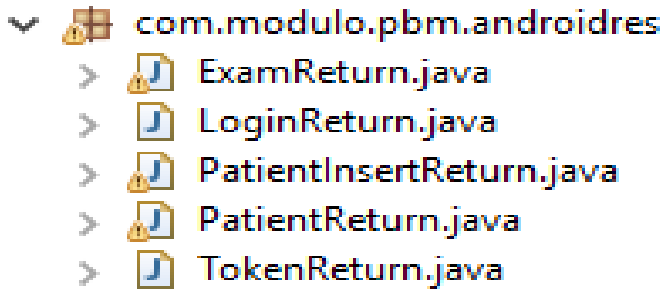


Figura 43. Recursos Web Services.

4.3.2.2.1 Inicio de Sesión

Para modelar la representación Inicio de Sesión, se crea una clase de representación de recursos LoginReturn.java que proporciona un objeto POJO con campos, constructores y atributos para los datos validate, username, message, displayUser y role.

```
package com.modulo.pbm.androidres;
public class LoginReturn {

        String validate;
        String username;
        String mesagge;
        String displayUser;
        String role;

//getters and setters

}
```

El campo validate es un identificador único para el inicio de sesión que indica el estado de la petición, username, message, displayUser y role forman la representación textual de la solicitud.

A continuación, se crea el controlador de recursos que servirá estas respuestas. Las solicitudes HTTP son manejadas por un controlador. Estos componentes se identifican fácilmente mediante la anotación @RestController.

```
@RequestMapping(value = "/s/{username}/{password}", method = RequestMethod.GET)
@ResponseBody
public LoginReturn getpatient(@PathVariable String username, @PathVariable String password) {
    logger.info("getting user with id: {}", username);

    boolean users = userService.find(username);
    User user = userService.select(username);
    LoginReturn data = new LoginReturn();

        if (users) {
            if (user.getPassword().equals(password)&& user.isPatient()) {
                Patient datapatient = patientService.select(username);

                data.setUsername(username);
                data.setValidate("1");
                data.setRole("3");
                data.setDisplayUser(datapatient.getGiven() +" "+ datapatient.getFamily());
                data.setMesagge("Bienvenido " + datapatient.getGiven() +" "+ datapatient.getFamily());
                System.out.println(user.getRole());
                return data;

            }
            else {
                data.setUsername(username);
                data.setValidate("2");
                data.setMesagge("Contraseña incorrecta");
                return data;

            }
    }
    data.setUsername(username);
    data.setValidate("3");
    data.setMesagge("Usuario no registrado");
    return data;

}
```

Figura 44. Controlador Web Service inicio de sesión.

De la figura 58, la anotación @RequestMapping asegura que las solicitudes HTTP mapeen al método. La anotación @ResponseBody hará que Spring entienda que el valor de retorno del método debe estar vinculado al cuerpo de respuesta web.

La implementación del método crea y devuelve un objeto LoginReturn con sus atributos en formato JSON. A continuación, se muestra la respuesta JSON a un usuario que ingresa correctamente sus datos de inicio de sesión.


```
{
  "validate":"1",
  "username":"2",
  "mesagge":"Bienvenido Usuario Medico 1",
  "displayUser":"Usuario Medico 1",
  "role":"1"
}
```

4.3.2.2.2 Detalles de usuario Paciente

Para la representación del recurso detalles de usuario se crea la clase PatientReturn.java que proporciona un objeto JSON con los campos y atributos para los datos validate, message y patient.

```
package com.modulo.pbm.androidres;

import java.util.List;
import com.modulo.pbm.model.Exam;
import com.modulo.pbm.model.Patient;

public class PatientReturn {
        String estate;
        String message;
        Patient patient;

        //getters and setters
}
```

El campo validate es un identificador único para la solicitud de los detalles de un usuario paciente que indica el estado de la petición, message y patient la representación textual de la solicitud.

A continuación, la figura muestra el controlador.

```
@RequestMapping(value = "/search/{username}", method = RequestMethod.GET)
public @ResponseBody PatientReturn getPatient(@PathVariable("username") String username) {

    PatientReturn rpatient = new PatientReturn();
    boolean fpatient = patientService.find(username);

    if (fpatient) {
        Patient patient = patientService.select(username);

        rpatient.setEstate("1");
        rpatient.setMessage("ok");
        rpatient.setPatient(patient);
    } else {
        rpatient.setEstate("2");
        rpatient.setMessage("Paciente no registrado");

    }
    return rpatient;

}
```

Figura 45. Controlador Web Service detalles de usuario paciente.

La implementación del método crea y devuelve un objeto PatientReturn con sus atributos en formato JSON que contiene toda la información del paciente solicitado por número de documento. A continuación, la respuesta JSON a la solicitud de búsqueda de un paciente existente en la base de datos.


```
{
   "estate": "1",
   "message": "ok",
   "patient": {
      "given": "Usuario ",
      "family": "Paciente 1",
      "password": null,
      "ndi": "C.C",
      "ndivalue": "4",
      "gender": "F",
      "birthDate": "02-02-1999",
      "maritalStatus": "M",
      "telhome": "4",
      "telmobile": "4",
      "telwork": "4",
      "email": "ncamiloq@hotmail.com",
      "line": "su municipio",
      "city": "su casa",
      "givenc": "su contacto ",
      "familyc": "apellido su contacto",
      "telc": "4",
      "relationship": "family",
      "managingOrganization": "asmet salud",
      "bloodtype": "O+",
      "id": 3,
      "confirmPassword": null,
       "age": "19 años, 0 meses y 1 días",
      "estadocivil": "Casado(a)",
      "parentesco": "Familiar",
      "genero": "Femenino",
       }
}
```


4.3.2.2.3 Autorizar examen

A través de la anotación @RequestBody, Spring enlazará el cuerpo de la solicitud HTTP entrante con el parámetro exam, una vez hecho eso Spring usará convertidores de mensajes HTTP para convertir el cuerpo de solicitud HTTP en objeto de dominio.

```
@RequestMapping(value = "/iex", method = RequestMethod.POST)
public @ResponseBody ExamReturn subscribeExam(@RequestBody Exam exam) {

    ExamReturn rexam = new ExamReturn();
    examService.insert(exam);

    rexam.setEstate("1");
    rexam.setMessage("Resultado registrado correctamente ");

    return rexam;

}
```

Figura 46. Controlador Web Service autorizar examen.

La figura 46, muestra el método del controlador para manejar la típica solicitud POST (para la url /iex), este, crea un examen y devuelve un objeto ExamReturn que contiene el estado y mensaje de la operación.

Una vez creado el Web Service, se empieza a implementar las funciones de la aplicación teniendo en cuenta los elementos e interacciones de la arquitectura que son necesarios.

- Crear un patrón Volley Singleton para las peticiones.
- Crear la petición personalizada para tratar respuestas JSON.
- Crear un adaptador que procese los elementos del recycler view.
- Tratar los eventos para la comunicación de datos a través de los controles.

4.3.2.3 Volley Singleton

Debido a que la aplicación hace uso constante de la red, para que sea más eficiente se configura una única instancia de RequestQueue para la vida útil de la aplicación. Por lo que se crea una clase VolleySingleton.java que contiene RequestQueue y addToRequestQueue.

A continuación, se muestra el código de la clase VolleySingleton.java.

```
public final class VolleySingleton {
  // Atributos
  private static VolleySingleton singleton;
  private RequestQueue requestQueue;
```

```
  private static Context context;

  private VolleySingleton(Context context) {
     VolleySingleton.context = context;
     requestQueue = getRequestQueue();
  }
 //  Retorna la instancia unica del singleton

  public static synchronized VolleySingleton getInstance(Context context) {
     if (singleton == null) {
        singleton = new VolleySingleton(context.getApplicationContext());
     }
     return singleton;
  }
   //Obtiene la instancia de la cola de peticiones

  public RequestQueue getRequestQueue() {
     if (requestQueue == null) {
        requestQueue = Volley.newRequestQueue(context.getApplicationContext());
     }
     return requestQueue;
}
   //  Añade la petición a la cola
  public <T> void addToRequestQueue(Request<T> req) {
     getRequestQueue().add(req);
  }
}
```

getInstance() asigna memoria a la instancia del singleton, donde se llama al constructor privado de la clase, a este método se le añade la propiedad synchronized, ya que la instancia será accedida desde varios hilos por lo que es necesario evitar bloqueos de acceso. El método getRequestQueue() obtiene la instancia de la cola de peticiones que se usará a través de la aplicación. Para agregar una nueva petición se hace uso del método addToRequestQueue().

Para acceder a las URLs del web service, se crea la clase Constantes.java para añadir todas las referencias que contienen las direcciones y otras variables si se necesita como se muestra en la figura 47.

```java
package com.panelbasic.msi.mipmb.model;


public class Constantes {
    /**
     * URLs del Web Service
     * http://url/basicpanel/
     */
    private static final String IP = "192.168.1.9:8080";
    public static final String URL_SERVER = "http://" + IP + "/basicpanel/";

    public static final String GET_LIST_EXAM_NO_DONE = "http://" + IP + "/basicpanel/lend";
    public static final String GET_BY_ID_EXAM_NO_DONE = "http://" + IP + "/basicpanel/end";
    public static final String LOGIN = "http://" + IP + "/basicpanel/s/";
    public static final String INSERT_TOKEN = "http://" + IP + "/basicpanel/it";
    public static final String GET_BY_ID_LIST_EXAM = "http://" + IP + "/basicpanel/lex/";
    public static final String GET_BY_ID_EXAM = "http://" + IP + "/basicpanel/ex/";
    public static final String GET_DATA_CHART = "http://" + IP + "/basicpanel/dc/";


    public static final String EXTRA_ID = "IDEXTRA";
    public static final String EXTRA_ORDER = "ORDER";


}
```

Figura 47. Clase Costantes.java.

