UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

Proyecto de Investigación previo a la obtención del título de Ingeniero en Sistemas y Computación.

TRABAJO DE TITULACIÓN

APLICACIÓN WEB SPA PARA LA GESTIÓN DE FICHAS MÉDICAS EN EL HOSPITAL UNIVERSITARIO ANDINO UTILIZANDO SERVICIOS REST.

Autor:

Alexander David Bonilla Adriano

Tutor:

MsC Milton Paúl López Ramos

Riobamba – Ecuador

Año 2021

DICTAMEN DE CONFORMIDAD DEL PROYECTO ESCRITO DE INVESTIGACIÓN

Facultad: Ingeniería
Carrera: Sistemas y Computación

1. DATOS INFORMATIVOS DOCENTE TUTOR Y MIEMBROS DEL TRIBUNAL

Tutor:	Milton Paul López Ramos	**Cédula:**	0602048035
Miembro tribunal:	Jorge Edwin Delgado Altamirano	**Cédula:**	0604059741
Miembro tribunal:	Wayner Xavier Bustamante Granda	**Cédula:**	1103951677

2. DATOS INFORMATIVOS DEL ESTUDIANTE

Apellidos: Bonilla Adriano
Nombres: Alexander David
C.I / Pasaporte: 0604059741
Título del Proyecto de Investigación: APLICACIÓN WEB SPA PARA LA GESTIÓN DE FICHAS MÉDICAS EN EL HOSPITAL UNIVERSITARIO ANDINO UTILIZANDO SERVICIOS REST.
Dominio Científico: Desarrollo territorial productivo y hábitat sustentable para mejorar la calidad de vida.
Línea de Investigación: Ingeniería Informática

3. CONFORMIDAD PROYECTO ESCRITO DE INVESTIGACIÓN

Aspectos	Conformidad Si/No	Observaciones
Titulo	SI	
Resumen	SI	
Introducción	SI	
Objetivos: general y específicos	SI	
Estado del arte relacionado a la temática de investigación	SI	
Metodología	SI	
Resultados y discusión	SI	
Conclusiones y recomendaciones	SI	
Referencias bibliográficas	SI	
Apéndice y anexos	SI	

Fundamentado en las observaciones realizadas y el contenido presentado, SI(✓) / NO( ) es favorable el dictamen del Proyecto escrito de Investigación, obteniendo una calificación de: 10 sobre 10 puntos.

Campus Norte | Av. Antonio José de Sucre, Km 1 ½ vía a Guano | Telefonos: (593-3) 3730880 - Ext: 1255 - 2212

Mgs. Milton Lopez
TUTOR

MsC. Jorge Delgado
MIEMBRO DEL TRIBUNAL

Mgs. Wayner Bustamante
MIEMBROS DEL TRIBUNAL

Campus Norte | Av. Antonio José de Sucre, Km 1 ½ vía a Guano | Teléfonos: (593-3) 3730880 - Ext.: 1255 - 2212

DEDICATORIA

El presente trabajo de investigación está dedicado a mis padres por haberme educado como la persona que soy al día de hoy, quienes con su ejemplo y apoyo incondicional me han secundado permitiéndome cosechar este logro.

A mi hermana, para demostrarle que todos los objetivos que nos propongamos a lo largo de nuestras vidas se las puede conseguir mediante esfuerzo y dedicación.

AGRADECIMIENTO

Agradezco a mis padres por ser los mentores de mi vida, por los valores que me han inculcado, por brindarme la oportunidad de seguir aprendiendo y poder servir a la sociedad.

Así mismo, deseo expresar mi más sincero agradecimiento a mi hermana, por ser mi compañera de barrabasadas y motivarme a ser una mejor versión de mi mismo.

Agradezco además a todos mis profesores por transmitirme sus conocimientos y ser el eje fundamental de mi formación profesional, en especial al MsC Milton López por ayudarme incondicionalmente durante el desarrollo de este trabajo de investigación.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo de investigación se lo realiza con el objetivo de analizar y describir los beneficios de los servicios REST (Representational state transfer) y las Single Page Aplications (SPA, aplicación de página única) a través del desarrollo e implementación de una aplicación web SPA para la gestión de fichas médicas en el Hospital Universitario Andino de Chimborazo (HUA).

Durante la fase de desarrollo de la aplicación web SPA y el servidor REST se utilizó la metodología ágil Scrum, metodología que permite generar entregables del aplicativo en periodos cortos de tiempo, además de herramientas de software libre para reducir los costos de desarrollo.

Finalmente se evaluó el rendimiento del servidor REST, utilizando métricas de desempeño basadas en la norma ISO/IEC 25010, considerando este estándar el adecuado para medir el rendimiento del servidor REST. El modelo utilizado se denomina, nivel de servicio basado en la métrica del tiempo de respuesta, en función a la cantidad de solicitudes realizadas al servidor REST.

Para analizar el nivel de rendimiento del servidor REST, se aplicó una carga simulada de transacciones de nivel medio-alto, con el objetivo de medir los tiempos de respuesta, las peticiones simultáneas en tiempo real fueron ejecutadas por la herramienta Apache JMeter configurada para que simule el comportamiento del usuario. De acuerdo con los resultados obtenidos en la investigación, se puede concluir que los servicios web tipo REST tienen un trabajo óptimo al combinarlos con una aplicación SPA.

Palabras Clave: Ficha médica, Aplicación web SPA, Node.js, Vue.js, LoopBack, Servicios web REST, Apache JMeter.

ABSTRACT

ABSTRACT

This research work carried out with the objective of analyzing and describing the benefits of REST (Representational state transfer) services and Single Page Applications (SPA) through the development and implementation of SPA web application for the management of medical records at Hospital Universitario Andino de Chimborazo (HUA).During the development phase of the SPA web application and the REST server, the agile Scrum methodology used, a methodology that allowed the generation of application deliverables in short periods of time, as well as free software tools to reduce development costs. Finally, the performance of the REST server evaluated, using performance metrics based on the ISO / IEC 25010 standard, considering this standard the appropriate one to measure the performance of the REST server. The model used called, service level based on the response time metric, based on the number of requests made to the REST server. To analyze the performance level of the REST server, a simulated load of medium-high level transactions applied, in order to measure response times, the simultaneous requests in real time executed by the Apache JMeter tool configured to simulate user behavior. According to the results obtained in the research, it can be concluded that REST-type web services have an optimal job when combined with SPA application.

Keywords: Medical file, SPA single-page application, Node.js, Vue.js, LoopBack, REST web services, Apache JMeter.

Reviewed by:
Mgs. Maritza Chávez Aguagallo
ENGLISH PROFESSOR
c.c. 0602232324

INTRODUCCIÓN

En la actualidad las entidades de salud deben gestionar una gran cantidad de información, ya sea referente a ingresos, egresos, traslados, historial clínico, etc. Por esta razón nacen los sistemas de Grupos Relacionados con el Diagnóstico (GRDs) que es una herramienta de gestión clínica, por ejemplo IR-GRD sistema que se está utilizando en Chile desde el año 2009 tanto en el ámbito público como privado, permitiendo de esta forma la clasificación de los casos que presentan los pacientes y sus peculiaridades (Zapata, 2018).

Los avances tecnológicos dentro de la web, llevan a múltiples compañías como Twitter, YouTube, Facebook, entre otras, a optar por la implementación de aplicaciones que consumen datos de una API REST para mejorar la experiencia del usuario. (Arsaute et al., 2018).

Como parte de esta investigación se desarrolló un sistema para la gestión y registro de fichas médicas basado en una arquitectura cliente-servidor; el servidor está construido bajo la arquitectura API REST y el cliente web es una aplicación SPA, la hipótesis que se desea contrastar establece que: una aplicación Web SPA brinda un rendimiento aceptable en conjunto con el servidor API REST en aplicaciones que gestionan las fichas médicas en el Hospital Universitario Andino de Chimborazo (HUA).

Los usuarios hoy en día están acostumbrados a la cercanía y comodidad de las aplicaciones nativas para computadora personal (PC) y móvil sobre diferentes sistemas operativos, demandan la misma cercanía y simplicidad en el contenido que ofrece una aplicación web, pero estás aún son inflexibles no pudiendo separarse de los navegadores que hacen de intermediarios entre la aplicación y el usuario (Lanza Ortega et al., 2019). Las aplicaciones web se han vuelto necesarias para que las empresas puedan ganar y fidelizar a sus clientes, para lo cual las aplicaciones Web SPA se han convertido en una gran opción brindando experiencia fluida al usuario, este tipo de aplicaciones otorgan una ventaja competitiva (Yan, 2019). En pocas

palabras SPA es un tipo de aplicación web donde todas las pantallas las presenta en la misma página, sin recargar el navegador (Alvarez, 2016).

La investigación cumple con la siguiente estructura: el capítulo I plantea la problemática, justificación y los objetivos del trabajo de investigación, en el capítulo II se plantea el marco teórico y se describe a las aplicaciones web SPA, servicios REST, metodología ágil scrum y las herramientas utilizadas para el desarrollo de la aplicación, el tercer capítulo presenta en detalle la metodología de investigación junto con la población y muestra para el análisis, finalmente el cuarto capítulo presenta los resultados obtenidos durante la elaboración de este trabajo de investigación.

CAPÍTULO 1

PLANTEAMIENTO DEL PROBLEMA

1.1. PROBLEMA

Actualmente, el Hospital Universitario Andino de Chimborazo (HUA) cuenta con un sistema informático para la automatización de sus procesos denominado Sigcenter, el cual se utiliza para administrar la información de los pacientes, el problema se presenta debido a que el sistema informático pertenece a una empresa externa, esto dificulta que el sistema se adapte a las nuevas necesidades y requerimientos de la institución, tampoco tiene el código fuente ni acceso a la administración de la base de datos.

La propuesta para solucionar la problemática del HUA es el desarrollo de una aplicación web SPA, se considera que este tipo de aplicación es la adecuada. Tomando en cuenta su contenido y manipulación constante de información, el usuario tendrá una mejor experiencia al utilizar y gestionar la información. Las aplicaciones SPA no necesitan de mucho ancho de banda reduciendo el tiempo de espera en la presentación de información (Shivakumar, 2020).

El presente trabajo de investigación es desarrollar una aplicación web SPA capaz de apoyar con el registro de la información de las fichas médicas de los pacientes del HUA.

1.2. JUSTIFICACIÓN

Los SPA integran lo mejor de dos mundos, las aplicaciones nativas y las aplicaciones web tradicionales, al desarrollar una SPA se obtiene la portabilidad y la funcionalidad multiplataforma de una aplicación web, así como la capacidad de respuesta de gestión del estado del cliente de la aplicación nativa móvil o de escritorio (Nygård et al., 2015). Las páginas destinadas a la gestión o administración de cualquier tipo de servicio, paneles de control tareas similares son muy adecuadas para las SPA. El resultado final es una aplicación web que se comporta similar a un software de escritorio (Alvarez, 2016).

Las aplicaciones SPA utilizan servicios REST porque permiten un acoplamiento flexible del cliente (front-end) y del servidor (back-end), REST es el encargado de proporcionar la autenticación, autorización, validación y manipulación de datos, mientras que la aplicación del cliente se encarga de recuperar los datos y enviarlos a la API, al estar separados el back-end del front-end se puede desarrollar webs dinámicas (Nygård et al., 2015).

Una SPA es un enfoque moderno centrado en el desarrollo de aplicaciones web, está siendo utilizada por empresas a nivel mundial como Google con Gmail, Twitter, Facebook, etc. (Yan, 2019).

Dado que no existe una aplicación propia, la aplicación SPA se desarrolla para permitir el almacenamiento y control de los datos del paciente, aprovechando las ventajas de las aplicaciones de página única, para que el Hospital Universitario HUA pueda mejorar la calidad de los servicios brindados a sus pacientes y gestione de manera eficiente la información.

1.3. OBJETIVOS

1.3.1. General

Implementar una aplicación web SPA para la gestión de fichas médicas en el Hospital Universitario Andino utilizando servicios REST.

1.3.2. Específicos

- Analizar las principales características y beneficios de los servicios REST y de las aplicaciones web SPA.
- Desarrollar una aplicación web SPA para la gestión de fichas médicas del Hospital Universitario Andino, integrando servicios REST.
- Evaluar el rendimiento de los servicios API REST mediante el uso de la herramienta Apache JMeter.

CAPÍTULO 2

MARCO TEÓRICO

2.1. Aplicación web

Una aplicación web es un software que realiza tareas a través de internet, para que una aplicación web funcione correctamente se necesita un servidor web, un servidor de aplicaciones y un servidor de base de datos. El servidor web procesa las solicitudes del cliente y el servidor de aplicaciones completa las tareas solicitadas. La base de datos es responsable de almacenar información. Para acceder a una aplicación web se utiliza un navegador web como Google Chrome, Mozilla Firefox entre otros (Rouse, 2019). Una aplicación web puede tener uno o varios clientes, cada uno con una interfaz de usuario diferente. Las aplicaciones web pueden ser accedidas desde diferentes dispositivos móviles o de escritorio, cada cliente puede tener una interfaz de usuario ligeramente diferente, pero cada cliente interactúa con la misma aplicación al conectarse al servidor. El cliente interactúa con el servidor realizando peticiones de Creación, Lectura, Actualización y Eliminación de datos (CRUD por sus siglas en ingles) (Fox, 2020).

2.2. Servicios web

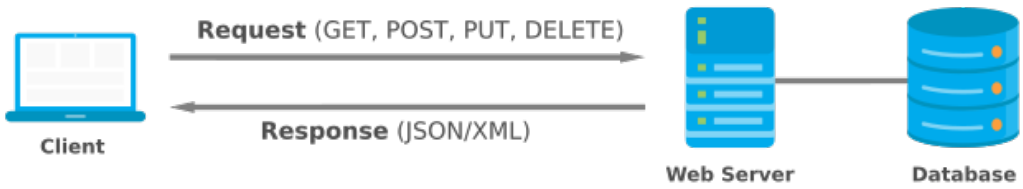
Los servicios web son responsables de publicar la funcionalidad de la aplicación al resto del mundo, los servicios web son agnósticos al lenguaje de programación con los que están desarrolladas las aplicaciones. Una aplicación desarrollada en Python puede comunicarse con una aplicación desarrollada con Java. Principalmente, existen dos tipos de servicios web, los servicios web SOAP (Simple Object Access Protocol) y los servicios REST (Representational State Transfer). El SOAP es un protocolo basado en XML, en donde las solicitudes al servidor y las respuestas del servidor están en formato XML. Por otro lado, REST representa el estilo arquitectónico de los servicios web y admite la transmisión de datos en varios formatos, como

HTML, XML, JSON. En comparación con los servicios REST, los servicios SOAP son más lentos porque requieren mayor ancho de banda (Ramidi, 2017).

2.3. Servicios REST

Los servicios REST son un tipo de arquitectura para el desarrollo de servicios web orientados a recursos. REST transmite los datos sobre el protocolo estandarizado HTTP, pero no es el único protocolo con el cual está asociado. Debido al desarrollo de los servicios REST, no es necesario realizar programación exclusiva para cada plataforma, ni estar restringido por lenguaje, hardware y ejecución, pero es suficiente con exponer los servicios web para uso del cliente (Haro et al., 2019).