4.3.2.4 Objetos de información

Para obtener los datos se definen los objetos de información como paciente y examen para los cuales se crea una clase Patient.java y Exam.java respectivamente indicando los atributos que lo forman e identificándolos con campos de la base de datos. En estos objetos es donde se guardará la información obtenida de las peticiones.

```java
package com.panelbasic.msi.mipmb.model;

public class Exam {

    private Integer order;
    private String code;
    private String systemCode;
    private String displayCode;
    private String displayCodeFull;
    private String displaySubject;
    private String referenceSubject;
    private String low;
    private String high;
    private String value;
    private String unit;
    private String labComments;
    private String performerComments;
    private String issued;
    private String referencePerformer;
    private String displayPerformer;
    private String referenceLab;
    private String displayLab;

    private Integer done;
    private String interpretation;
    private String valref;

    // metodos get y set
```

```java
package com.panelbasic.msi.mipmb.model;

public class Patient {

    public String given;
    public String family;
    public String password;
    public String ndi;
    public String ndivalue;
    public String gender;
    public String birthDate;
    public String maritalStatus;
    public String telhome;
    public String telmobile;
    public String telwork;
    public String email;
    public String line;
    public String city;
    public String givenc;
    public String familyc;
    public String telc;
    public String relationship;
    public String managingOrganization;
    public String bloodtype;
    public int id;
    public String confirmPassword;
    public int counta;
    public String age;
    public String estadocivil;
    public String parentesco;
    public String genero;
    public String profesion;
    //metodos get y set...
```

Figura 48. Clases Exam.java y Patient.java

4.3.2.5 Inicio de sesión

Para acceder a las funciones de la aplicación se debe primero iniciar sesión, para ello se crea la clase LoginActivity.java esta realiza la comunicación inicial con el servidor. Por lo que debemos dirigirnos al LoginActivity y generar una petición POST hacia el servidor.

```
public void IniciarSesion() {
   String usuario = user.getText().toString().trim();
   String contraseña = pass.getText().toString().trim();
   Map<String, String> map = new HashMap<String, String>();
   // Mapeo previo
   map.put("username", usuario);
   map.put("password", contraseña);
   // Crear nuevo objeto Json basado en el mapa
   JSONObject jobject = new JSONObject(map);
   // obtener datos en el servidor
   VolleySingleton.getInstance(this).addToRequestQueue(
         new JsonObjectRequest(
               Request.Method.POST, Constantes.LOGIN, jobject,
               new Response.Listener<JSONObject>() {
                  @Override
                  public void onResponse(JSONObject response) {
                     // Procesar la respuesta del servidor
                   procesarRespuesta(response);
                  }
               },
               new Response.ErrorListener() {
                  @Override
                  public void onErrorResponse(VolleyError error) {
               Toast.makeText(LoginActivity.this, "SIN RESPUESTA DEL SERVIDOR",
Toast.LENGTH_LONG).show();
                  }
               }
         ) {
            @Override
            public Map<String, String> getHeaders() {
               Map<String, String> headers = new HashMap<String, String>();
               headers.put("Content-Type", "application/json; charset=utf-8");
               headers.put("Accept", "application/json");
               return headers;
            }
            @Override
            public String getBodyContentType() {
               return "application/json; charset=utf-8" + getParamsEncoding();
            }
         }
   );}
```

Esta es la clase encargada de realizar las peticiones al servidor para comprobar que el usuario que inicia sesión existe y es válido. Esta funcionalidad es accedida desde el evento de pulsación del botón de inicio de sesión, el cual llama al método IniciarSesion() de la clase, este método será el encargado de verificar los detalles del usuario en el servidor realizando la petición HTTP POST gracias a la librería Volley. Antes de llamar a este método se comprueba que los campos estén completos.

En el método IniciarSesion(), primero se crea un ProgressDialog para indicar que el sistema está llevando a cabo una actividad que puede durar más tiempo de lo normal, luego se crea la petición HTTP, dentro de la cual encontramos todos los métodos que serán utilizados para controlarla, esta petición necesitará los siguientes parámetros:

- El tipo de petición, para este caso Method.POST
- La url a donde se dirige, definida en la clase Constantes, LOGIN.
- El objeto que contiene la información del usuario, número de documento y contraseña.
- Un listener en caso de que la petición sea correcta, procesa la respuesta invocando el método procesarRespuesta().
- Un listener en caso de que la petición devuelva un error si la petición solicitada al servidor ha devuelto algún tipo de error, ya sea de tiempo de espera o de otro tipo, este se mostrará al usuario, permitiéndole volver a introducir sus credenciales para un nuevo intento.

```
private void procesarRespuesta(JSONObject response) {
    String documento = user.getText().toString().trim();
    try { // Obtener atributo "mensaje"
        String validate = response.getString("validate");
        String username = response.getString("username");
        String mesagge =response.getString("mesagge");
        String displayUser =response.getString("displayUser");
        String role = response.getString("role");
        switch (validate) {
            case "1":
                ndivalue = user.getText().toString().trim();
                session.setLoggedin(true, displayUser, username);
                session.setRole(role);
                if(role.equals("3")) {
                    session.setDataPatient(displayUser, username);
                }
                //inserttoken();
```

```
            guardarToken();
            Intent intent = new Intent(LoginActivity.this, MainActivity.class);
            //intent.putExtra(Constantes.EXTRA_ID, documento);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            Toasty.success( this, mesagge, Toast.LENGTH_LONG).show();
            pDialog.dismiss();
            finish();
            break;
        case "2":
            Toasty.error( this, mesagge, Toast.LENGTH_LONG).show();
            pDialog.dismiss();
            break;
        case "3":
            Toasty.info( this, mesagge, Toast.LENGTH_LONG).show();
            pDialog.dismiss();
            break;
      }
    } catch (JSONException e) {
      e.printStackTrace();
    }

}
```

En el código anterior, en el método procesarRespuesta() primero se obtienen los elementos del objeto response JSON enviado por el servidor, seguidamente se crea un condicional de selección switch() y se le pasa el parámetro validate para 3 estados:

1. Cuando el valor de validate es 1, el inicio de sesión se hace correcto y se guardan los datos de usuario para recordar sesión, además se llama el método insertarToken() para guardar en el servidor el token de la aplicación que se usara para las notificaciones. Por último, se inicia la nueva actividad ActivityMain.java y se notifica al usuario mediante un elemento Toasty la bienvenida.

2. Cuando el valor de validate es 2, se notifica mediante un Toasty que la contraseña es incorrecta.

3. Por último, cuando el valor es 3, se notifica mediante un Toasty que el usuario no está registrado.

También cabe destacar que al inicializar la actividad se crearán referencias a todos los controles de la pantalla y se vinculará la actividad a su layout xml para establecer los vínculos y referencias necesarias para el correcto funcionamiento de la misma.

```
setContentView(R.layout.activity_login);
```

4.3.2.6 Lista exámenes

Esta funcionalidad contiene una clase que será la encargada de obtener la lista de exámenes de un usuario del servidor. Por lo cual se genera el método cargarAdaptador().

```
public void cargarAdaptador() {
  // Petición GET
  pDialog = new ProgressDialog(getActivity(), R.style.AppTheme_Dark_Dialog);
  pDialog.setMessage("Buscando...");
  pDialog.setIndeterminate(false);
  pDialog.setCancelable(false);
  pDialog.show();
  VolleySingleton.
        getInstance(getActivity()).
        addToRequestQueue(
              new JsonObjectRequest(
                    Request.Method.GET,
                    newURL,
                    null,
                    new Response.Listener<JSONObject>() {
                      @Override
                      public void onResponse(JSONObject response) {
                        // Procesar la respuesta Json
                        procesarRespuesta(response);
                      }
                    },
                    new Response.ErrorListener() {
                      @Override
                      public void onErrorResponse(VolleyError error) {
                        Log.d(TAG, "Error Volley: " + error.toString());
                        pDialog.dismiss();
                        msg("SIN CONEXIÓN");
                        }
                    } )
        );}
```

Esta funcionalidad es accedida desde el evento de pulsación del menú resultados, el cual llama al método cargarAdaptador(), este método será el encargado de realizar la petición HTTP GET.

En el método cargarAdaptador(), primero se crea un ProgressDialog, para indicar que el sistema está llevando a cabo una actividad que puede durar más tiempo de lo normal, luego se crea la petición HTTP, dentro de la cual encontramos todos los métodos que serán utilizados para controlarla, esta petición necesitará los siguientes parámetros:

1. El tipo de petición, para este caso Method.GET

2. La url a donde se dirige, definida en la clase Constantes, GET_LIST_EXAM
3. Debido a que es una petición GET no se le pasa un objeto, por tanto, se coloca null.
4. Un listener en caso de que la petición sea correcta, procesa la respuesta invocando el método procesarRespuesta().
5. Un listener en caso de que la petición devuelva un error si la petición solicitada al servidor ha devuelto algún tipo de error, ya sea de tiempo de espera o de otro tipo, este se mostrará al usuario, permitiéndole al usuario actualizar.

```
private void procesarRespuesta(JSONObject response) {
    try { // Obtener atributo "estado"
        String estado = response.getString("estate");
        switch (estado) {
            case "1": // EXITO
                // Obtener array "result" Json
                JSONArray mensaje = response.getJSONArray("result");
                // Parsear con Gson
                Exam[] exams = gson.fromJson(mensaje.toString(), Exam[].class);
                // Inicializar adaptador
                adapter = new ExamAdapter(Arrays.asList(exams), getActivity());
                // Setear adaptador a la lista
                pDialog.dismiss();
                getActivity().setResult(Activity.RESULT_OK);
                lista.setAdapter(adapter);
                mEmptyStateContainer.setVisibility(View.GONE);
                break;
            case "2": // FALLIDO
                mEmptyStateContainer.setVisibility(View.VISIBLE);
                String mensaje2 = response.getString("message");
                pDialog.dismiss();
                msg(mensaje2);
                break;
        }
    } catch (JSONException e) {
        Log.d(TAG, e.getMessage());
    }
}
```

En el código anterior, en el método procesarRespuesta() primero se obtiene el elemento state del objeto response JSON enviado por el servidor, seguidamente se crea un condicional de selección switch() y se le pasa el parámetro state para 2 estados:

1. Cuando el valor de state es 1, se obtiene el array del objeto response para luego mapearlo a la clase modelo Exam.java; una vez hecho eso se inicia el adaptador y se setea el adaptador a la lista.

2. Cuando el valor de state es 2, se notifica mediante un snakbar que no hay resultados y mostrando un botón para actualizar.

También cabe destacar que al inicializar la actividad se crearán referencias a todos los controles de la pantalla y se vinculará la actividad a su layout xml para establecer los vínculos y referencias necesarias para el correcto funcionamiento de la misma.

```
setContentView(R.layout.fragment_list_exam);
```

4.3.2.7 Resultado examen

Esta funcionalidad contiene una clase que será la encargada de obtener los detalles de un resultado de un examen del servidor. Por lo cual se genera el método cargarDatos()

```
public void cargarDatos() {
  // Petición GET
  String newURL = Constantes.GET_BY_ID_EXAM + orden;
  VolleySingleton. getInstance(this). addToRequestQueue(
          new JsonObjectRequest( Request.Method.GET,
              newURL, null,
              new Response.Listener<JSONObject>() {
                @Override
                public void onResponse(JSONObject response) {
                  // Procesar la respuesta Json
                  procesarRespuesta(response);
                  }
              },
              new Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                  Log.d(TAG, "Error Volley: " + error.toString());
        pDialog.dismiss();
                  Toasty.info( DetailExamActivity.this, "SIN RESPUESTA DEL SERVIDOR",
                      Toast.LENGTH_LONG).show();
                  finish();
                }
              } )
      );}
```

Esta funcionalidad es accedida desde el evento de pulsación de un ítem de la lista de resultados, el cual llama al método cargarDatos(), este método será el encargado de realizar la petición HTTP GET.

En el método cargarDatos(), primero se crea un ProgressDialog, para indicar que el sistema está llevando a cabo una actividad que puede durar más tiempo de lo normal, luego se crea la petición HTTP, dentro de la cual encontramos todos los métodos que serán utilizados para controlarla, esta petición necesitará los siguientes parámetros:

1. El tipo de petición, para este caso Method.GET
2. La url a donde se dirige, definida en la clase Constantes, GET_BY_ID_EXAM que se concatena con el id del resultado.
3. Debido a que es una petición GET no se le pasa un objeto, por tanto, se coloca null.
4. Un listener en caso de que la petición sea correcta, procesa la respuesta invocando el método procesarRespuesta().
5. Un listener en caso de que la petición devuelva un error si la petición solicitada al servidor ha devuelto algún tipo de error, ya sea de tiempo de espera o de otro tipo, se mostrará al usuario nuevamente la lista, permitiéndole al usuario poder realizar la acción.

```
private void procesarRespuesta(JSONObject response) {
  try {
    // Obtener atributo "estado"
    String estado = response.getString("estate");
    switch (estado) {
      case "1": // EXITO
          // Obtener objeto "exam"
        JSONObject object = response.getJSONObject("exam");
        //Parsear objeto
        Exam exam = gson.fromJson(object.toString(), Exam.class);
        issued.setText(exam.getIssued());
        effectivedatetime.setText(exam.getEffectiveDateTime());
        order.setText(" " + exam.getOrder().toString());
        displayPerformer.setText(exam.getDisplayPerformer());
        displaylab.setText(exam.getDisplayLab());

        if (!exam.getLabComments().equals("")) {
          labcomments.setText(exam.getLabComments());
        }
        if (!exam.getPerformerComments().equals("")) {
          performercomments.setText(exam.getPerformerComments());
        }
```

```
            pDialog.dismiss();
            break;
          case "2": // FALLIDO
            String mensaje2 = response.getString("message");
            pDialog.dismiss();
            Toasty.info( this,  mensaje2, Toast.LENGTH_LONG).show();
            finish();
            break;
      }

    } catch (JSONException e) {
      Log.d(TAG, e.getMessage());

    }

}
```

En el código anterior, en el método procesarRespuesta() primero se obtiene el elemento state del objeto response JSON enviado por el servidor, seguidamente se crea un condicional de selección switch() y se le pasa el parámetro state para 2 estados:

1. Cuando el valor de state es 1, se obtiene el objeto exam del response para luego mapearlo a la clase modelo Exam.java; una vez hecho eso se setean los valores a los TextView.

2. Cuando el valor de state es 2, se notifica mediante un Toasty que no hay respuesta del servidor o el resultado no existe.

También cabe destacar que al inicializar la actividad se crearán referencias a todos los controles de la pantalla y se vinculará la actividad a su layout xml para establecer los vínculos y referencias necesarias para el correcto funcionamiento de la misma.

```
setContentView(R.layout.  activity_detail_exam);
```

4.3.2.8 internacionalización

Para implementar esta función, se añade una carpeta diferente para cada copia del fichero strings.xml. De este modo, se quiere que la aplicación maneje dos idiomas español e inglés, por tanto, se tiene estos dos directorios:

- res/values (valores por defecto aplicación)

- res/values-en

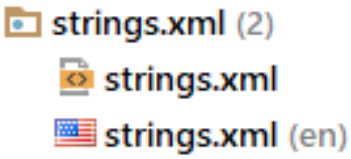


Figura 49. Recursos idiomas.

Cada fichero de texto, contendrá claves y valores:

Tabla 25. Claves y valores, español e ingles

/res/values/strings.xml	/res/values-en/strings.xml
... <string name="GLUCOSA"> GLUCOSA </string> <string name="CREATININA"> CREATININA </string> <string name="NITROGENO_UREICO">NITROGENO UREICO</string> <string name="SODIO">SODIO</string> <string name="POTASIO">POTASIO</string> <string name="CALCIO">CALCIO</string> <string name="CLORO">CLORO</string> <string name="DIOXIDO_CARBONO">DIOXIDO DE CARBONO</string> ...	... <string name="GLUCOSA">GLUCOSE</string> <string name="CREATININA"> CREATININE </string> <string name="NITROGENO_UREICO">UREIC NITROGEN</string> <string name="SODIO">SODIUM</string> <string name="POTASIO">POTASSIUM</string> <string name="CALCIO">CALCIUM</string> <string name="CLORO">CHLORIDE</string> <string name="DIOXIDO_CARBONO">CARBON DIOXID</string> ...

(Elaboración propia)

Ahora para utilizar el valor de las claves, en los XML se invocan de la siguiente manera.

```
<TextView
    android:text="@string/GLUCOSA"/>
```

Y mediante java se accede de la siguiente forma.

```
getString(R.string.GLUCOSA)
```

Para que el usuario pueda realizar el cambio de idioma cuando desee, para darle esta característica es necesario utilizar un Locale por defecto programáticamente para la aplicación en el caso de que dentro de ella se pueda definir el idioma a utilizar y obviar por tanto la configuración definida en el sistema. Para ello se ha implementado la siguiente función.

```
public void setLocale(String lang) {

    Locale myLocale = new Locale(lang);
    Resources res = getResources();
    DisplayMetrics dm = res.getDisplayMetrics();
    Configuration conf = res.getConfiguration();
    conf.locale = myLocale;
    res.updateConfiguration(conf, dm);
    onConfigurationChanged(conf);
}
@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);

}
```

4.3.3 Interfaz de usuario

En esta sección se muestran los aspectos más relevantes respecto a la interfaz de usuario final.

4.3.3.1 Aplicación web

Figura 50. Página de inicio.