Figura 1: Interacción de los servicios REST



Elaborado por: Alexander Bonilla

Los servicios REST utilizan un conjunto de verbos idempotentes que permiten realizar una determinada acción varias veces, estos verbos se los describe en la **Tabla 1**, por tanto, la arquitectura REST tiene como objetivo céntrese más en interactuar con los recursos. Los servicios web REST se los utiliza como protocolo de intercambio y manipulación de datos(Arsaute et al., 2018).

Tabla 1: Verbos para las peticiones HTTP

Verbo	Descripción	Seguro	Idempotente
GET	Solicitar un recurso.	Si	Si
HEAD	Recupere información sobre recursos. Efectivamente igual que la solicitud GET.	Si	Si
PUT	Actualiza un recurso existente.	No	Si
DELETE	Elimina un recurso existente.	No	Si
POST	Agregar un nuevo recurso o solicitar una acción en una solicitud existente	No	No
OPTIONS	Consultar los verbos HTTP admitidos por un recurso.	Si	Si
PATCH	Actualiza una parte de un recurso existente.	No	Si

Elaborado por: (Soni and Ranga, 2019)

Ventajas

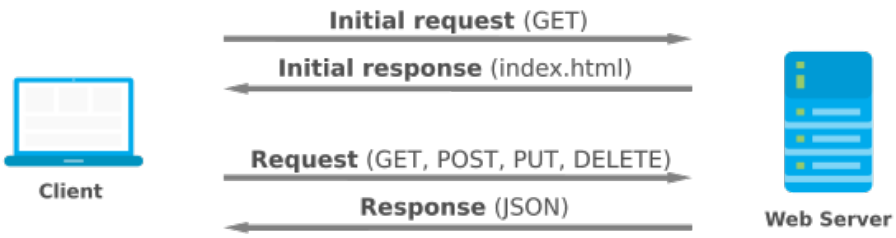
- Separa el cliente del servidor
- Permite proporcionar aplicaciones fiables, escalables y flexibles que son fáciles de desarrollar y mantener.
- Asigna las funcionalidades CRUD a diferentes endpoints a través de diferentes verbos HTTP.
- Tiene seguridad basada en HTTP/HTTPS y permite implementar otras tecnologías de seguridad según los requisitos del proyecto.
- Permite enviar respuestas al cliente en un formato ligero como JSON, acortando así el tiempo de respuesta a las acciones del usuario.
- Puede ser consumido desde aplicaciones web, de escritorio y móviles, independientemente del lenguaje de programación que utilicen.

2.4. Aplicación SPA

Una Single Page Aplication es una aplicación basada en la web que utiliza solo una página HTML como la columna vertebral de todas las sub páginas de la aplicación, no necesita actualizarse durante su utilización, una vez cargado el fichero HTML se descarga todo el código fuente necesario para la funcionalidad de la aplicación (Solovei et al., 2018). Para mejorar la interacción del usuario con la aplicación se implementa JavaScript y CSS. A diferencia de los sitios web tradicionales que se cargan en el servidor, las aplicaciones SPA entrega esa responsabilidad al navegador (Stepniak and Nowak, 2017).

Las SPA realizan la petición del marcado (HTML) y los datos (JSON) de forma independiente, una vez cargado el fichero HTML y los datos, el navegador es responsable de renderizar y visualizar la página web. Una SPA interactúa con el servidor web únicamente con la transferencia de datos, lo que reduce la sobrecarga en la red, permitiendo a la aplicación ser mucho más rápida en comparación a las aplicaciones web tradicionales (Solovei et al., 2018).

Figura 2: Interacción una aplicación SPA



Elaborado por: Alexander Bonilla

Una SPA puede estar compuesta por una gran cantidad de scripts y estilos, pero no se descargan todos los archivos con la petición inicial, los recursos se solicitan al servidor dependiendo a la necesidad de la sub página, posterior se almacena en la caché del navegador para no volver a solicitarlos (Solovei et al., 2018).

Ventajas

- Existen varios frameworks que permiten el desarrollo, despliegue y depuración de forma sencilla.
- Al cargar todo el contenido en una sola página, el contenido se puede mostrar de una

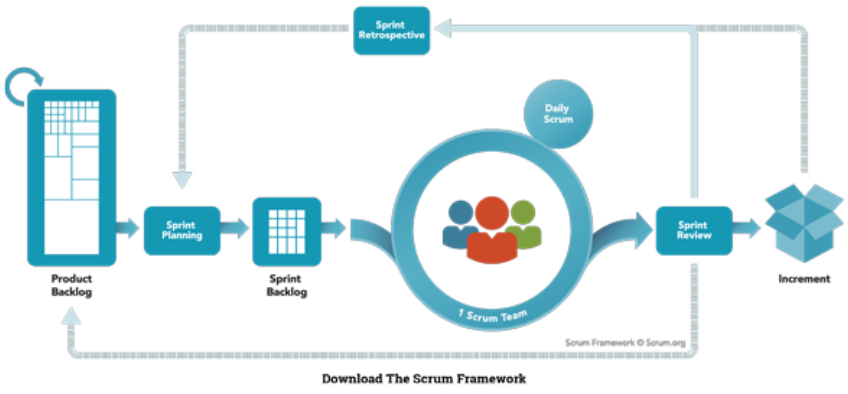
manera simple, elegante y veloz.

- Una vez que la aplicación realiza la solicitud inicial y muestra el contenido en el navegador, solo enviará y recibirá datos, reduciendo así el tiempo de respuesta a las acciones del usuario.
- El navegador es el encargado de ejecutar la lógica, lo que permite que la interacción del usuario sea más rápida.
- Las aplicaciones SPA aceleran el proceso de carga al minimizar el código JavaScript, eliminar las reglas CSS no utilizadas y fusionar recursos.

2.5. Metodología ágil Scrum

Actualmente, para brindar una solución informática con estándares de calidad se opta por metodologías de desarrollo ágil, en donde el proyecto de desarrollo es subdividido en proyectos mucho más pequeños (Montero et al., 2018). Scrum es una metodología ágil que permite la planificación y proceso de gestión durante el desarrollo de los productos de software. El uso del método Scrum en cualquier proyecto tiene ventajas como la adaptabilidad, que permite que el proyecto incorpore cambios, creando así un entorno abierto para la retroalimentación continua, mediante la división del proyecto en tareas más pequeñas para generar entregables que se pueden llevar a cabo en periodos cortos de tiempo, junto con la evaluación del progreso para mejorar gradualmente. En el proyecto Scrum, el problema se resuelve más rápido (Ramírez et al., 2019).

Figura 3: Ciclo de vida de Scrum



Elaborado por: scrum.org

2.6. Node.js

Node.js está diseñado para utilizarse como entorno de ejecución para JavaScript asincrónico controlado por eventos, también fue diseñado para crear aplicaciones de red escalables (Young et al., 2017). Node.js puede procesar muchas conexiones simultáneamente, para cada conexión se activa un callback (devolución de llamada), pero si no hay trabajo por hacer Node.js entrará en suspensión (Haro et al., 2019).

2.7. TypeScript

TypeScript es un lenguaje de programación que permite extender la funcionalidad y mejorar las falencias que JavaScript posee por defecto, además, permite crear un código más limpio y escalable (Cherny, 2019). TypeScript puede utilizar bibliotecas JavaScript mediante anotaciones opcionales que permite la compatibilidad con el entorno de desarrollo integrado (IDE) (Kristensen and Møller, 2017). Node.js no es capas de ejecutar código TypeScript por lo que se debe utilizar un transcompilador para convertirlo en código JavaScript (Cherny, 2019).

2.8. LoopBack

Su sitio web oficial lo define como un framework Node.js y TypeScript altamente escalable basado en Expressjs. LoopBack permite crear rápidamente APIs y microservicios compuestos por sistemas back-end como bases de datos y servicios SOAP o REST (StrongLoop, 2020).

2.9. Vue.js

Vue.js es un ecosistema enfocado en la creación de aplicaciones del lado del cliente (front-end) bastante robustas, se puede extender el ecosistema dependiendo de las necesidades que la aplicación tenga (Macrae, 2018). Vue.js también se describe como un framework que puede ser utilizado para el desarrollo de pequeños proyectos o para la creación de aplicaciones complejas como una SPA en toda regla (Wohlgethan, 2018). Utiliza principalmente JavaScript, sin embargo a partir de la versión 2.5.0 también posee soporte a TypeScript (Wohlgethan, 2018). Actualmente empresas de reconocimiento mundial como Alibaba, GitLab están utilizando Vue.js para la creación de sus aplicaciones web (Daityari, 2020). Uno de los desafíos más grandes al construir una aplicación SPA es reducir los archivos HTML, JavaScript y CSS para que el tiempo de carga

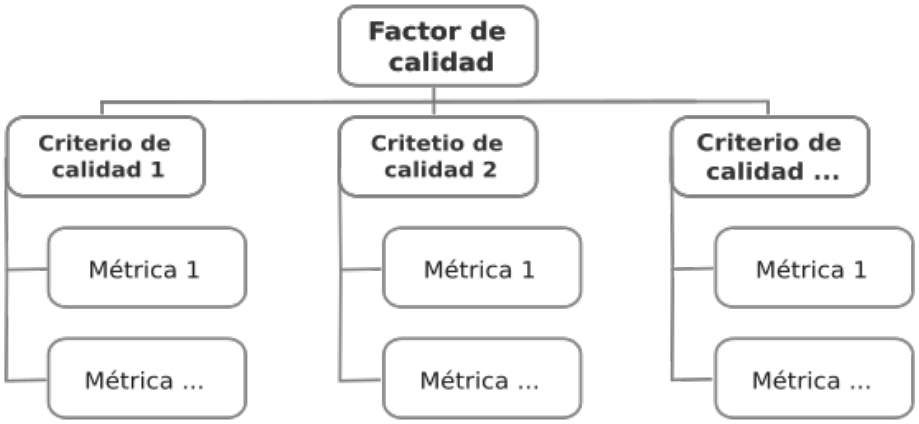
no se vea afectado. (Kaluža et al., 2018). Por lo tanto Vue.js minimiza el código JavaScript, elimina las reglas CSS no utilizadas y fusiona recursos, permitiendo que la aplicación se cargue rápido (Kaluža et al., 2018).

2.10. Calidad de software

Las herramientas de software actualmente son utilizadas por las organizaciones para la optimización de sus procesos, por lo que estas herramientas deben tener múltiples estándares para asegurar su calidad. La calidad de software hace referencia al grado de desempeño que el software debe cumplir durante su ciclo de vida, estos estándares garantizan la confiabilidad del software. (Callejas et al., 2017).

Con la finalidad de garantizar la calidad de una herramienta software se debe implementar un modelo o estándar con medidas de calidad establecidas que permita gestionar los atributos del software durante su construcción (Callejas et al., 2017). La estructura general de los modelos de calidad del software se muestra en la **Figura 4**.

Figura 4: Estructura de la calidad de software



Elaborado por: (Callejas et al., 2017)

2.11. Parámetros de medición de rendimiento

El hecho es que los parámetros de medición indican la eficiencia y disponibilidad del sistema o proceso. Al mismo tiempo, mediante el uso de valores medidos, se pueden garantizar la salud del servidor de manera eficaz (Jader et al., 2019). A continuación se describe las métricas para la evaluación del rendimiento.

Rendimiento

El rendimiento es una de las métricas utilizadas para evaluar los servicios web en una red, se trata del número de peticiones que el servidor puede procesar en un periodo de tiempo (Chitra and Satapathy, 2017). Por lo general, se usa para evaluar la cantidad de carga de un servidor (Jader et al., 2019).

Tiempo de respuesta

También conocido como tiempo de latencia (Saeid and Ali Yahiya, 2018). Es el tiempo en milisegundos que le toma al servidor para procesar una petición y responder al cliente (Saeid and Ali Yahiya, 2018) (Chitra and Satapathy, 2017). Es la sumatoria de tiempos que al servidor le puede tomar en ejecutar las operaciones asincrónicas como la consulta a una base de datos (Jader et al., 2019).

Peticiones por segundo

Se refiere al número de peticiones que el servidor es capaz de procesar en un segundo. Cuanto mayor sea el rendimiento, más solicitudes se responderán en un segundo (Jader et al., 2019).

2.12. ISO/IEC 25010

Con la finalidad de crear herramientas de software que cumplan altos estándares de calidad se utiliza el modelo de calidad ISO llamadas también SQuaRE. Dentro de este modelo se encuentra la norma ISO/IEC 25010 que es la selección de la norma ISO/IEC 9126-1:2001. Este modelo de calidad consta de ocho características como se muestra en la **Figura 5** (Sánchez, 2016). La norma ISO/IEC 25010 propone evaluar los principales niveles de calidad relacionados, es decir, medir la calidad interna, calidad externa y calidad en uso como se muestra en la **Tabla 2** (Basson et al., 2016).

2.13. Métricas de la ISO/IEC 25010

La eficiencia de un software está relacionado directamente con el rendimiento temporal del equipo, incluyendo el uso de recursos y la capacidad o límites máximos de funcionamiento (Fierro et al., 2019).

Figura 5: Modelo de calidad definido por la ISO/IEC 25010

Elaborado por: ISO/IEC 25010

Tabla 2: Métricas ISO/IEC 25010

Externas	▪ Conectividad con Sistemas externos. ▪ Capacidad de intercambiar datos.
Internas	▪ Nivel de servicio. Tiempo de respuesta. Tiempo exigido con base en número de peticiones

Elaborado por: Alexander Bonilla

2.13.1. Tiempo de respuesta

Esta métrica se la aplica con el propósito de estimar el tiempo en completar una tarea desde que el cliente envía la petición hasta que el servidor devuelve una respuesta (Fierro et al., 2019). El tiempo de respuesta óptimo debe estar por debajo de los 200 ms (Google, 2018).

Dónde:

X : Tiempo de respuesta.

A : Tiempo de envío de petición.

B : Tiempo en recibir la primera respuesta.

Formula:

$$X = B - A$$

2.13.2. Rendimiento

Esta métrica se la aplica con el propósito de contabilizar las tareas procesadas durante una unidad de tiempo (Paz et al., 2017). El rendimiento se lo calcula tomando en cuenta el número de tareas completadas y el tiempo que se toma para completar todas las tareas (Fierro et al., 2019).

Dónde:

X : Rendimiento.
A : Número de tareas completadas.
T : Intervalo de tiempo.

Formula:

$$X = \frac{A}{T}$$

2.13.3. Rendimiento de servicio

El rendimiento del nivel de servicio tiene como objetivo medir el cumplimiento de las tareas requeridas dentro de un cierto período de tiempo. (Moreno, 2007).

Dónde:

w : Rendimiento de servicio.
T^w : Número de tareas completadas.
T_0 : Tiempo requerido.