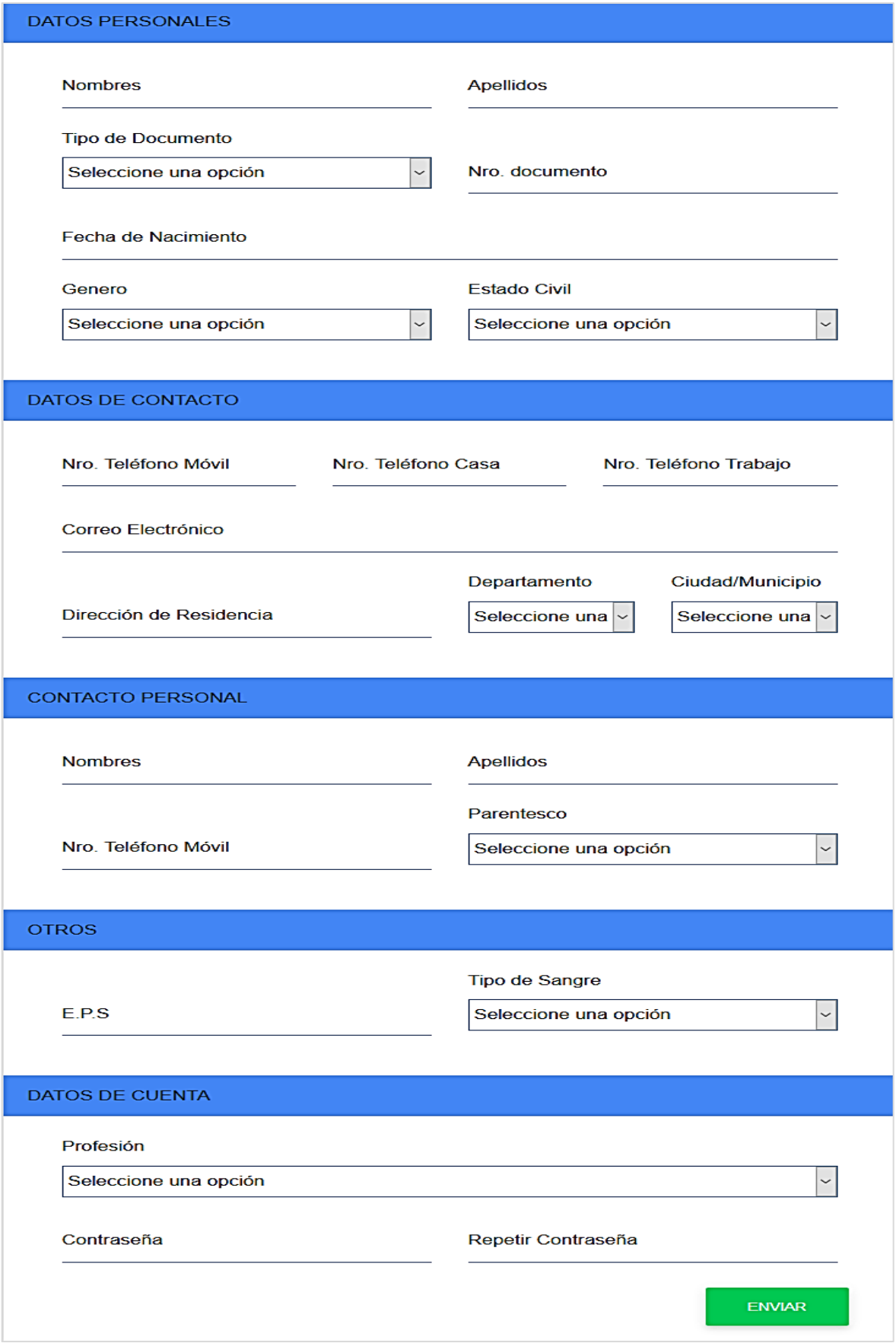


Figura 51. Formulario de registro de usuarios.

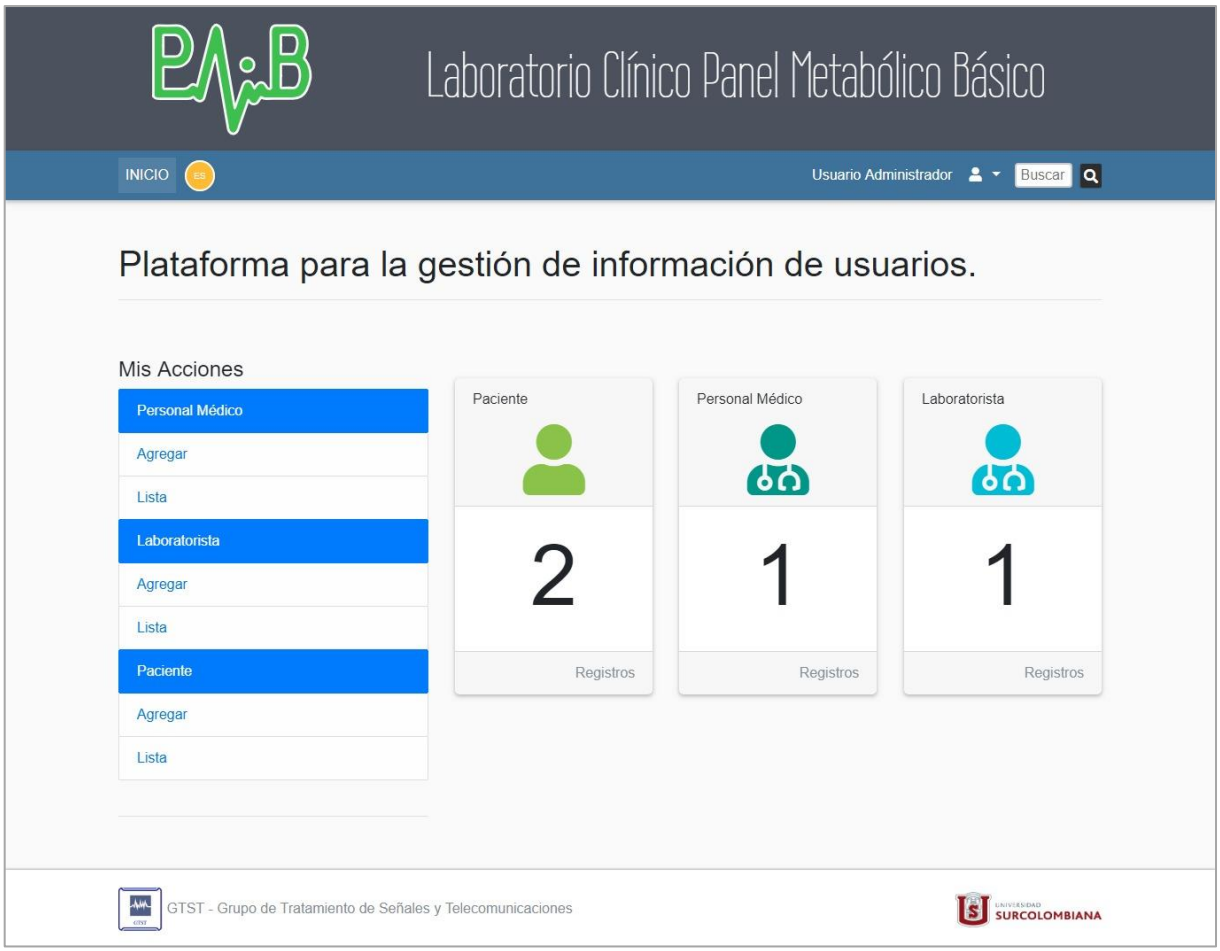


Figura 52. Página principal administrador.

Figura 53. Información de usuario.

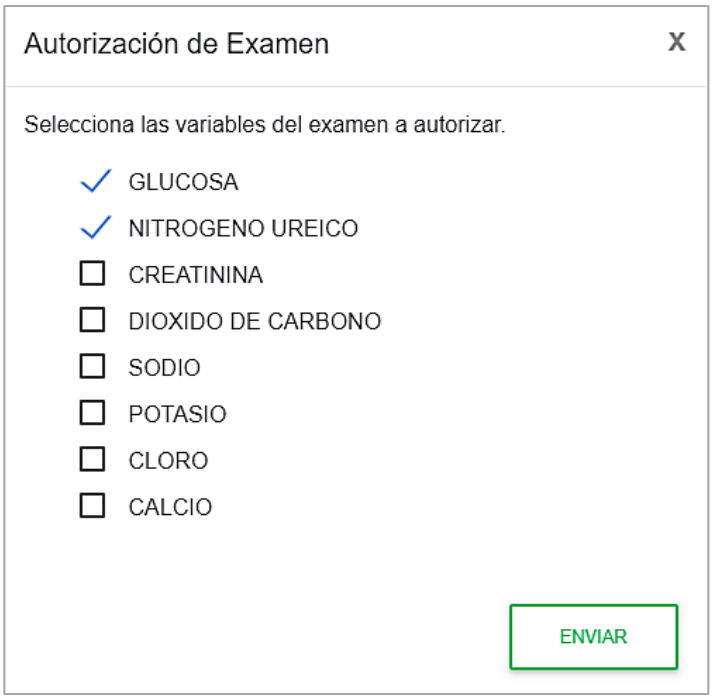


Figura 54. Autorizar examen.