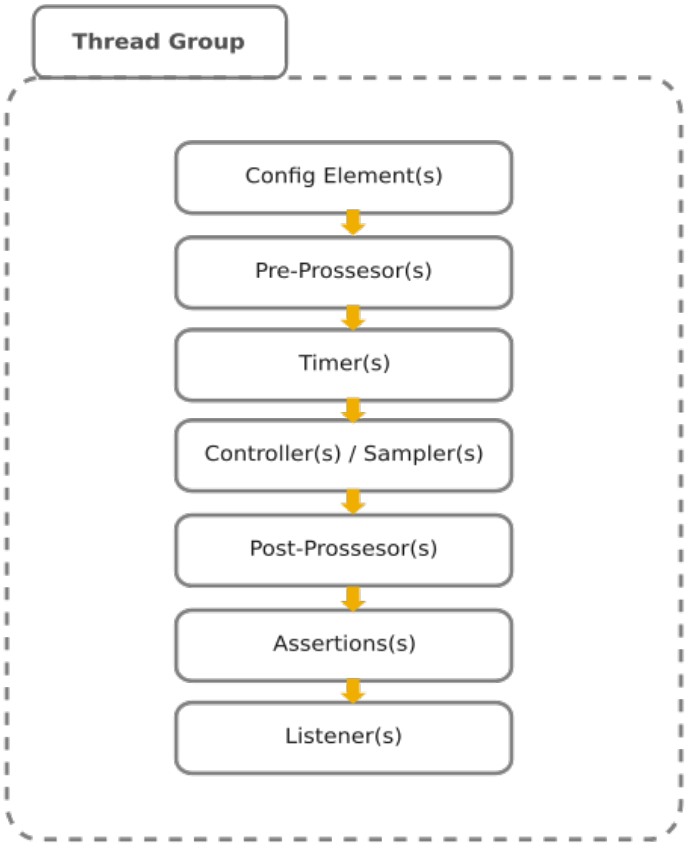
Formula:

$$w = \frac{Nro(T^w < T_0 \ ; \ en \ periodo \ t)}{Nro(Peticiones^w \ ; \ en \ periodo \ t)} \tag{2.1}$$

2.14. Herramienta de testing

La herramienta de testing destinada a medir y evaluar el rendimiento del servidor web es Apache JMeter. Esta herramienta se utiliza para testear cualquier tipo de recurso al aplicar una carga con el propósito de analizar todo el rendimiento bajo diferentes tipos de cargas. (Chitra and Satapathy, 2017) El script de prueba consta de una estructura jerárquica y componentes secuenciales dispuestos en un árbol (Matam and Jain, 2017).

Figura 6: Secuencia de ejecución Apache JMeter



Elaborado por: (Matam and Jain, 2017)

Cada componente cumple con un propósito dentro del árbol de ejecución, en la **Tabla 3** se describe la funcionalidad de cada componente (Matam and Jain, 2017).

Tabla 3: Componentes de una prueba en Apache JMeter

Test Plan	En el componente raíz del script de prueba.
Thread Group	Es un elemento hijo de un plan de prueba, permite el comportamiento de los elementos del plan de pruebas.
Controller	Permite aplicar una lógica para el flujo de ejecución de la siguiente petición.
Sampler	Permite enviar peticiones al servidor.
Listener	Escucha la respuesta emitida por el servidor.
Timer	Inserta un retraso entre peticiones para simular a un usuario.
Assertions	Verifica la respuesta del servidor.
Config Element	Permite configurar propiedades y variables.
Pre-Processors	Modifica los valores de una petición antes de ser enviada al servidor.
Post-Processors	Modifica los valores de una respuesta después de que el servidor envíe la respuesta.

Elaborado por: (Matam and Jain, 2017)

CAPÍTULO 3

METODOLOGÍA

3.1. METODOLOGÍA

Durante la presente investigación se consideró desarrollar una aplicación web SPA mediante la metodología ágil Scrum, que permite la planificación y gestión del desarrollo del aplicativo mediante sprints basados en los requerimientos que fueron solicitados por el área de Tics del Hospital Universitario HUA, el analista de Tics cumple con el rol de product owner dentro de dicha metodología. Posterior al desarrollo y ejecución de la de las pruebas de rendimiento se aplicó un enfoque cuantitativo que permitió el análisis e interpretación del rendimiento del servidor REST en aplicación para la gestión de fichas médicas.

Planificación del sprint

La planificación se realiza a partir de cada módulo de la aplicación SPA y el servidor REST dentro de un sprint como se detalla en la **Tabla 5**, tomando en cuenta los requerimientos funcionales y no funcionales expuestos por el HUA.

Tabla 4: Requerimientos funcionales

Requerimiento	**Descripción**
Autentificación de Usuarios	El usuario se identificará en el sistema web el cual mostrará los módulos correspondientes al tipo de usuario identificado.
Registro de Usuarios	Se crea un registro de un usuario (administrativo, médico, enfermería, analista) con sus datos personales según lo solicite el sistema para obtener acceso a los diferentes módulos, según corresponda al tipo de usuario creado.
Registro del paciente	El usuario según su rol (analista) realizará el registro de un paciente con su información personal.
Registro de Especialidades	Se crea un registro de especialidades médicas en el sistema web que pueden ser atendidos en el centro de salud.
Registro de signos vitales	El usuario con su rol (enfermería) debe tomar los signos vitales de un paciente previo a la revisión por el profesional médico.
Registro de la ficha médica	El usuario según su rol (médico, enfermería) realizará el registro de una ficha médica mediante un formulario solicitado por el sistema.
Consultar Información	El usuario tendrá acceso mediante el sistema web al módulo de vista de fichas médicas atendidas correspondientes a cada paciente
Gestionar Reportes	Permite al usuario (médico) exportar reportes de la ficha médica de un paciente en formato pdf.

Elaborado por: Alexander Bonilla

Etapa de desarrollo

En esta etapa se elabora y configura el proyecto, posteriormente se fue agregando cada módulo de la aplicación descrito y planificado en cada sprint de la metodología scrum, cuando se presentaron obstáculos se procedió a estimar el tiempo y prolongar la culminación del sprint.

Tabla 5: Sprints planificados

Sprint	**Product Backlog**	**Descripción**
Sprint N° 1	Definición del proyecto	Aquí se define los requerimientos y alcance del proyecto.
Sprint N° 2	Módulo de usuarios	Se implementa el módulo de usuarios con el sistema de autentificación y permisos de usuario con sus respectivos roles en el lado del back-end.
Sprint N° 3	Módulo de Administración	Creación de usuarios, asignación de roles, acceso a la cuenta, restauración de cuentas en el lado del cliente.
Sprint N° 4	Módulo de médico	Creación del módulo de médicos relacionado con la especialidad a la que pertenece un médico.
Sprint N° 5	Módulo de enfermería	Creación de la vista para el rol enfermería junto con la vista de signos vitales.
Sprint N° 6	Módulo de paciente	Creación del CRUD del paciente junto con sus antecedentes médicos.
Sprint N° 7	Módulo de Ficha Médica.	Se desarrolla el módulo de ficha médica junto con el reporte generado.
Sprint N° 8	Pruebas de rendimiento	Se prepara y aplica las pruebas de rendimiento.

Elaborado por: Alexander Bonilla

Revisión del sprint

En esta etapa se presenta el avance logrado en cada sprint del proyecto al encargado del área de Tics del HUA.

Retroalimentación

En esta etapa se considerará: que se hizo mal durante el sprint, que se hizo bien, y los inconvenientes encontrados. Todo esto es para mejorar las cosas que se hicieron mal y mantener lo que generó buenos resultados, permitiendo el avance del proyecto.

3.1.1. IDENTIFICACIÓN DE VARIABLES

Variable Independiente

Tiempo de respuesta del servidor REST.

Variable Dependiente

El nivel de rendimiento del servidor API REST para la gestión de fichas médicas en el Hospital Universitario HUA

3.1.2. OPERACIONALIZACIÓN DE LAS VARIABLES

Tabla 6: Identificación de Variables

Variable	Tipo	Definición Conceptual	Dimensión	Indicadores
Endpoint públicos	Independiente	Las aplicaciones web SPA es una tecnología que permite reducir el tiempo de carga; Solo refresca los datos de los endpoint públicos y no toda la página web.	Desarrollo de una aplicación Web SPA que consume de un servidor REST	▪ Peso de la transacción de petición. ▪ Peso de la transacción de respuesta.
Nivel de rendimiento de los servicios API REST	Dependiente	Las Aplicaciones Web SPA solo consumen datos, por lo que es necesario tener un servidor API REST que responda con dichos datos.	Prueba de rendimiento	▪ Prueba de carga, número de peticiones y tiempo de carga. ▪ Tiempo de respuesta máximo. ▪ Tiempo de respuesta mínimo.

Elaborado por: Alexander Bonilla

3.2. TIPO Y DISEÑO DE LA INVESTIGACIÓN

3.2.1. TIPO DE ESTUDIO

SEGÚN LA FUENTE DE INFORMACIÓN

Investigación Bibliográfica: La investigación se apoyó en la revisión documental durante la etapa teórica y elaboración del documento, esta información se obtiene de: libros, artículos científicos, publicaciones y revistas.

SEGÚN EL MÉTODO A UTILIZAR

Método Deductivo: Según los resultados de rendimiento de Apache JMeter, se puede inferir si el rendimiento del servidor en la aplicación de fichas médicas del HUA es óptimo.

Método Bibliográfico: Identificar los recursos más importantes que brindan la información relevante y la documentación necesaria sobre los beneficios de las aplicaciones web SPA, Servicios REST y las métricas necesarias para determinar si el rendimiento del servidor REST es el óptimo.

3.3. UNIDAD DE ANÁLISIS

A partir del análisis, se establecieron un conjunto de peticiones que la herramienta de testing Apache JMeter solicitó al servidor REST para aplicarle una carga y simular las peticiones HTTP mediante los diferentes verbos HTTP como GET, POST, PATCH, etc. Los indicadores de calidad utilizados en este estudio se establecieron después de analizar estudios en el desempeño de servidores REST con estándares de calidad ISO/IEC 25010, determinando así el método ideal para analizar sistemas informáticos y servicios web.

3.4. POBLACIÓN Y MUESTRA

3.4.1. POBLACIÓN

En el presente trabajo investigativo se estableció a las peticiones realizadas como la población a estudiar, debido a que el número de peticiones realizadas a un servidor suele ser indefinidas se optó por tomar una población infinita.

3.4.2. MUESTRA

Considerando que se establece una población infinita, la muestra se puede obtener aplicando una fórmula de muestreo a la población infinita. Esta muestra probabilística permite inferir valores muestrales de la población, reduciendo así el uso de tiempo y recursos.

Dónde:

n : Muestra

Z : Nivel de confianza.

p : Probabilidad de éxito.

q : Probabilidad de fracaso.

e : Error de muestra.

Formula:

$$n = \frac{Z^2 * p * q}{e^2}$$

Cálculo de Muestra:

$Z = 95{,}5\,\% => 2{,}005$: (Según la tabla de distribución Normal)

$p = 50\,\% => 0{,}5$

$q = 50\,\% => 0{,}5$

$e = 4{,}5\,\% => 0{,}045$

$$\begin{aligned} n &= \frac{Z^2 * p * q}{e^2} \\ &= \frac{2{,}005^2 * 0{,}5 * 0{,}5}{0{,}045^2} \\ &= \frac{1,00500625}{0,002025} \\ &= 496,299382716 \end{aligned} \tag{3.1}$$

3.5. TÉCNICAS DE RECOLECCIÓN DE DATOS

Después de ejecutar una prueba de rendimiento con Apache JMeter, esta herramienta devuelve un conjunto de resultados como: el tiempo de respuesta, bytes recibidos, bytes enviados, porcentaje de errores que ocurrieron en el servidor, etc.

3.6. TÉCNICAS DE ANÁLISIS E INTERPRETACIÓN DE RESULTADOS

3.6.1. HERRAMIENTAS UTILIZADAS

Las herramientas necesarias para analizar e interpretar los resultados de rendimiento del servidor REST se describen a continuación.

Tabla 7: Herramientas utilizadas para interpretar los resultados de rendimiento

VirtualBox	Software de virtualización para desplegar la aplicación.
Apache JMeter	Herramienta para probar el comportamiento funcional y medir el rendimiento.
ISO/IEC 25010	Modelo de Calidad de Software.
Google Sheets	Servicio web de hojas de cálculo, utilizado para la generación de gráficos.

Elaborado por: Alexander Bonilla

3.6.2. DESARROLLO DE LA APLICACIÓN

Para el desarrollo de la aplicación web SPA y los servicios web REST se utiliza la metodología ágil Scrum. Además de un conjunto de tecnologías para agilizar la implementación del sistema informático que se describe a continuación: Para la gestión de datos se utilizó el Sistema Gestor de Base de Datos PostgreSQL junto con Beekeeper Studio para realizar consultas a la base de datos, el back-end está soportado por el framework especializado en la creación de API REST LoopBack y la aplicación cliente está desarrollada con el framework Vue.js para construir las interfaces de usuario SPA. La codificación del back-end y front-end se realizó utilizando el lenguaje de programación TypeScript y para la ejecución del servidor se utilizó Node.js.

Tabla 8: Herramientas utilizadas para el desarrollo

ClickUp	Herramienta para la gestión de proyectos Scrum.
Git	Software de control de versiones.
GitHub	Portal web basado en Git para alojar el código fuente.
Node.js	Entorno de ejecución de JavaScript.
Beekeeper Studio	Editor SQL y gestor de bases de datos.
PostgreSQL	Sistema Gestor de Base de Datos.
Insomnia	Cliente API para servicios REST.
Visual Studio Code	Es un editor de código

Elaborado por: Alexander Bonilla

3.6.3. PRUEBAS DE RENDIMIENTO

Durante la etapa de pruebas de rendimiento se utilizó las métricas de calidad de la norma ISO/IEC 25010 para interpretar los resultados de respuesta del servidor arrojados por Apache JMeter.

CAPÍTULO 4

RESULTADOS Y DISCUSIÓN

Para esta investigación se consideró desarrollar una aplicación web SPA (Sisho), destinada a la gestión de fichas médicas en el Hospital Universitario Andino de la ciudad de Riobamba, utilizando servicios web de tipo REST. Esta investigación tiene un enfoque cuantitativo, utiliza los resultados de operación de la aplicación Sisho para la evaluación de parámetros de calidad y rendimiento, además el análisis estadístico asociado a estas características. La toma de los datos necesarios pertenecientes a este trabajo de investigación provienen del registro de transacciones generadas por la herramienta Apache JMeter configurada para que emule el comportamiento del usuario y simule las respectivas peticiones al servidor web REST considerando un orden lógico de las operaciones y distintos tiempos del lanzamiento para cada transacción. Basado en los estándares de calidad propuestos en la norma ISO/IEC 25010 se seleccionó el que corresponde a *Eficiencia de desempeño* para medir el nivel de rendimiento de las aplicaciones con base en la arquitectura de cliente-servidor.

4.1. RESULTADOS

De acuerdo a la metodología propuesta en el marco metodológico para el análisis del estudio de rendimiento, en base a la ecuación 3.1 de población y muestra se aplicó una carga de 496 solicitudes a los diferentes endpoints que componen los módulos del sistema. En la **Tabla 9** se detalla los endpoints a los que se aplicó las cargas simuladas. También se estableció un tiempo exigido "T_0"de 400 ms de acuerdo a lo que se establece en la métrica de rendimiento del servicio y se aplicó la fórmula 2.1.

Tabla 9: Endpoints a los que se aplicaron las pruebas de rendimiento

Módulo	**Descripción**	**Endpoints**
Login	Acceso al sistema informático.	▪ /api/account/login ▪ /api/account/me ▪ /api/myrole/modules
Pacientes	Registro de los pacientes.	▪ /api/patient
Consultas	Consulta de los registros.	▪ /api/profiles ▪ /api/medics ▪ /api/diseasetypes ▪ /api/diseasetype/:id/diseases ▪ /api/examtypes ▪ /api/examtype/:id/exams ▪ /api/patients
Ficha médica	Registro de las fichas médicas.	▪ /api/medics ▪ /api/patients ▪ /api/patient/:id/medicalrecord ▪ /api/medicalrecord/:id/vitalsign ▪ /api/medicalrecord/:id ▪ /api/medicalrecord/:id/rpe ▪ /api/medicalrecord/:id/cros ▪ /api/medicalrecord/:id/medicalexams ▪ /api/medicalrecord/:id/diagnostics

Elaborado por: Alexander Bonilla

Las pruebas de rendimiento se aplicaron al servidor que se encuentra desplegado en un equipo con las características que se describen a continuación:

Tabla 10: Características del hardware en el servidor

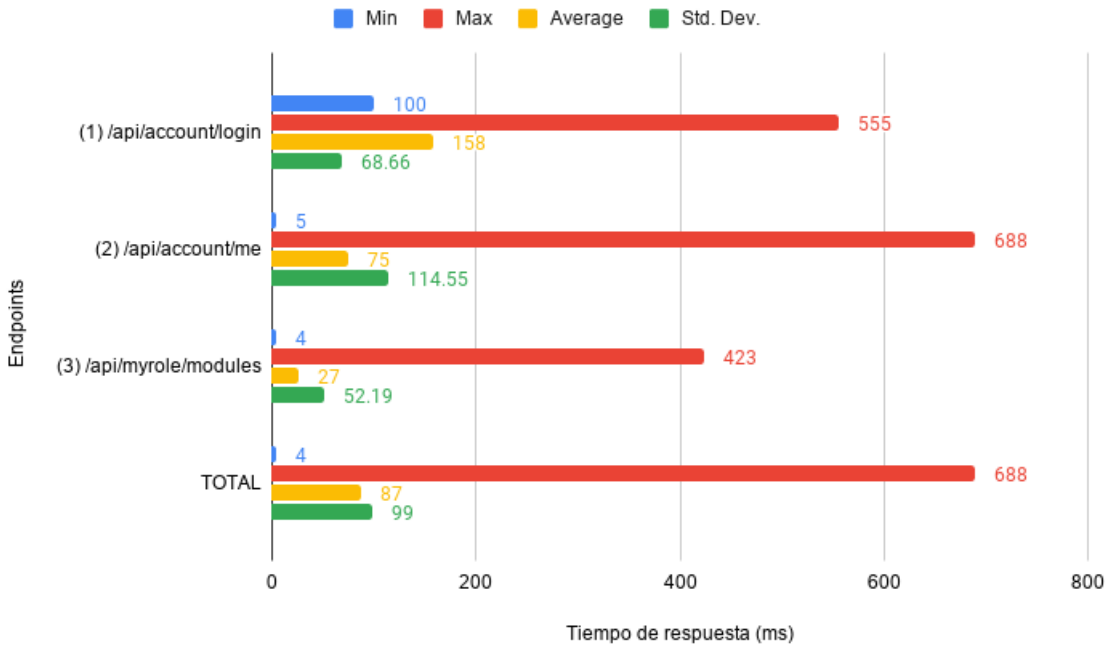
Sistema Operativo	Ubuntu 20.04.1 LTS
Arquitectura	x64
Memoria RAM	2048 MB
Almacenamiento	30 GB

Elaborado por: Alexander Bonilla

4.1.1. Módulo login

Como se describe en la **Tabla 9**, el módulo de login cuenta con tres endpoints que trabajan en conjunto para proporcionar acceso a la aplicación. Por cada operación de inicio de sesión se sigue la secuencia que se muestra en la **Figura 7**.

Figura 7: Rendimiento del módulo de login.



Elaborado por: Alexander Bonilla

Los valores utilizados para el cálculo del rendimiento se los puede observar en el anexo 5.9.

$$w = \frac{Nro(T^w < T_0 \ ; \ en\ periodo\ t)}{Nro(Peticiones^w \ ; \ en\ periodo\ t)} = \frac{1467}{1488} = 0{,}98581 = 98{,}58\,\% \qquad (4.1)$$

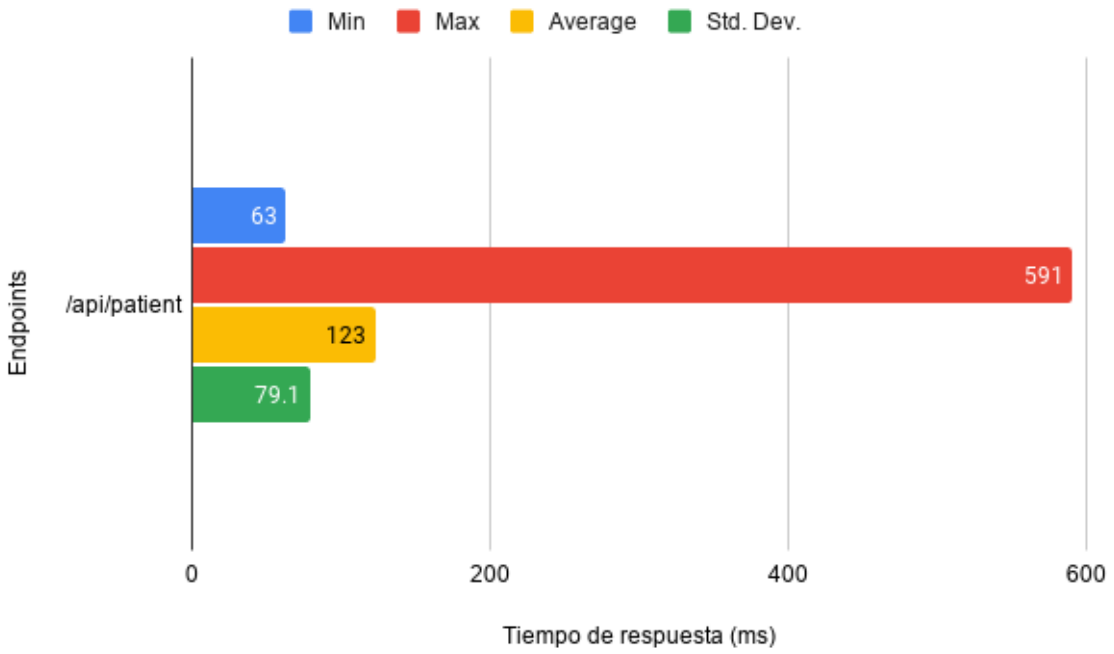
Análisis

La gráfica muestra el tiempo que tarda el servidor en devolver una respuesta al cliente. Se puede observar que en los momentos de menor carga el tiempo de respuesta es de apenas 4 ms, mientras con una mayor carga los tiempos de respuesta pueden llegar hasta un valor máximo de 688 ms, sin embargo el promedio del tiempo de respuesta no supera los 99 ms. Después de aplicar la fórmula de rendimiento de servicio, el módulo de login alcanzará un rendimiento de 97,58 %.

4.1.2. Módulo de pacientes

Este módulo se utiliza para registrar la información del paciente, por lo que es un módulo con un solo endpoint como se muestra en la **Tabla 9**.

Figura 8: Rendimiento del módulo de pacientes.



Elaborado por: Alexander Bonilla

$$w = \frac{Nro(T^w < T_0 \ ; \ en\ periodo\ t)}{Nro(Peticiones^w \ ; \ en\ periodo\ t)} = \frac{484}{496} = 0{,}97581 = 97{,}58\,\% \qquad (4.2)$$

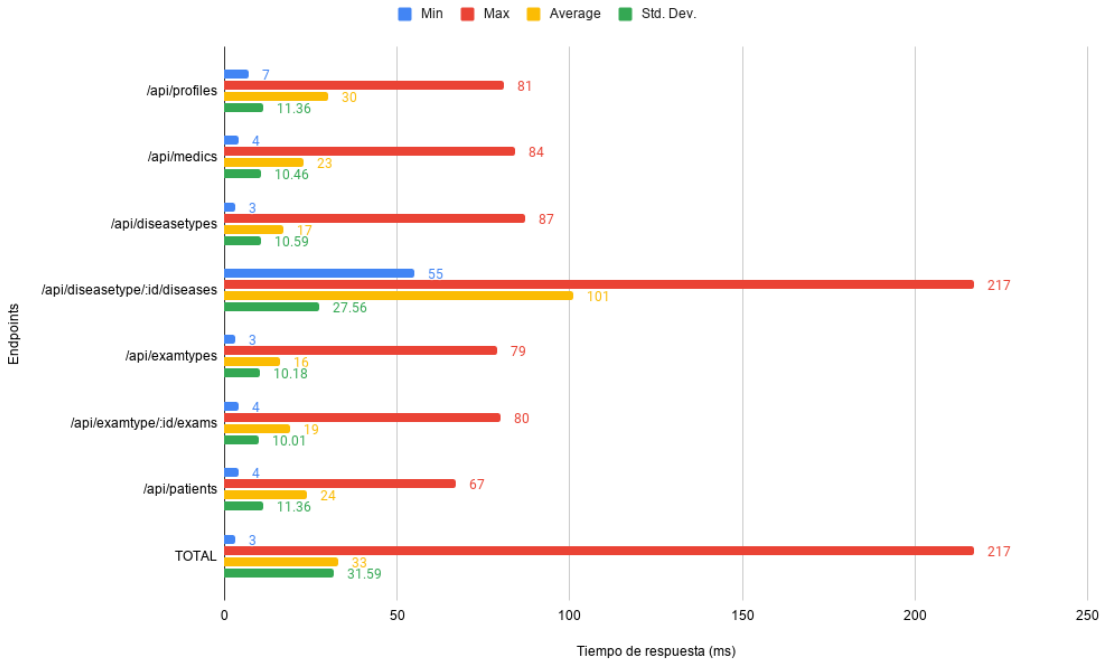
Análisis

Se puede observar que en los momentos de menor carga el tiempo de respuesta es de 63 ms, mientras con una mayor carga pueden llegar hasta un valor máximo de 591 ms y un promedio del tiempo de respuesta igual a 79.1 ms. Brindando un rendimiento del 97.58 %.

4.1.3. Módulo de consultas

Este conjunto de endpoints proporciona información sobre todos los registros existentes en la base de datos.

Figura 9: Rendimiento del módulo de consultas.



Elaborado por: Alexander Bonilla

$$w = \frac{Nro(T^w < T_0 \; ; \; en \; periodo \; t)}{Nro(Peticiones^w \; ; \; en \; periodo \; t)} \quad = \frac{3472}{3472} = 1{,}0 \quad = 100\,\% \tag{4.3}$$

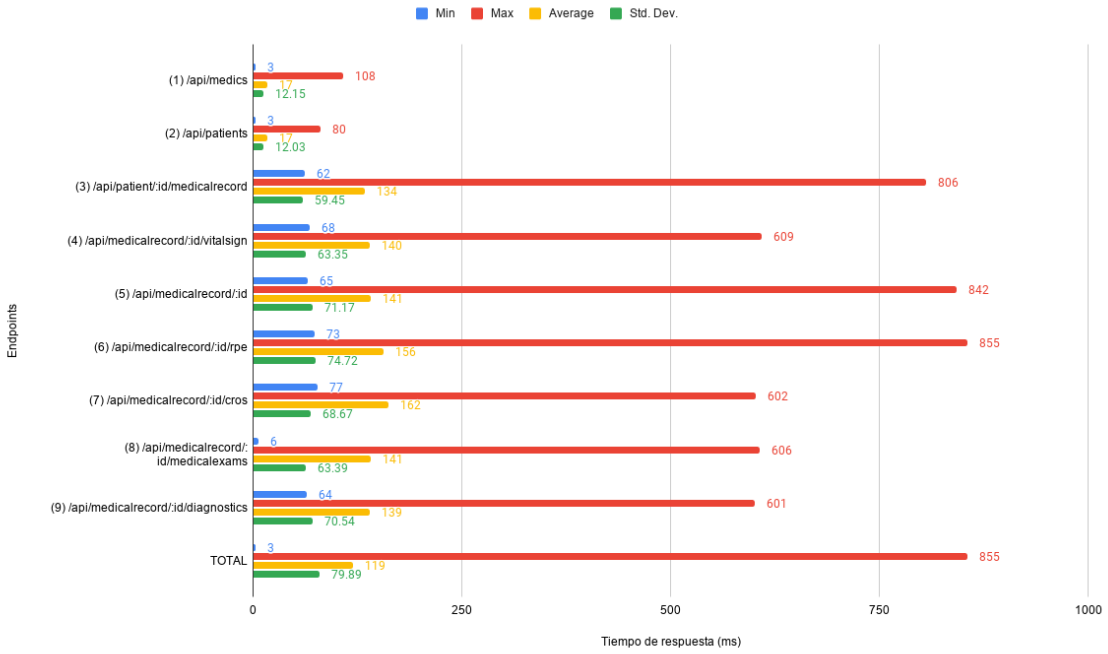
Análisis

Esta gráfica muestra que a menor carga el tiempo de espera es de solo 3 ms, a mayor carga el tiempo alcanzar el valor máximo de 217 ms y un tiempo de respuesta promedio igual a 33 ms. Mostrando que este módulo esta por debajo del tiempo establecido por la métrica de rendimiento de servicio mostrando un rendimiento del 100 %.

4.1.4. Módulo de ficha médica

Este módulo administra la información de la ficha médica, por lo que utiliza una gran cantidad de endpoints y por cada operación de registro de una ficha médica se sigue la secuencia descrita en la **Tabla 9** y representada a continuación en la **Figura 10**.

Figura 10: Rendimiento del módulo de ficha médica.



Elaborado por: Alexander Bonilla

$$w = \frac{Nro(T^w < T_0 \ ; \ en\ periodo\ t)}{Nro(Peticiones^w \ ; \ en\ periodo\ t)} = \frac{4901}{4960} = 0{,}9881 = 98{,}81\,\% \qquad (4.4)$$

Análisis

Se puede observar que en los momentos de menor carga el tiempo de respuesta es de 3 ms, mientras con una mayor carga puede llegar hasta un valor máximo de 855 ms con un promedio del tiempo de respuesta igual a 119 ms. El rendimiento del módulo de la ficha médica es del 98.81 %.

Al promediar el rendimiento de los cuatro módulos: login, paciente, búsquedas y ficha médica el rendimiento de la aplicación es de 99.12 % como se evidencia en el anexo 5.9.