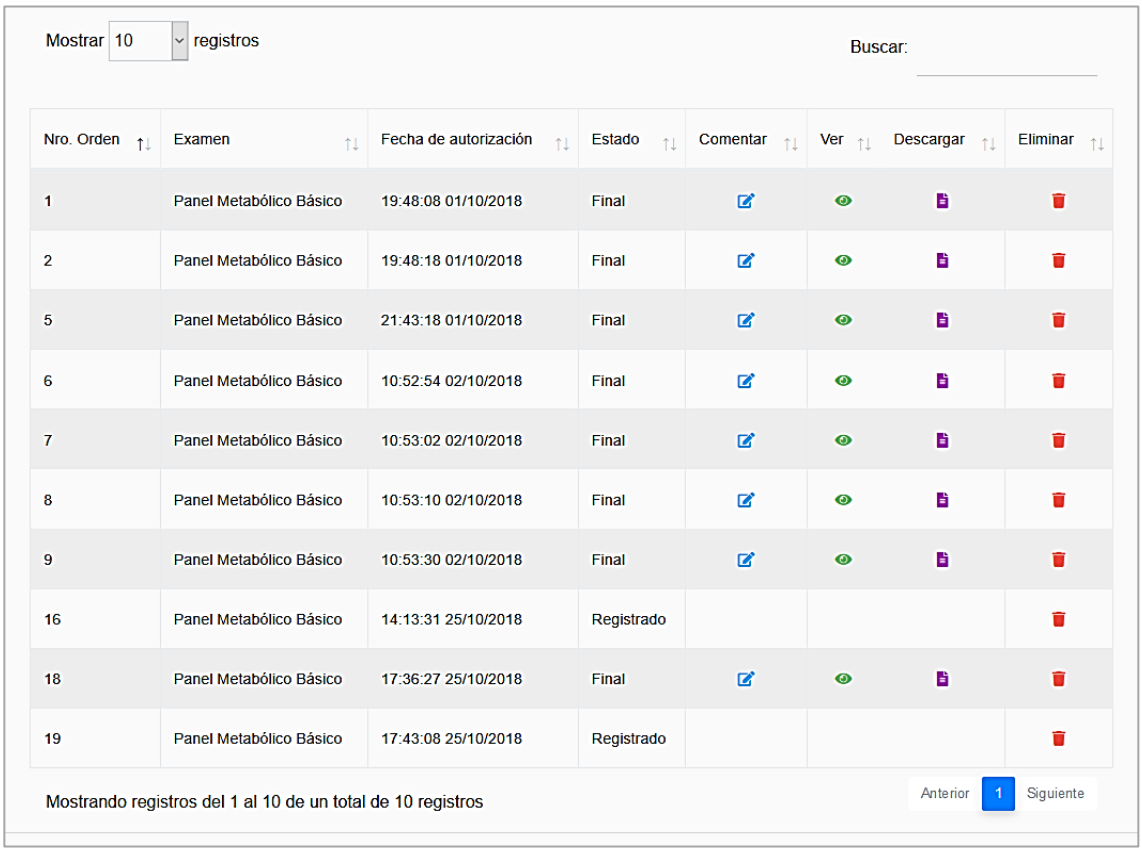


Figura 55. Lista de exámenes.

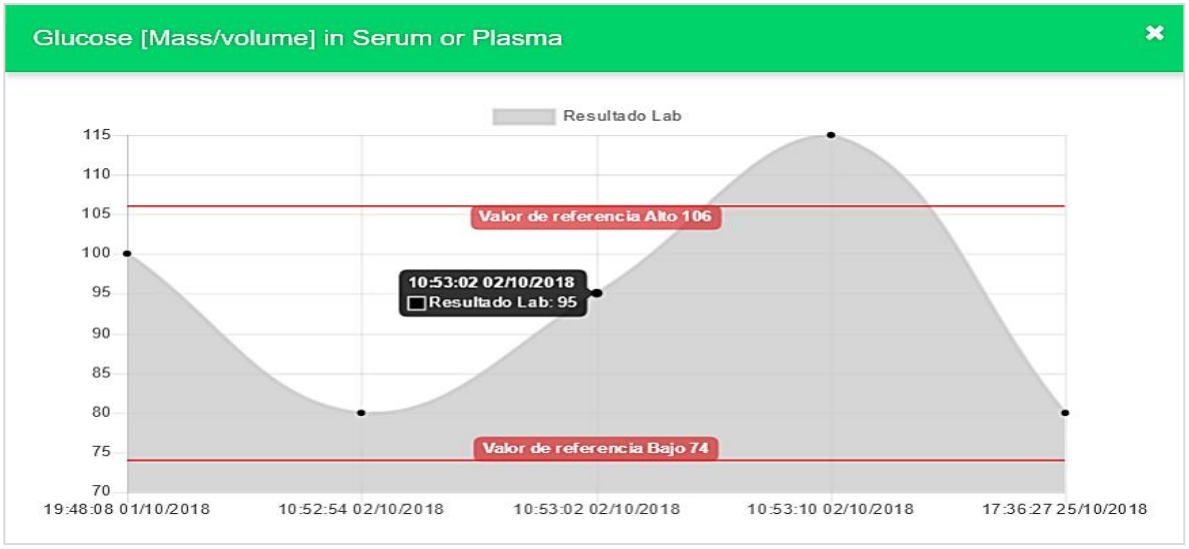


Figura 56. Ejemplo Gráfica datos Glucosa.

HOSPITAL DEPARTAMENTAL
LABORATORIO CLINICO

Paciente	: Usuario Paciente 1	**Orden**	: 2
Nº Documento	: 4	**Fecha**	: 14:36:10 14/07/2018
Edad	: 18 años, 1 meses y 19 dias	**Fecha de Impresion**	: 12:08:25 29/08/2018
Sexo	: genderM	**Medico**	: Usuario Personal 1
EPS	: asmet salud	**Bacteriologo**	: Usuario Laboratorista 1

ANALISIS	RESULTADO	UNIDADES	VALORES DE REFERENCIA
GLUCOSA	110	mg/dL	74 - 106
NITROGENO UREICO-BUN	24	mg/mL	10 - 20
CREATININA	2	mg/dL	0 - 1.5
SODIO	160	mmol/L	136 - 155
POTASIO	4	mmol/L	3.5 - 5.0
CALCIO	11	mg/dL	9 - 10.5
CLORO	100	mmol/L	98 - 106

COMENTARIOS LABORATORISTA:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

COMENTARIOS MEDICO:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

"La interpretacion de este y todo examen de laboratorio corresponde exclusivamente al Médico"

Figura 57. Reporte PDF.

4.3.3.2 Aplicación Android

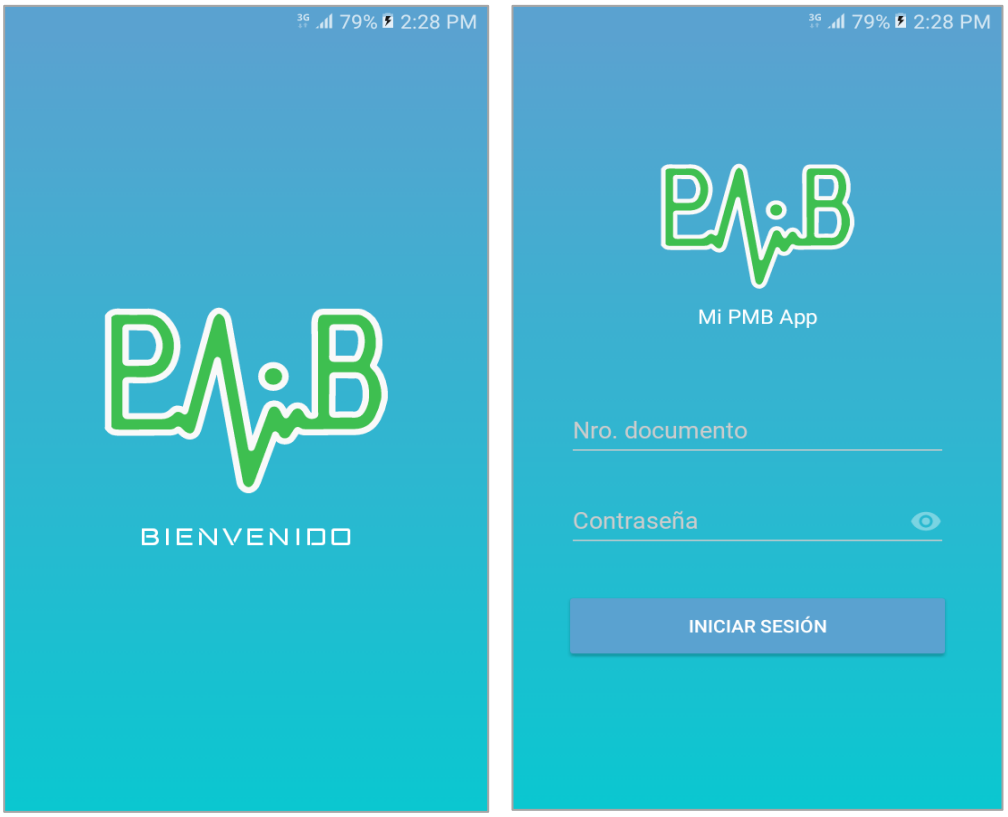


Figura 58. Interfaz de inicio.

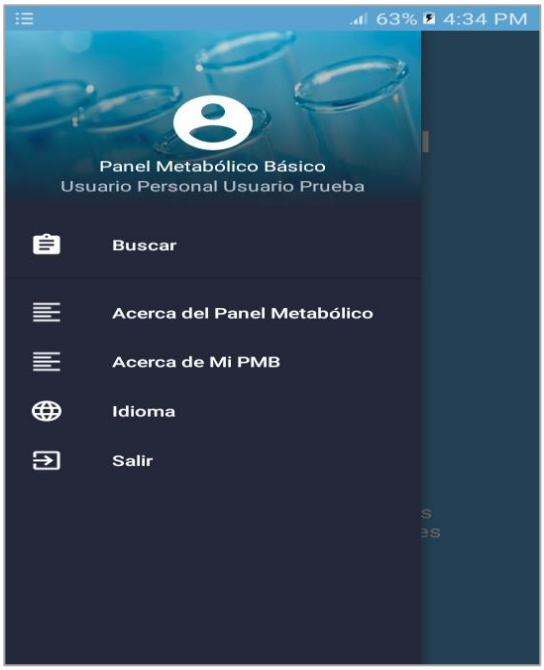


Figura 59. Menú principal.