4.2. DISCUSIÓN

A partir de la investigación realizada considerando las características y beneficios tanto de las aplicaciones web SPA y de los servicios REST se determina que una aplicación web SPA es adecuada para la gestión de información, razón por la que se desarrolló una aplicación web SPA (Sisho) para la gestión de fichas médicas en el Hospital Universitario Andino de Chimborazo (HUA). Para evaluar el rendimiento del servidor se aplicó un conjunto de cargas para estresar el servidor y simular el trabajo del usuario, con el propósito de saber si el rendimiento de la aplicación se ajustaría a las necesidades operativas de la institución al gestionar las fichas médicas.

Al analizar los resultados registrados por las pruebas de rendimiento se evidenció que el tiempo de respuesta promedio es inferior a los 400 ms establecido por la métrica de rendimiento de servicio, lo que permite contrastar de acuerdo con las directrices de rendimiento de servicio que el servidor REST para administrar las fichas médicas proporciona un rendimiento satisfactorio en conjunto con una aplicación Web SPA.

CONCLUSIONES

- Se determinó que las aplicaciones web SPA y los servidores REST son tecnologías idóneas para el desarrollo de software que gestiona información sobre historias clínicas de unidades médicas. Tecnologías que gracias a la arquitectura cliente-servidor permiten obtener aplicaciones web más veloces al cargar la aplicación completa solo una vez, y evitar recargar con cada operación que el usuario realiza.

- Durante el proceso de desarrollo se aplicó la metodología ágil Scrum para construir la aplicación web SPA y el servidor REST para la gestión las fichas médicas, esta metodología permitió generar entregables en periodos cortos de tiempo, además para agilizar el desarrollo se utilizó frameworks especializados como LoopBack y Vue.js en la creación de aplicaciones SPA y servidores REST. Esto garantiza costos de desarrollo bajos al ser herramientas de software libre.

- La herramienta Apache JMeter permite evaluar el rendimiento de los servicios REST aplicando un conjunto de cargas y arrojando los tiempos de respuesta mínimo, máximo y promedio de las peticiones realizadas a los diferentes endpoint pertenecientes a los módulos de la aplicación, a partir de los resultados proporcionados por Apache JMeter se puede concluir que el servidor REST proporciona un rendimiento satisfactorio en la gestión de ficha médicas con tiempos de respuesta promedio medio menores a 400 ms de acuerdo con la metrica de rendimiento de servicio.

RECOMENDACIONES

- Para analizar los beneficios de las aplicaciones SPA y los servicios REST se recomienda utilizar diferentes fuentes bibliográficas sobre las tecnologías que permiten la creación de este tipo de aplicaciones, además investigar las tecnologías que permiten la creación de aplicaciones web no tradicionales para poder obtener una mayor comprensión de los beneficios de las aplicaciones SPA y los servidores REST.

- Durante el desarrollo del back-end se recomienda la utilización de pruebas unitarias aplicadas a los endpoints sel servidor, puesto que con los continuos cambios en el código pueden producir varios errores. También se recomienda la utilización de una herramienta para el control de versiones como Git con la finalidad de encontrar de forma rápida los errores que se pueden producir durante la codificación de la aplicación y control de cambios.

- Para simular cargas al servidor REST se recomienda configurar la herramienta Apache JMeter de forma que simule el comportamiento de los usuarios respecto al tiempo entre la ejecución de cada petición, la exportación de los resultados se debe realizar al finalizar las pruebas para que Apache JMeter no presente fallos.

BIBLIOGRAFÍA

Alvarez, M. (2016). Qué es una spa.

Arsaute, A., Zorzán, F. A., Daniele, M., González, A., and Frutos, M. (2018). Generación automática de api rest a partir de api java, basada en transformación de modelos (mdd). In *XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste).*

Basson, H., Bouneffa, M., Matsuda, M., Ahmad, A., Chung, D., and Arai, E. (2016). Qualitative evaluation of manufacturing software units interoperability using iso 25000 quality model. In *Enterprise interoperability VII*, pages 199–209. Springer.

Callejas, M., Alarcón Aldana, A. C., and Álvarez Carreño, A. M. (2017). Modelos de calidad del software, un estado del arte. *Entramado*, 13(1):236–250.

Cherny, B. (2019). *Programming TypeScript: making your JavaScript applications scale*. O'Reilly Media.

Chitra, L. P. and Satapathy, R. (2017). Performance comparison and evaluation of node. js and traditional web server (iis). In *2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET)*, pages 1–4. IEEE.

Daityari, S. (2020). Angular vs react vs vue: Which framework to choose in 2021.

Fierro, F. A. S., Manosalvas, C. A. P., Rodríguez, N. N. C., and Landeta, P. (2019). Análisis de la eficiencia de desempeño en aplicaciones de realidad aumentada utilizando la normativa iso/iec/25010. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E22):256–267.

Fox, N. (2020). Secure web application development. *The Students' Guide to Learning Design and Research*.

Google, D. (2018). Improve server response time.

Haro, E., Guarda, T., Peñaherrera, A. O. Z., and Quiña, G. N. (2019). Desarrollo backend para aplicaciones web, servicios web restful: Node. js vs spring boot. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E17):309–321.

Jader, O. H., Zeebaree, S., and Zebari, R. R. (2019). A state of art survey for web server performance measurement and load balancing mechanisms. *International Journal of Scientific & Technology Research*, 8(12):535–543.

Kaluža, M., Troskot, K., and Vukelić, B. (2018). Comparison of front-end frameworks for web applications development. *Zbornik Veleučilišta u Rijeci*, 6(1):261–282.

Kristensen, E. K. and Møller, A. (2017). Type test scripts for typescript testing. *Proceedings of the ACM on Programming Languages*, 1(OOPSLA):1–25.

Lanza Ortega, E. et al. (2019). Aplicación web progresiva (pwa) para la gestión de pagos de estacionamiento en superficie.

Macrae, C. (2018). *Vue. js: Up and Running: Building Accessible and Performant Web Apps*. .ºReilly Media, Inc.".

Matam, S. and Jain, J. (2017). *Pro Apache JMeter: web application performance testing*. Apress.

Montero, B. M., Cevallos, H. V., and Cuesta, J. D. (2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Espirales revista multidisciplinaria de investigación*, 2(17).

Moreno, M. M. (2007). Métricas para la interoperabilidad de la información en el gobierno electrónico. *Santiago de Chile*.

Nygård, K. et al. (2015). Single page architecture as basis for web applications.

Paz, J. A. M., Gómez, M. Y. M., and Rosas, S. C. (2017). Análisis sistemático de información de la norma iso 25010 como base para la implementación en un laboratorio de testing de software en la universidad cooperativa de colombia sede popayán. In *Memorias de Congresos UTP*, pages 149–154.

Ramidi, R. R. (2017). Performance evaluation of client-coordinated service chain.

Ramírez, M. R., Soto, M. d. C. S., Moreno, H. B. R., Rojas, E. M., Millán, N. d. C. O., and Cisneros, R. F. R. (2019). Metodología scrum y desarrollo de repositorio digital. *Revista Ibérica de Sistemas e Tecnologias de Informação*, (E17):1062–1072.

Rouse, M. (2019). Aplicación web (aplicación web).

Saeid, S. and Ali Yahiya, T. (2018). Load balancing evaluation tools for a private cloud: A comparative study. *ARO-The Scientific Journal of Koya University*, 6(2):13–19.

Sánchez, M. (2016). Assessing the quality of mooc using iso/iec 25010. In *2016 XI Latin American Conference on Learning Objects and Technology (LACLO)*, pages 1–4. IEEE.

Shivakumar, S. K. (2020). Modern web performance patterns. In *Modern Web Performance Optimization*, pages 273–300. Springer.

Solovei, V., Olshevska, O., and Bortsova, Y. (2018). The difference between developing single page application and traditional web application based on mechatronics robot laboratory onaft application. *Automation of technological and business processes*, 10(1).

Soni, A. and Ranga, V. (2019). Api features individualizing of web services: Rest and soap. *International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN*, pages 2278–3075.

Stepniak, W. and Nowak, Z. (2017). Performance analysis of spa web systems. In *Information Systems Architecture and Technology: Proceedings of 37th International Conference on Information Systems Architecture and Technology–ISAT 2016–Part I*, pages 235–247. Springer.

StrongLoop (2020). A highly extensible node.js and typescript framework for building apis and microservices.

Wohlgethan, E. (2018). *SupportingWeb Development Decisions by Comparing Three Major JavaScript Frameworks: Angular, React and Vue. js*. PhD thesis, Hochschule für Angewandte Wissenschaften Hamburg.

Yan, K. (2019). *An Overview of Modern Web Application Frameworks*. bravoka.

Young, A., Meck, B., and Cantelon, M. (2017). *Node. js in Action*. Manning Publ.

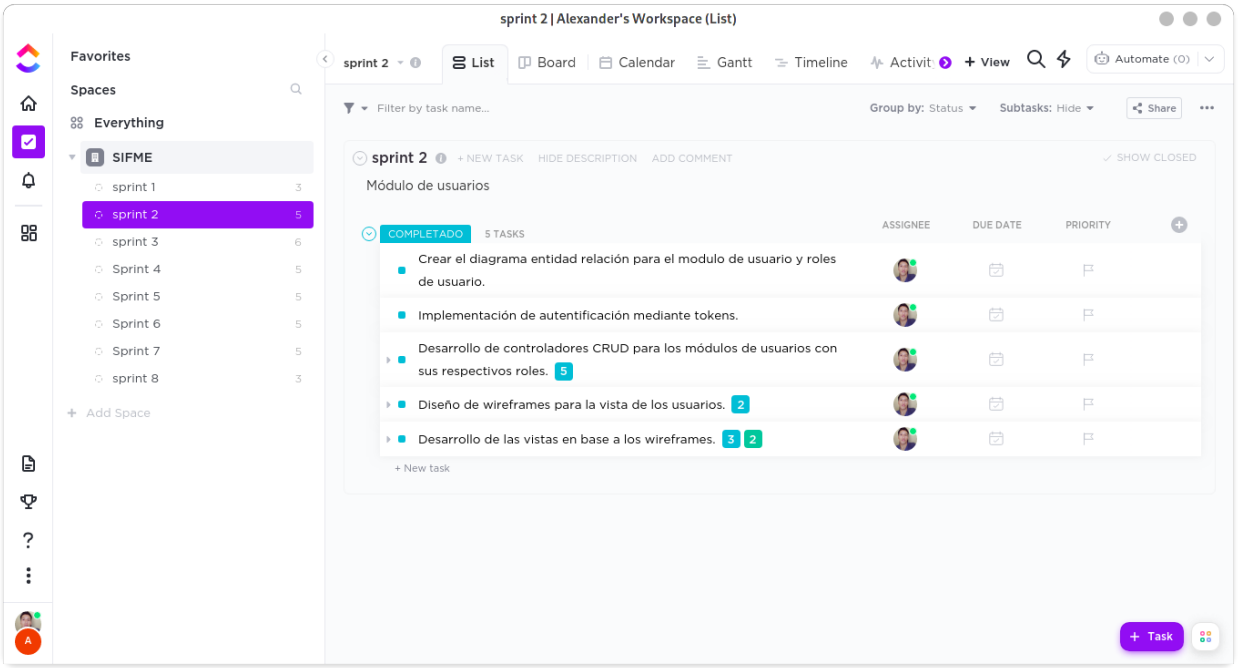
Zapata, M. (2018). Importancia del sistema grd para alcanzar la eficiencia hospitalaria. *Revista Médica Clínica Las Condes*, 29(3):347–352.

ANEXOS

5.1. Planificación de tareas

La herramienta ClickUp permite gestionar las tareas o sprints de la metodología ágil Scrum.

Figura 11: Planificación del módulo de usuarios.



5.2. Diseño y arquitectura de software

Previo al desarrollo de la aplicación se determina la funcionalidad de la aplicación con base en la toma de requerimientos funcionales y no funcionales presentados por la institución.

Figura 12: Casos de usos de los roles de usuario.

Figura 13: Modelo entidad relacional.

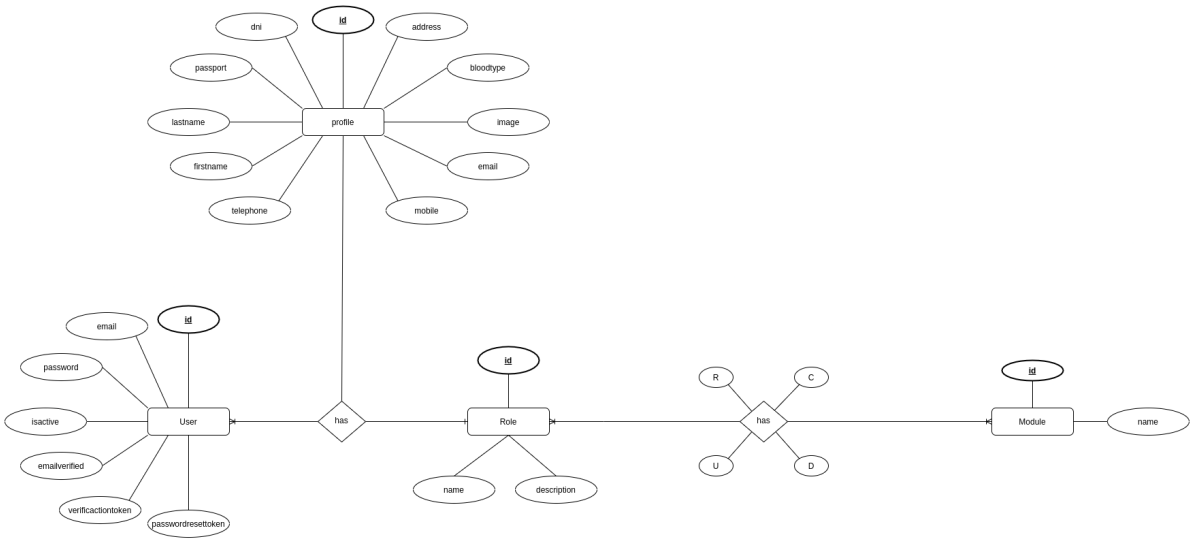
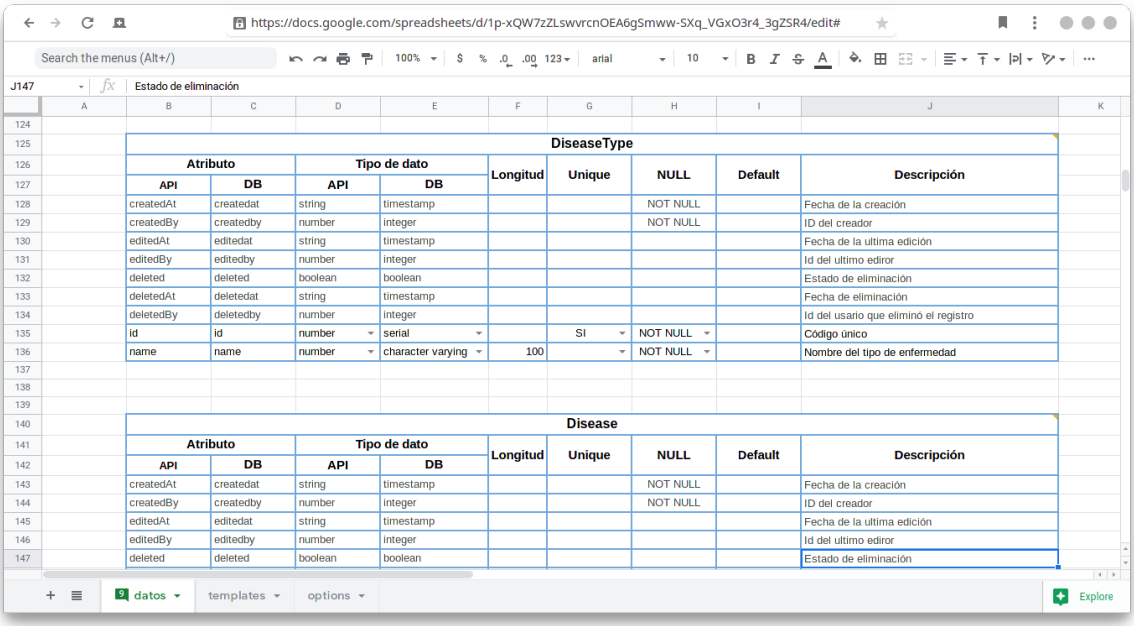


Figura 14: Diccionario de datos.



https://docs.google.com/spreadsheets/d/1p-xQW7zZLswvrcnOEA6gSmww-SXq_VGxO3r4_3gZSR4/edit#

Search the menus (Alt+/)

J147 | Estado de eliminación

124									
125	DiseaseType								
126	Atributo		Tipo de dato		Longitud	Unique	NULL	Default	Descripción
127	API	DB	API	DB					
128	createdAt	createdat	string	timestamp			NOT NULL		Fecha de la creación
129	createdBy	createdby	number	integer			NOT NULL		ID del creador
130	editedAt	editedat	string	timestamp					Fecha de la ultima edición
131	editedBy	editedby	number	integer					Id del ultimo ediror
132	deleted	deleted	boolean	boolean					Estado de eliminación
133	deletedAt	deletedat	string	timestamp					Fecha de eliminación
134	deletedBy	deletedby	number	integer					Id del usario que eliminó el registro
135	id	id	number	serial		SI	NOT NULL		Código único
136	name	name	number	character varying	100		NOT NULL		Nombre del tipo de enfermedad
137									
138									
139									
140	Disease								
141	Atributo		Tipo de dato		Longitud	Unique	NULL	Default	Descripción
142	API	DB	API	DB					
143	createdAt	createdat	string	timestamp			NOT NULL		Fecha de la creación
144	createdBy	createdby	number	integer			NOT NULL		ID del creador
145	editedAt	editedat	string	timestamp					Fecha de la ultima edición
146	editedBy	editedby	number	integer					Id del ultimo ediror
147	deleted	deleted	boolean	boolean					Estado de eliminación

datos | templates | options | Explore

Figura 15: Diagrama de la base de datos.

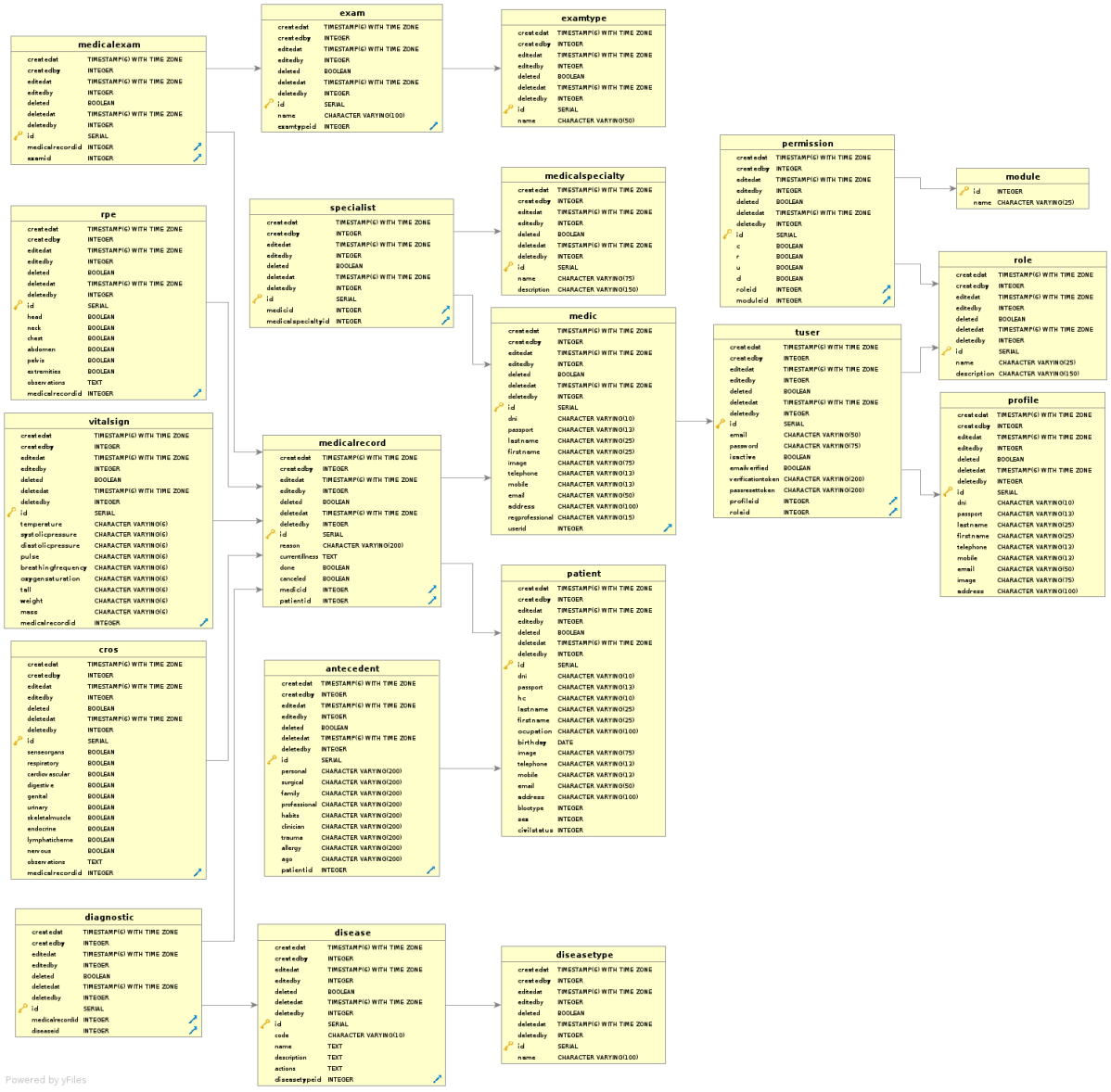


Figura 16: Wireframe de inicio de sesión.

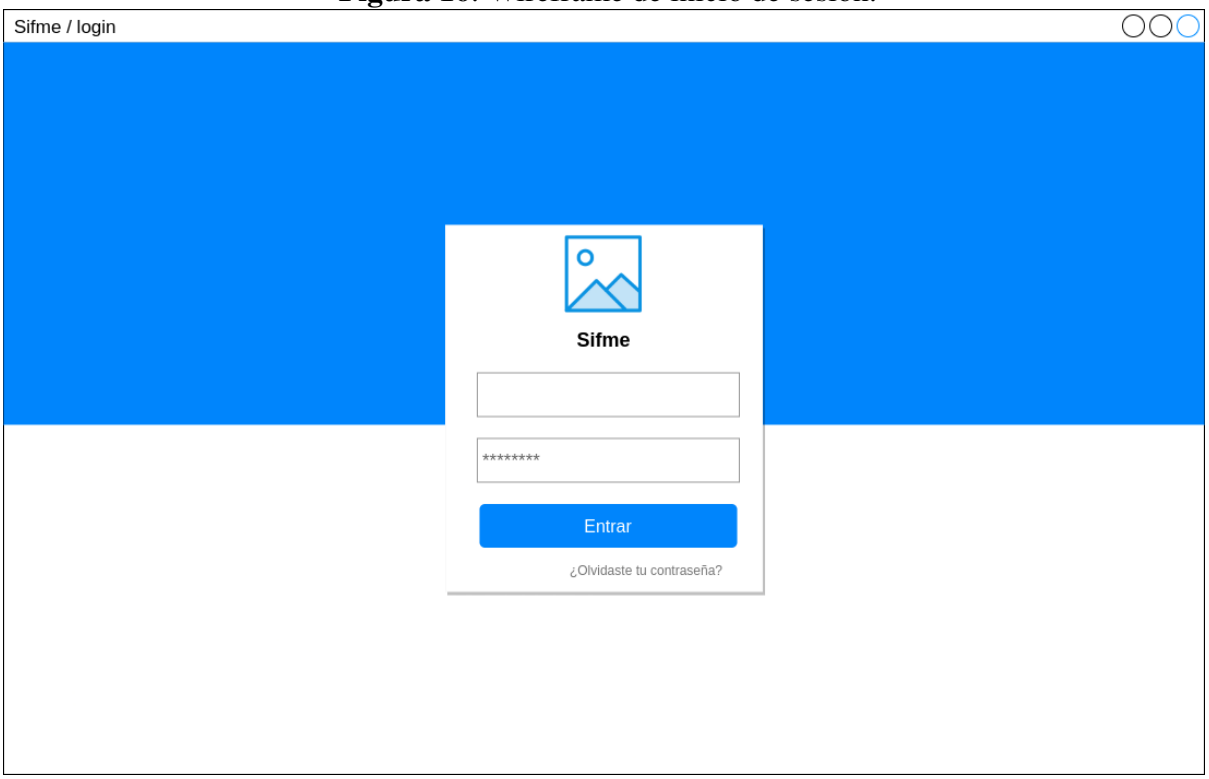
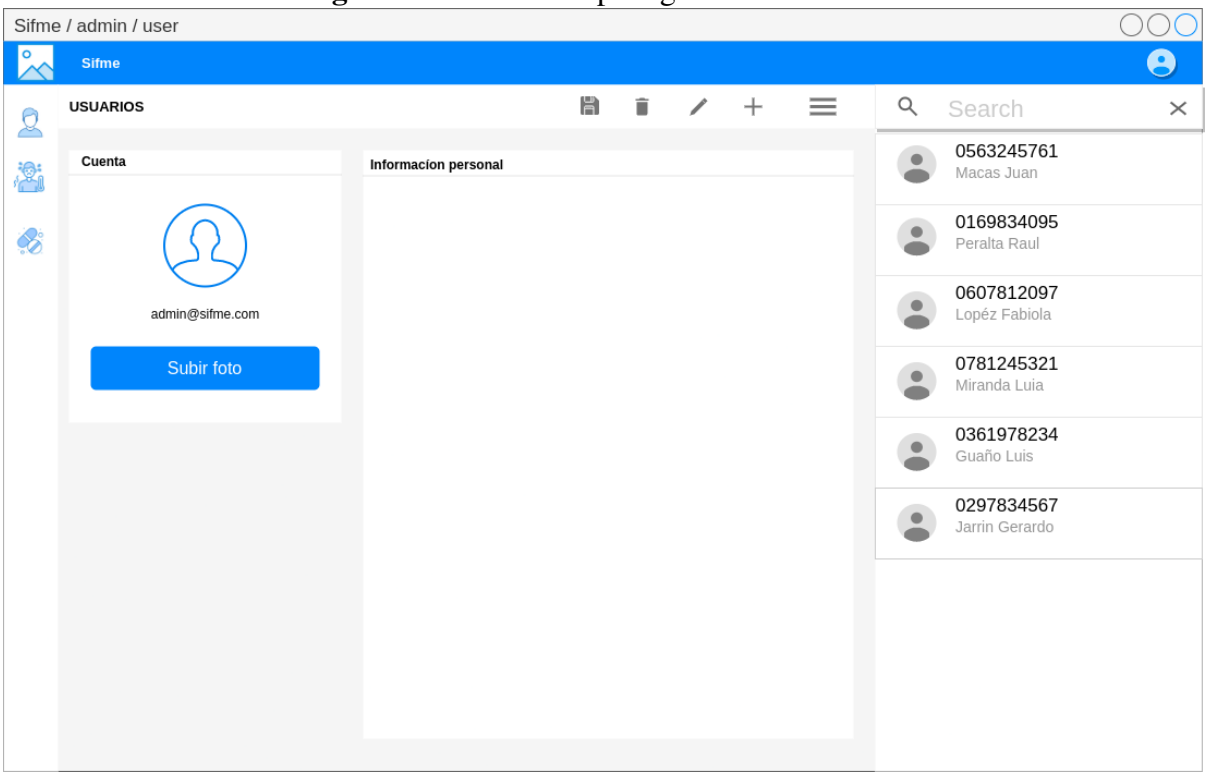


Figura 17: Wireframe para géstion de usuarios.



5.3. Documentación de los endpoints

El servidor REST tiene una serie de endpoints públicos y documentados, algunos son de acceso libre, mientras que en su mayoría son accesibles mediante un token de acceso perteneciente a una cuenta de usuario que solamente el administrador puede crear.

Figura 18: Endpoints del públicos del servidor REST.

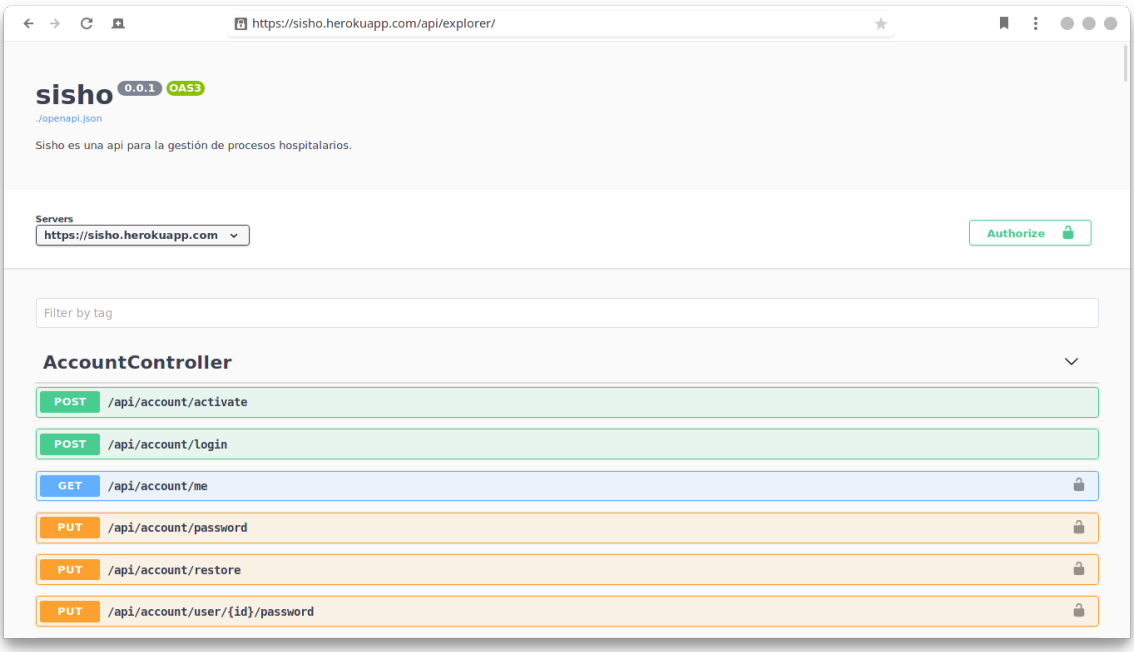


Figura 19: Esquema de datos para la realización de peticiones al servidor REST.