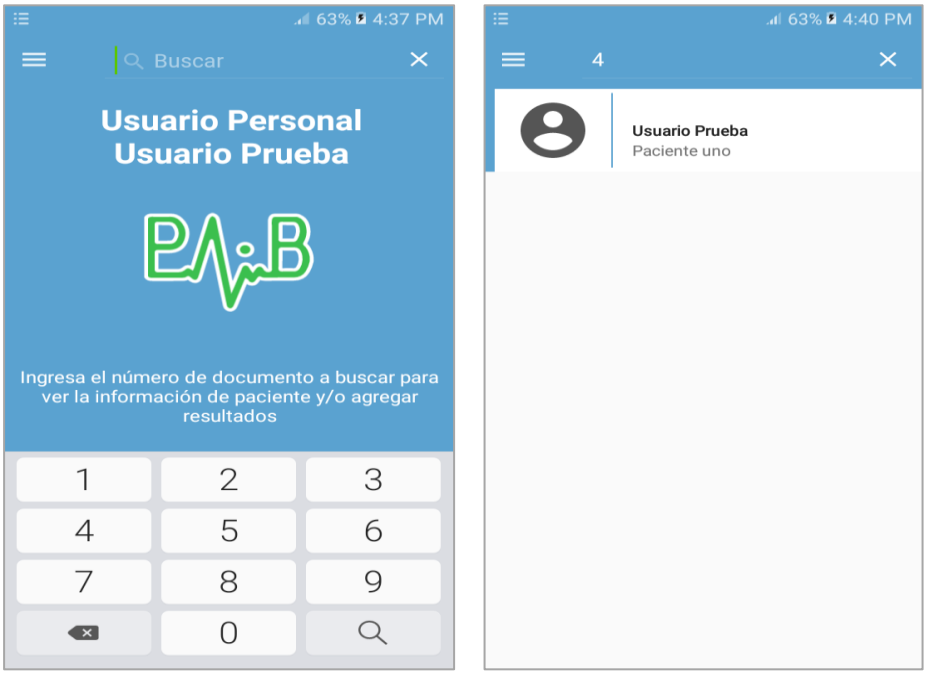


Figura 60. Interfaz de búsqueda.

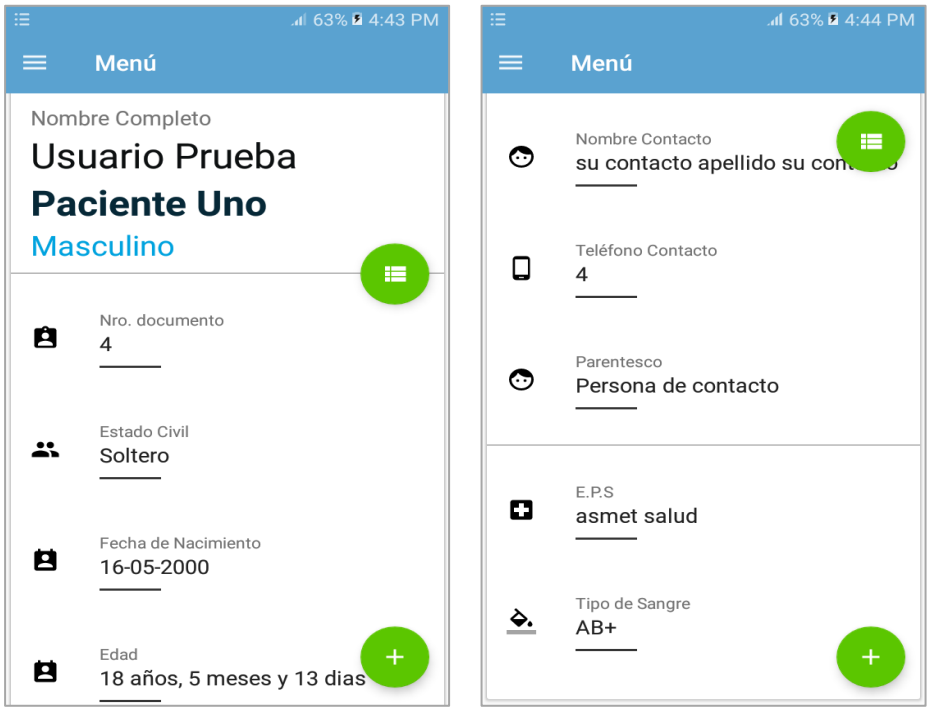


Figura 61. Información de usuario paciente.

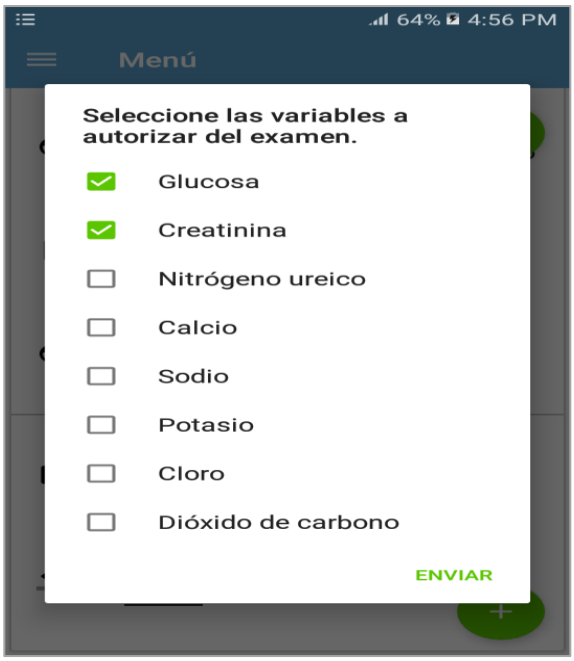


Figura 62. Autorización de examen.

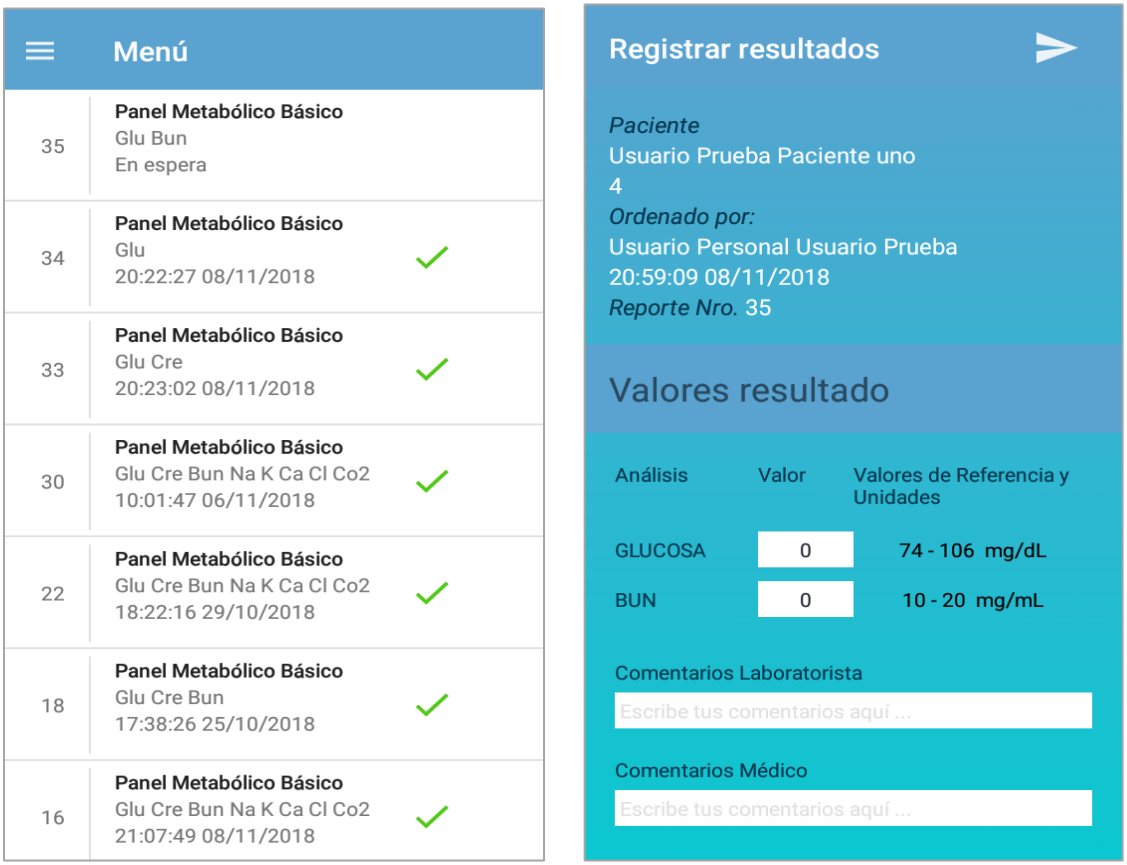


Figura 63. Lista de exámenes y formulario ingresar resultados.

Figura 64. Detalles de examen.