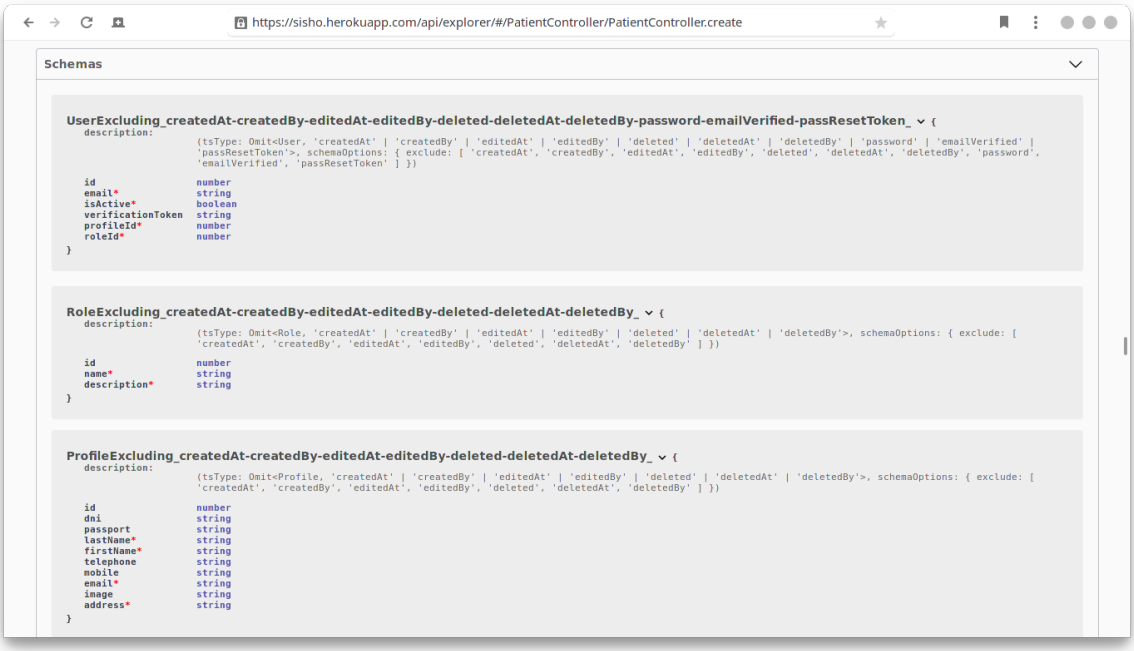
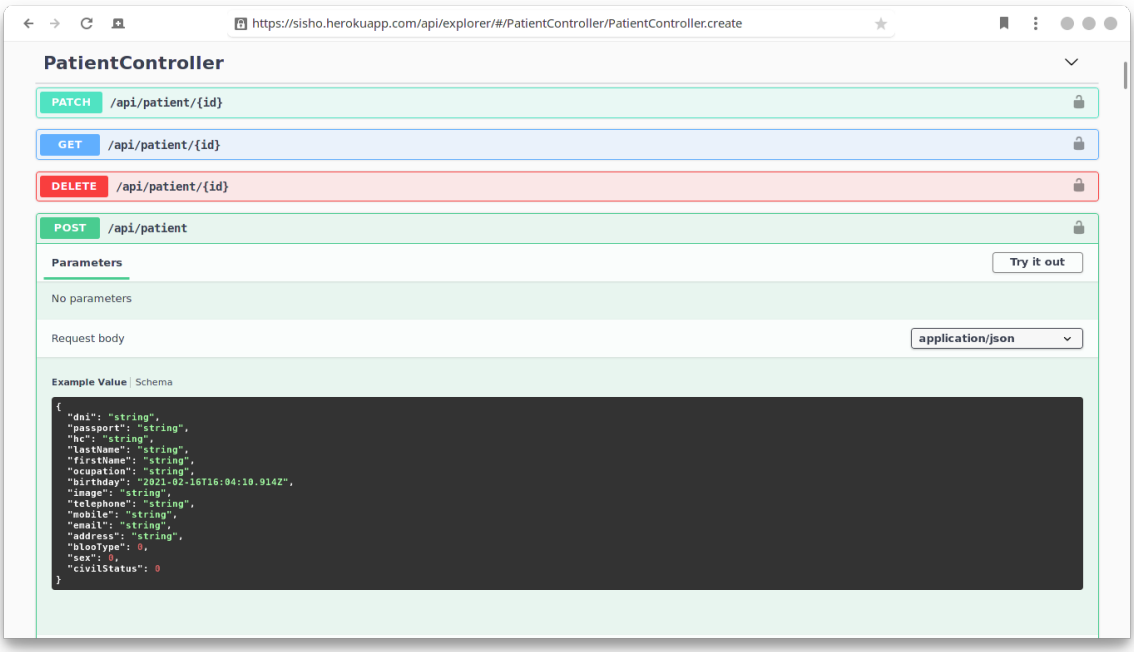


Figura 20: Petición de prueba a un endpoint.



5.4. Accesso a la aplicación web SPA

La aplicación posee un sistema de autentificación y permisos. El usuario con el rol de administración debe otorgar a los usuarios con diferente rol, para la activación de dicha cuenta se envía un correo electrónico con un codigo de activación único.

Figura 21: Splash de carga de la aplicación web SPA.

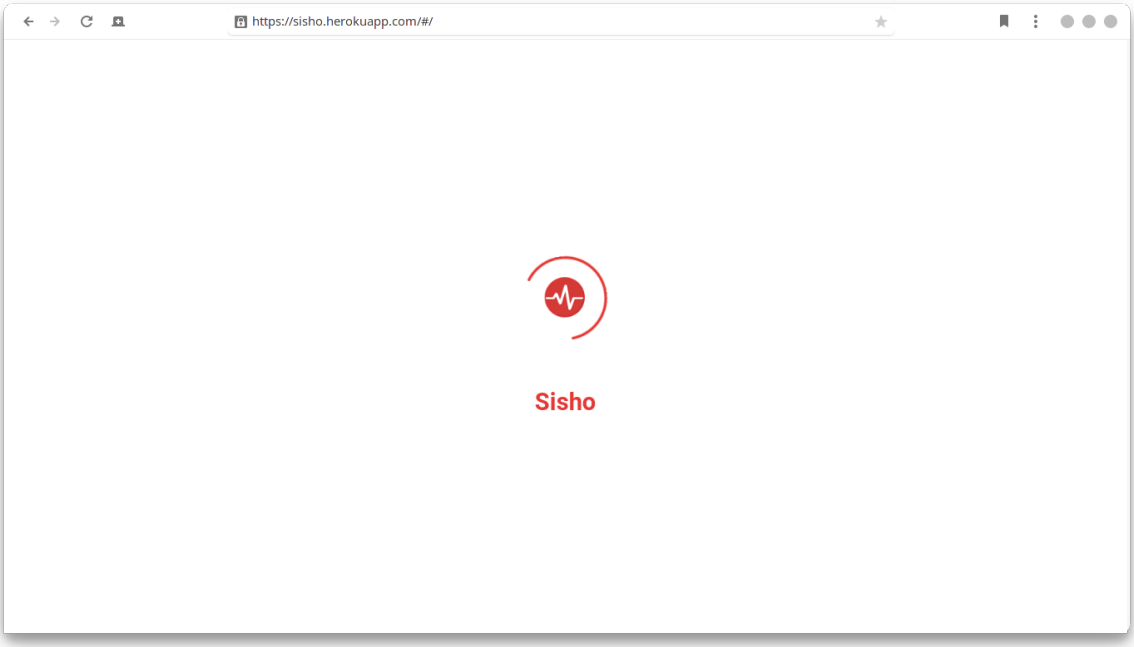


Figura 22: Correo de bienvenida con el código de activación de cuenta.