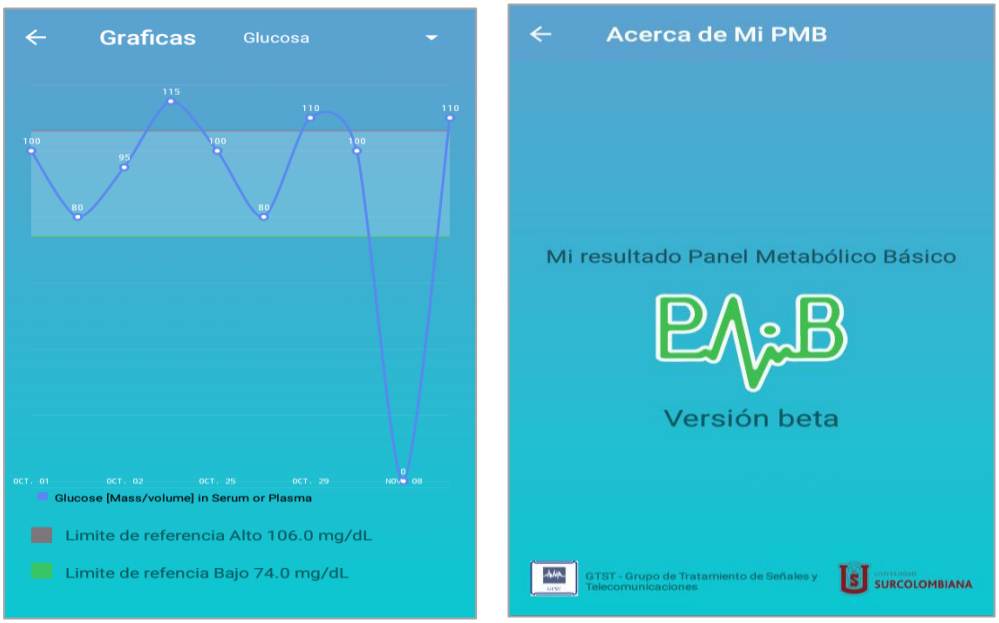


Figura 65. Ejemplo de gráfica datos glucosa y cerca de la aplicación.

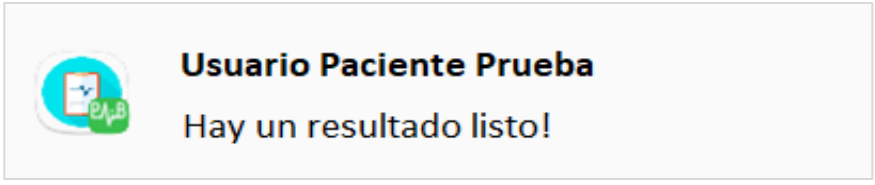


Figura 66. Notificación.

4.4 Evaluación

Para el desarrollo de este proyecto, se tuvieron en cuenta una serie de factores importantes al momento de realizar las pruebas correspondientes al sistema de información, que tienen como objetivo garantizar la calidad y cumplimiento de los requerimientos funcionales y no funcionales detallados anteriormente; se definieron una serie de criterios los cuales serán la base fundamental para el buen funcionamiento del sistema.

4.4.1 Funcionalidad

Las pruebas de funcionalidad permiten identificar errores en el procesamiento de datos, por lo que resultan de gran importancia en cualquier proyecto de desarrollo.

Las pruebas de caja negra son aquellas que examinan que las entradas apropiadas produzcan los resultados esperados, sin importar el funcionamiento interno del sistema sino el resultado final. Los resultados presentados a continuación son producto de una etapa de pruebas realizadas con las funcionalidades más importantes del sistema y de acciones correctivas ante las pruebas que resultaron fallidas en el proceso.

Las acciones correctivas o ajustes en la aplicación se llevaron a cabo de forma inmediata una vez hecha la prueba, lo cual quiere decir que se manejó varias pruebas hasta obtener el resultado definitivo y exitoso presentado a continuación:

Tabla 26. Pruebas de funcionalidad sesiones

Administración de seguridad			
Caso de prueba	Condiciones	Respuesta esperada del sistema	Resultado
Inicio de sesión	Ingreso de Usuario o contraseña incorrectos	Mensaje indicando que el usuario o la contraseña están erróneos	OK
	Ingreso de Usuario y contraseña correctos	Redirecciona a la sección dependiendo del tipo de usuario	OK

	Ingreso de URL sin una sesión iniciada en la aplicación.	Redirecciona a la página de inicio de sesión.	OK
	Ingreso de URL de otro tipo de usuario, con una sesión ya iniciada.	Redirecciona a la página de error 403 acceso denegado.	OK

(Elaboración propia)

Tabla 27. Pruebas de funcionalidad administración de usuarios

Administración de usuarios			
Caso de prueba	Condiciones	Respuesta esperada del sistema	Resultado
Registrar usuario	Iniciar sesión como administrador. Completar todos los campos de formulario. Formato de información ingresada en los campos correcta.	Mensaje indicando que el usuario se ha registrado correctamente.	OK
	Iniciar sesión como administrador. Información campos de formulario incompleta. Formato de información ingresada en los campos incorrecta.	Mensaje de error en los campos de texto donde se presenta.	OK

Editar usuario	Sesión iniciada. Todos los campos completos de formulario. Formato de información en los campos correcta.	Mensaje indicando que la información de usuario ha sido actualizada.	OK
	Sesión iniciada. Campos incompletos de formulario. Formato de información en los campos incorrecta.	Mensaje de error en los campos de texto donde se presenta.	OK
Consultar usuario	Sesión iniciada. Ingresar id de usuario correcto.	Muestra la información del usuario.	OK
	Sesión iniciada. Ingresar id de usuario incorrecto.	Muestra mensaje indicando que el usuario no existe	OK
Eliminar usuario	Sesión iniciada como administrador.	Muestra mensaje que el usuario se eliminó correctamente.	OK

(Elaboración propia)

Tabla 28. Pruebas de funcionalidad administración de exámenes

Administración de exámenes			
Caso de prueba	Condiciones	Respuesta esperada del sistema	Resultado
Autorizar examen	Iniciar sesión como personal médico. Usuario paciente debe existir.	Mensaje indicando que el examen se ha autorizado.	OK

Eliminar examen	Iniciar sesión como personal médico. Examen debe existir.	Mensaje indicando que el examen ha sido eliminado.	OK
Consultar examen	Iniciar sesión como personal médico, laboratorista o paciente. Examen debe existir.	Muestra la información del examen.	OK
Descargar reporte PDF	Iniciar sesión como personal médico, laboratorista o paciente. Examen debe existir.	Genera un archivo en formato PDF y lo descarga.	OK
Registrar resultado	Iniciar sesión como laboratorista. Examen debe existir.	Muestra mensaje indicando que el resultado se ha registrado correctamente.	OK
Ver gráficas	Iniciar sesión como paciente. Examen debe existir.	Muestra una gráfica de la variable seleccionada del examen.	OK

(Elaboración propia)

4.4.2 Seguridad

Para realizar estas pruebas se hizo uso de OWASP Testing Guide v3.0, esta implementa procedimientos para la realización de pruebas de intrusión en aplicaciones Web, y explica cómo realizar la comprobación de cada vulnerabilidad, para llevar acabo la prueba de seguridad se tuvieron en cuenta los siguientes Test como se muestran en la tabla siguiente y su resultado.

Tabla 29. Pruebas de vulnerabilidad

Test	Referencia	Nombre	Vulnerabilidad

Autenticación	OWASP AT 004	Pruebas de fuerza bruta	Credenciales de prueba
	OWASP AT 005	Saltarse sistema de autenticación	No presenta en las pruebas
	OWASP AT 006	Sistemas de recordatorio/ reset de contraseñas	No presenta en las pruebas
Autorización	OWASP AZ 003	Prueba de escalada de privilegios	No presenta en las pruebas
Gestión de sesiones	OWASP SM 003	Prueba de fijación de sesión	No presenta en las pruebas
	OWASP SM 005	Prueba para csrf	No presenta en las pruebas

(Elaboración propia)

4.4.3 Desempeño

Las pruebas de desempeño permitieron evaluar la eficiencia de la aplicación para realizar procesos de consulta e inserción de información con respecto al tiempo. El desempeño de una aplicación es determinante ya que es una característica fundamental que influye en la aceptación y funcionalidad de la misma.

Para el sistema las pruebas se realizaron midiendo el tiempo de respuesta de la base de datos al momento de realizar registro de datos y realizar consultas, mediante el uso de herramientas como OWASP ZAP. Aunque esta herramienta se usa para la identificación de vulnerabilidades, también ofrecen información relevante con respecto al desempeño de las aplicaciones ya que registra el tiempo de respuesta de cada consulta que automáticamente realizan sobre el sitio, es por ello que se han tomado como datos de referencia para la evaluación de desempeño del sistema. Una vez obtenida la información con OWASP ZAP los resultados se muestran en la siguiente tabla.

Tabla 30. Pruebas de desempeño

Consulta	Nro. registros	Método	RTT	Resultado

Iniciar sesión	1	POST	1.08 s	ok
Buscar usuario	17	POST	2.27 s	ok
Autorizar examen	1	POST	2.22 s	ok
Enviar notificación	1	GET	1.85 s	ok
Eliminar examen	1	GET	468 ms	ok
Ver examen	1	GET	2.74 s	ok
Actualizar perfil	1	POST	3.06 s	ok
Ver gráfica	8	GET	370 ms	ok
Cerrar sesión	1	POST	403 ms	ok

(Elaboración propia)

5 Conclusiones y trabajo futuro

5.1 Conclusiones

Como resultado de la solución de este proyecto se puede concluir que la herramienta creada ha aportado información muy útil de los resultados del examen panel metabólico básico y de usuarios relacionados con este; además de ser un aporte fundamental del proyecto ya que los usuarios que hacen parte de la entidad prestadora de salud que posee dicho sistema tienen acceso a su información desde cualquier lugar mediante dispositivos tecnológicos, generando calidad y buen servicio, el sistema asegurará que los resultados puedan manejarse de manera efectiva, al tiempo que los hará más accesibles tanto para los pacientes como para el personal médico y disminuye la insatisfacción de los usuarios en referencia a la solicitud de estos reduciendo el tiempo que conlleva este procedimiento.

Las aplicaciones se han desarrollado como un proyecto de software gratuito que puede ser usadas libremente con funcionalidades incorporadas para convertirse en una herramienta fundamental en la gestión de información de resultados y de usuarios.

La arquitectura del sistema está basada en un modelo cliente/servidor de tres capas. Con esto se consigue que el cliente pueda acceder al sistema con un simple navegador, además el sistema funciona para los navegadores más importantes del mercado como son Microsoft Edge, Google Chrome y Firefox Mozilla.

El sistema implementado cumple con los requerimientos necesarios para la correcta interacción sistema usuario, ya que permite a los usuarios con distintos roles ejecutar las funcionalidades de registrar, consultar, modificar y/o eliminar información de usuarios y de resultados de exámenes Panel Metabólico Básico desde cualquier dispositivo con conexión a internet y que cuente con un navegador web o la aplicación móvil.

Spring es un framework poderoso para la construcción de aplicaciones web empresariales. También se puede integrar fácilmente con otros frameworks para desarrollar aplicaciones eficientes y debido a su característica ligera, es fácil de usar.

Otro aspecto importante en el desarrollo de la aplicación web es la construcción bien estructurada, basada en el patrón de diseño MVC (Modelo Vista Controlador). Con esto se consiguió una aplicación de fácil mantenimiento y escalable. De esta manera se asegura alargar la vida de la aplicación.

Android al ser un sistema operativo libre, facilito la implementación de la aplicación móvil ya que cuenta con muchas herramientas de software libre como el IDE Android Studio y el SDK Android entre otros, también de alguna manera se facilitó el uso de estas ya que Android emplea lenguaje JAVA con el cual se estuvo familiarizado antes.

Se implemento el estándar FHIR concluyendo que combina las mejores características de las líneas HL7 v2, v3 y CDA, y aprovecha los últimos estándares web aplicando un enfoque estricto en la aplicabilidad y queda claro que la interoperabilidad semántica en el entorno sanitario es una solución para conseguir comunicar la información sanitaria entre diferentes sistemas sin perder su significado.

5.2 Líneas de trabajo futuro

Se considera que la futura línea de investigación es darle continuidad al proyecto implementando más funcionalidades al sistema, soporte en la actualización de dependencias, optimización de las aplicaciones, mejora en la GUI y permitiendo que ofrezca muchos más servicios a las entidades que lo posea.

Entre las nuevas funcionalidades, está un chat en tiempo real importante para estar en contacto con el personal médico o administradores del sistema y solucionar problemas que se presenten durante el uso del mismo.

Otra nueva funcionalidad que se puede añadir es la de realizar un análisis de los datos del examen panel metabólico básico para obtener estadísticas de las enfermedades que más aquejan a los usuarios con respecto al examen de todas las entidades prestadoras de salud que lo posean.

También un buzón de sugerencias, crear el perfil público para el personal médico y encuetas de satisfacción de la prestación de servicios tanto del sistema como del personal médico.

Claro está en decir que todas las mejoras al sistema, se tendrá en cuenta la opinión de los usuarios y el nivel de necesidad de las mismas para implementarlas.

6 Referencias

Bodenreider, O. (1 de Junio de 2004). *Acid Nucleid Researchs*. Recuperado el 10 de Abril de 2018, de The Unified Medical Language System (UMLS): integrating biomedical terminology: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC308795/

Donnelly, K. (2006). *Researchgate.* Recuperado el 10 de Abril de 2018, de SNOMED-CT: The advanced terminology and coding system for eHealth: https://www.researchgate.net/publication/6701249_SNOMED-CT_The_advanced_terminology_and_coding_system_for_eHealth

Egham. (15 de Febrero de 2017). *Gartner*. Recuperado el 10 de Abril de 2018, de Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016: https://www.gartner.com/newsroom/id/3609817

Fernández Puerto, F. J. (2003). *Facultad de Medicina UNAM*. Recuperado el 10 de Abril de 2018, de www.facmed.unam.mx/emc/computo/ssa/HIS/his.pdf

Guia Soluciones TIC. (2018). Recuperado el 4 de Febrero de 2019, de Guia Soluciones TIC: https://www.guiadesolucionestic.com/soluciones-verticales/sector-salud-seguridad-social-/administracion-y-gestion-de-laboratorios-clinicos/2815-baxlab-software-para-laboratorio-clinico-interfaces-e-integrado-con-redes-hospitalarias

hl7.org. (2011). *FHIR*. Recuperado el 10 de Abril de 2018, de https://www.hl7.org/fhir/overview.html

hl7.org. (2011). *FHIR*. Recuperado el 10 de Abril de 2018, de https://www.hl7.org/fhir/http.html

hl7.org. (2011). *FHIR*. Recuperado el 10 de Abril de 2018, de https://www.hl7.org/fhir/security.html

hl7.org. (2011). *FHIR*. Recuperado el 10 de Abril de 2018, de https://www.hl7.org/fhir/datatypes.html

IZAMORAR. (2017). *Actividades básicas de un Sistema de Información*. Recuperado el 10 de Abril de 2018, de https://izamorar.com/actividades-basicas-de-un-sistema-de-informacion/

IZAMORAR. (2017). *Componentes de un sistema de información.* Recuperado el 10 de Abril de 2018, de https://izamorar.com/componentes-de-un-sistema-de-informacion/

McDonald, C. J. (Abril de 2003). *Clinical Chemistry.* Obtenido de LOINC, a Universal Standard for Identifying Laboratory Observations: A 5-Year Update: http://clinchem.aaccjnls.org/content/49/4/624.long

Ortiz, F. (11 de junio de 2014). *Reporte Digital*. Recuperado el 10 de Abril de 2018, de https://reportedigital.com/iot/interoperabilidad-estandares-ehealth-ventajas-organizaciones-salud/

powerdata. (12 de Agosto de 2015). *PowerData*. Recuperado el 10 de Abril de 2018, de El valor de la gestión de datos: https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/bid/406547/tipos-y-funci-n-de-los-gestores-de-bases-de-datos

procesosdesoftware. (10 de Abril de 2018). *Procesos de software*. Obtenido de Metodologia RUP: https://procesosdesoftware.wikispaces.com/METODOLOGIA+RUP

Repositorio Institucional de la Universidad de Guayaquil. (24 de Agosto de 2015). Recuperado el 4 de Febrero de 2019, de Repositorio Institucional de la Universidad de Guayaquil: http://repositorio.ug.edu.ec/handle/redug/13049

Rodríguez, J. M. (2003). *SISTEMAS DE INFORMACIÓN: ASPECTOS TÉCNICOS Y LEGALES*. Recuperado el 10 de Abril de 2018, de http://www.ual.es/~jmrodri/sistemasdeinformacion.pdf

Romero, D. F. (2006). *INTRAMED*. Recuperado el 10 de Abril de 2018, de http://www.intramed.net/contenidover.asp?contenidoID=44061

Silberschatz, A. (2002). *Fundamentos de base de datos* (4 ed.). Madrid: McGraw-Hill.

Spring. (2017). *Spring docs.* Recuperado el 10 de Abril de 2018, de Spring Framework Overview: https://docs.spring.io/spring/docs/current/spring-framework-reference/overview.html#overview

Spring. (2017). *Spring docs.* Recuperado el 10 de Abril de 2018, de Web on Servlet Stack: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#spring-web

Spring. (2017). *Spring docs*. Recuperado el 10 de Abril de 2018, de Web on Servlet Stack: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html#mvc-servlet

Spring. (2017). *Spring docs*. Recuperado el 10 de Abril de 2018, de Spring Security Introduction: https://docs.spring.io/spring-security/site/docs/4.0.0.CI-SNAPSHOT/reference/html/introduction.html

Spring. (2017). *Spring docs*. Recuperado el 10 de Abril de 2018, de Data Access: https://docs.spring.io/spring/docs/current/spring-framework-reference/data-access.html#spring-data-tier

Spring. (2017). *Spring docs*. Recuperado el 10 de Abril de 2018, de Integration: https://docs.spring.io/spring/docs/current/spring-framework-reference/integration.html#spring-integration

Spring. (2017). *Spring docs*. Recuperado el 10 de Abril de 2018, de The IoC container: https://docs.spring.io/spring/docs/current/spring-framework-reference/core.html#context-introduction

SYSLAB Ingeniería. (2018). Recuperado el 2 de Febrero de 2019, de SYSLAB Ingeniería: http://www.syslab.cl/

Telefonica, F. (Julio de 2008). Nuevas arquitecturas tecnologicas. En *Las TIC en la Administración Local del futuro.* España: Ariel.

TutorialsPoint. (2018). *Tutorialspoint.* Recuperado el 10 de Abril de 2018, de Spring - MVC Framework: https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

Universidad de los Andes repositorio Institucional. (2014). Recuperado el 4 de Febrero de 2019, de Universidad de los Andes repositorio Institucional: https://repositorio.uniandes.edu.co/handle/1992/16871

Wikipedia. (2017). *Wikipedia*. Recuperado el 10 de Abril de 2018, de Sistema Operativo Móvil: https://es.wikipedia.org/wiki/Sistema_operativo_m%C3%B3vil

Wikipedia. (2017). *Wikipedia*. Recuperado el 10 de Abril de 2018, de Android: https://es.wikipedia.org/wiki/Android

Wikipedia. (2017). *Wikipedia*. Recuperado el 10 de Abril de 2018, de Servicio Web: https://es.wikipedia.org/wiki/Servicio_web

Wikipedia. (2018). *Wikipedia.* Recuperado el 10 de Abril de 2018, de Spring Framework: https://es.wikipedia.org/wiki/Spring_Framework