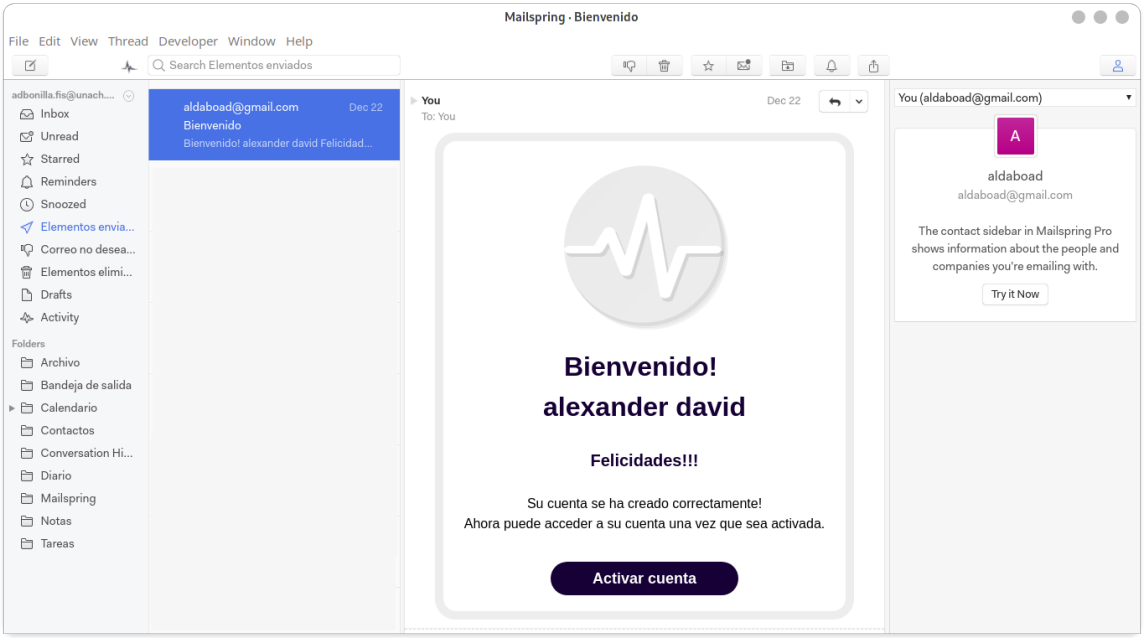


Figura 23: Página de activación de cuenta

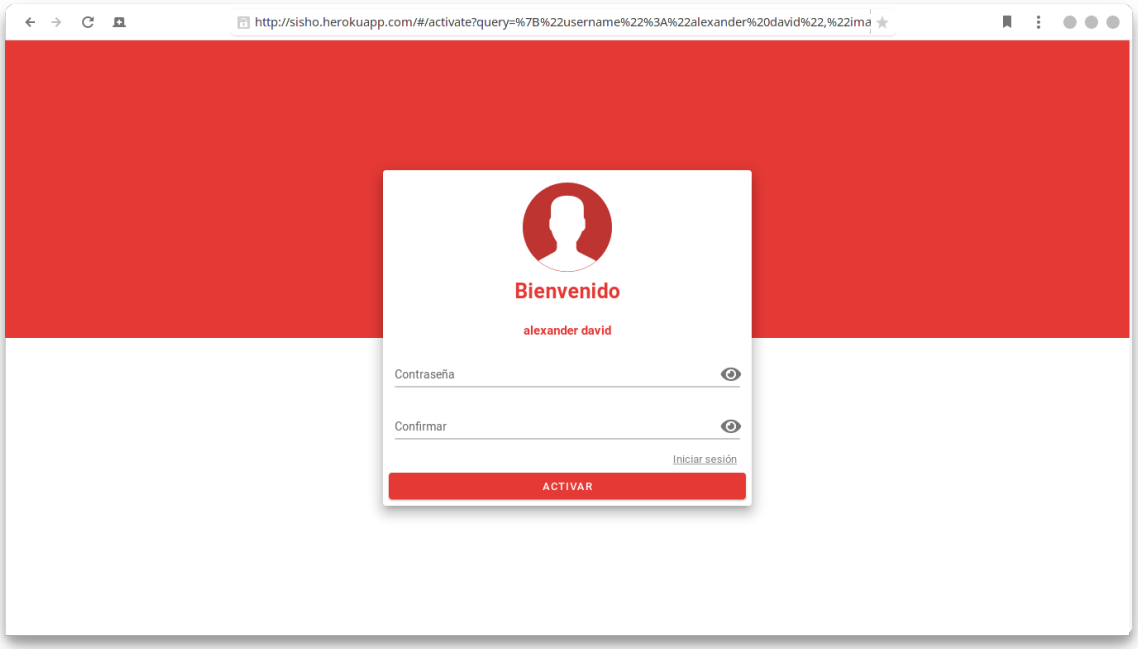


Figura 24: Página de login para todos los roles de usuario permitidos

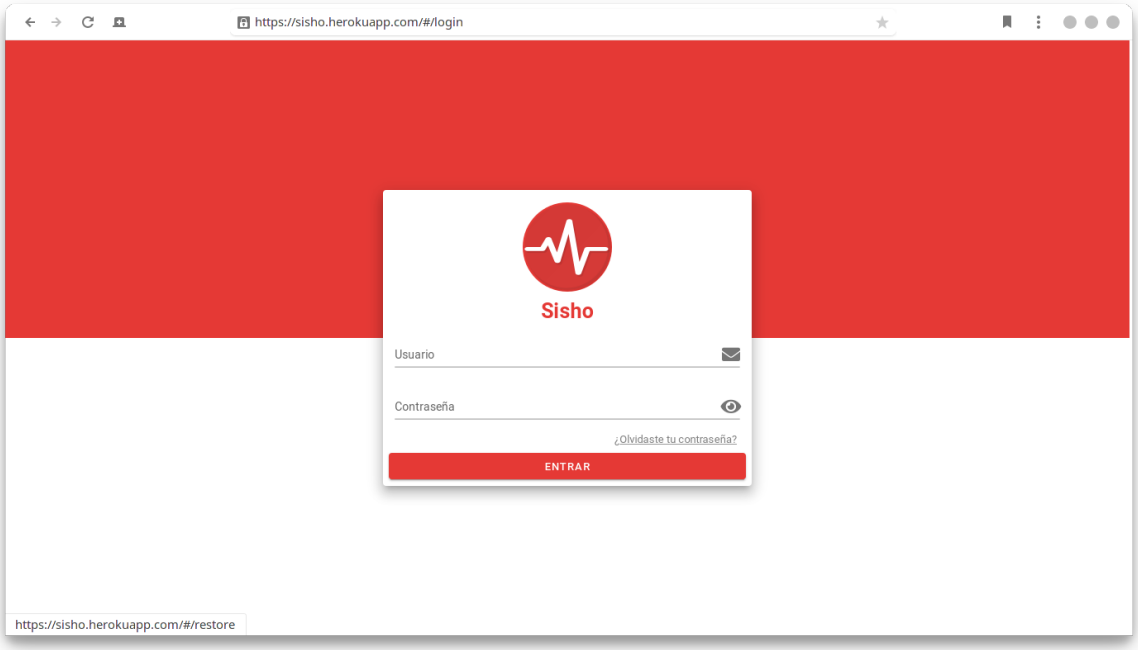
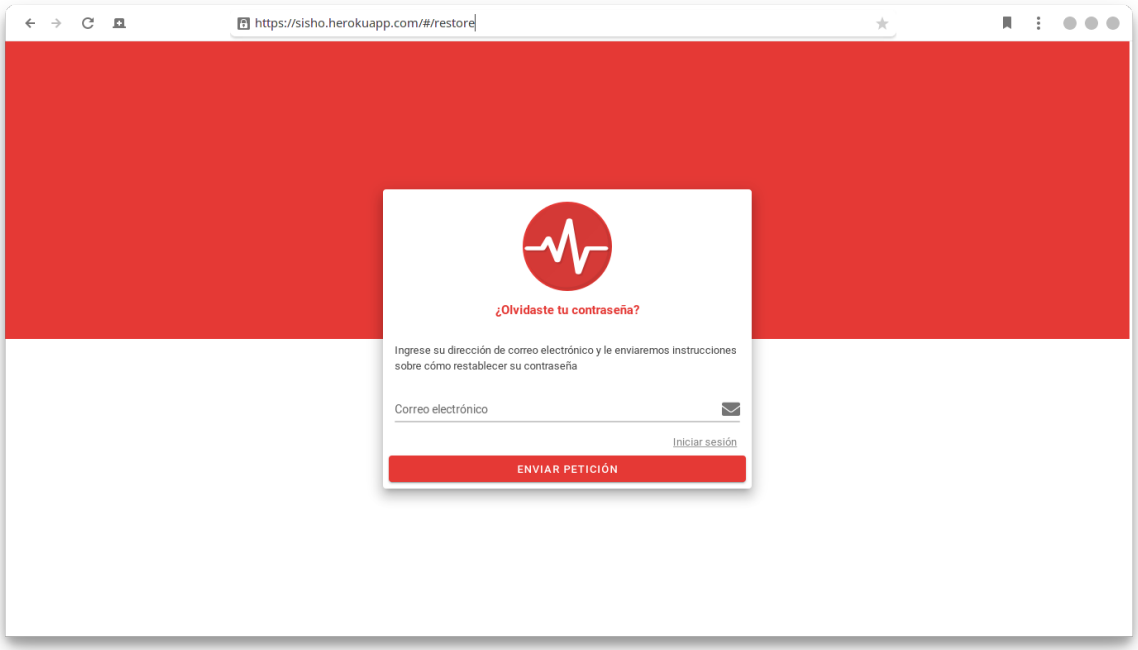


Figura 25: Página de restauración de contraseña



5.5. Aplicación web SPA para la administración

El usuario con rol de administración es el encargado de proporcionar toda la información necesaria para que el sistema funcione adecuadamente, así como mantener actualizada la información del personal.

Figura 26: Página principal de la aplicación

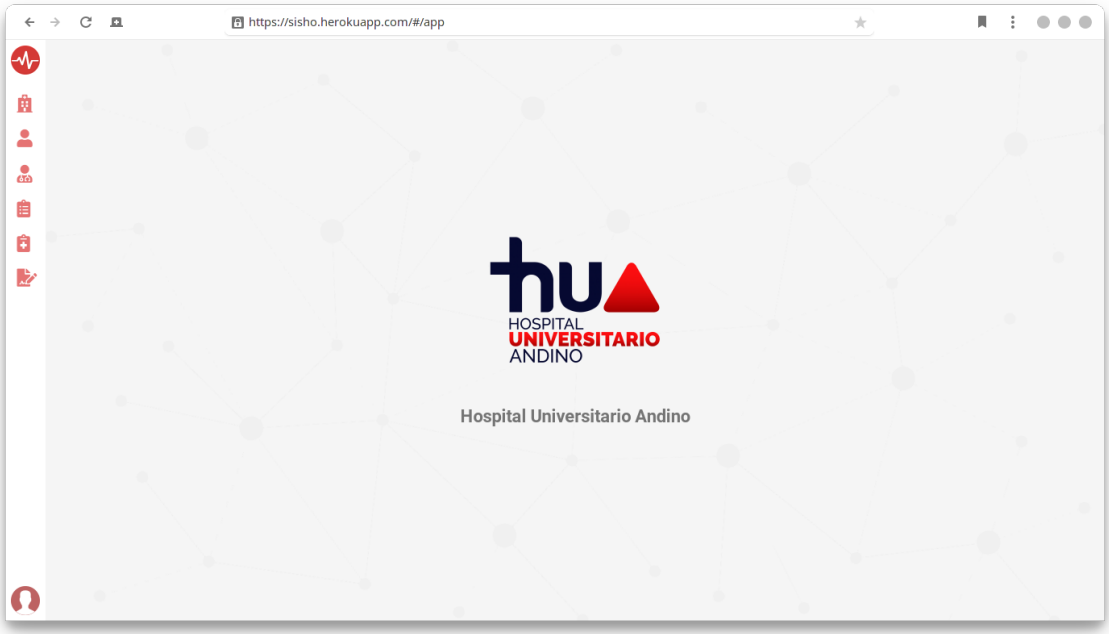


Figura 27: Página de configuración de la información del hospital.

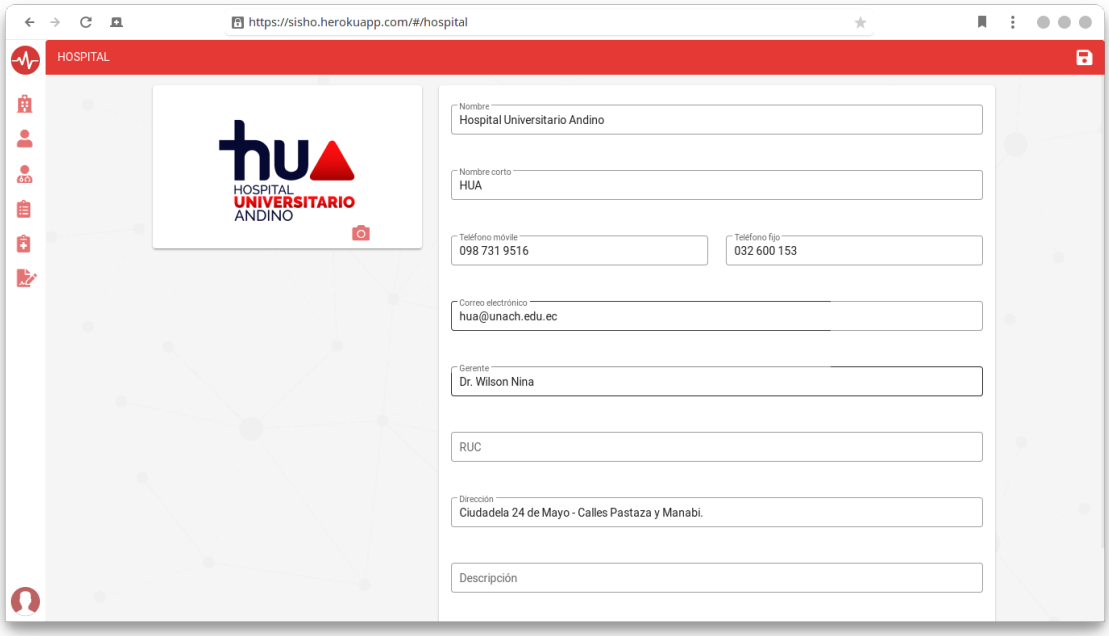


Figura 28: Página que muestra los usuarios del sistema.

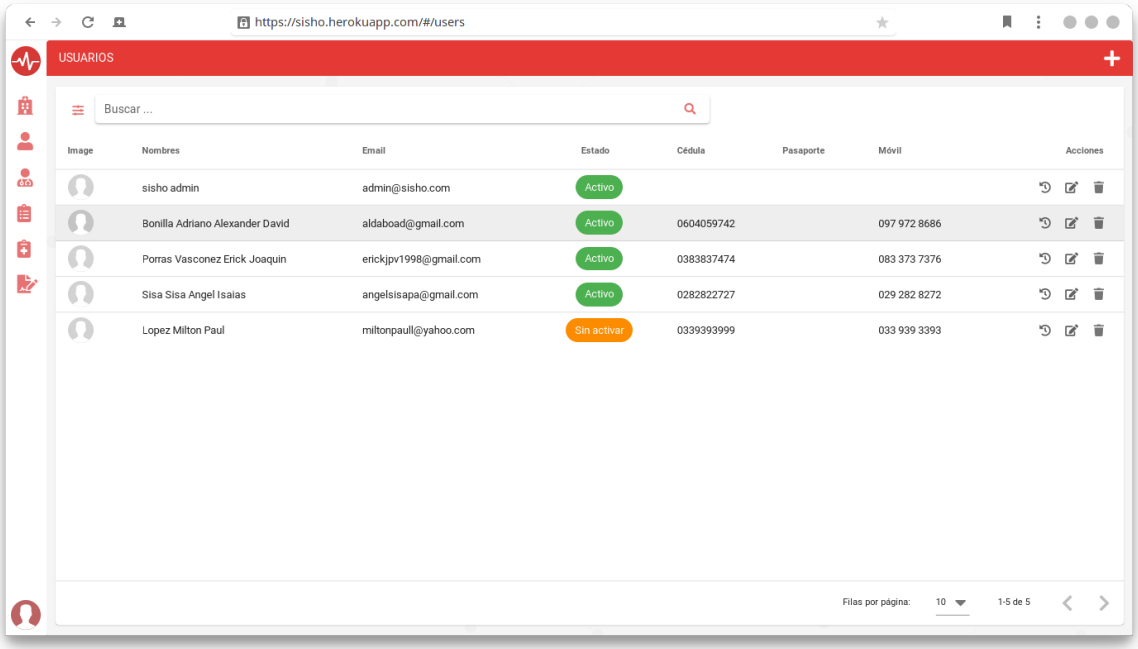


Figura 29: Página para la manipulación de los datos del usuario.

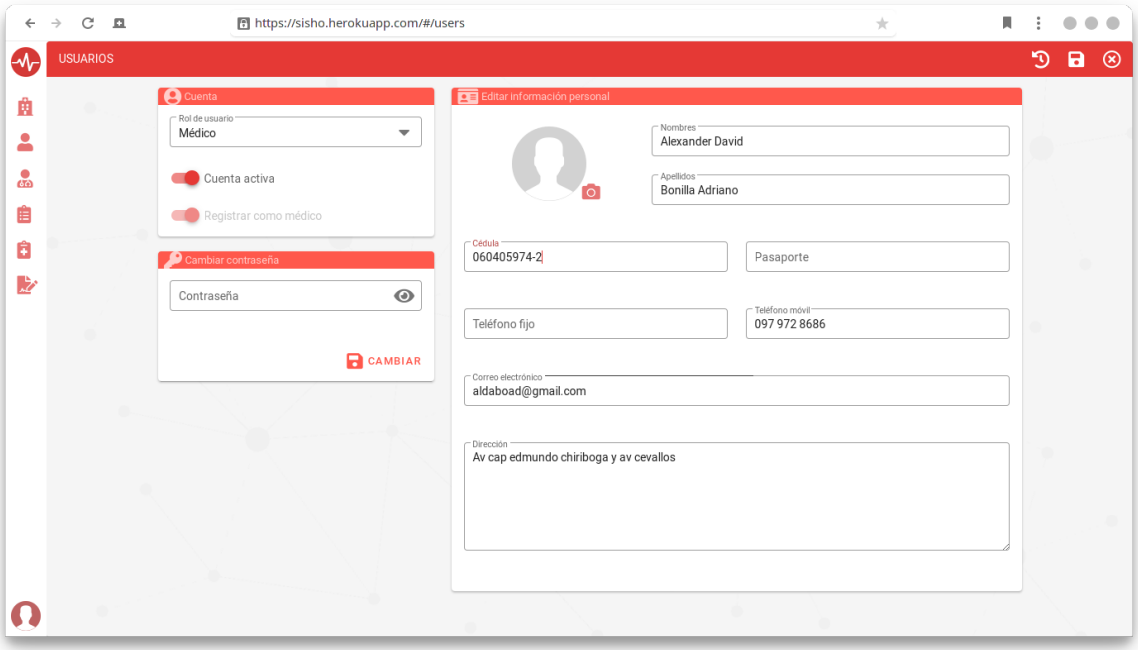


Figura 30: Página que muestra los médicos que poseen acceso al sistema.

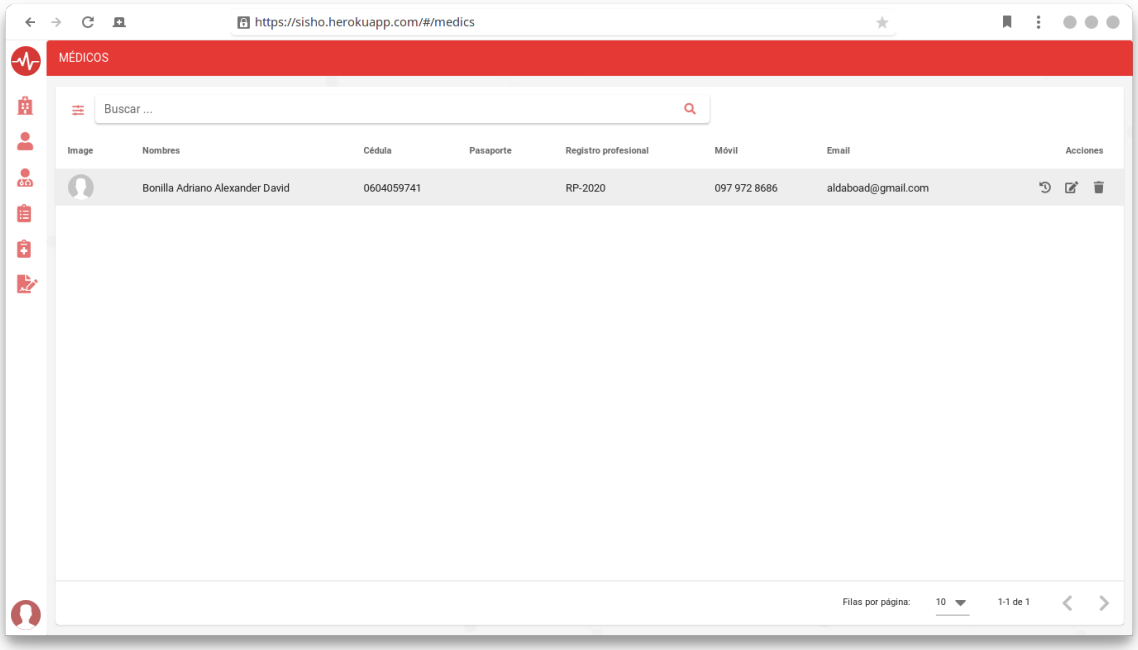


Figura 31: Página de examenes médicos.

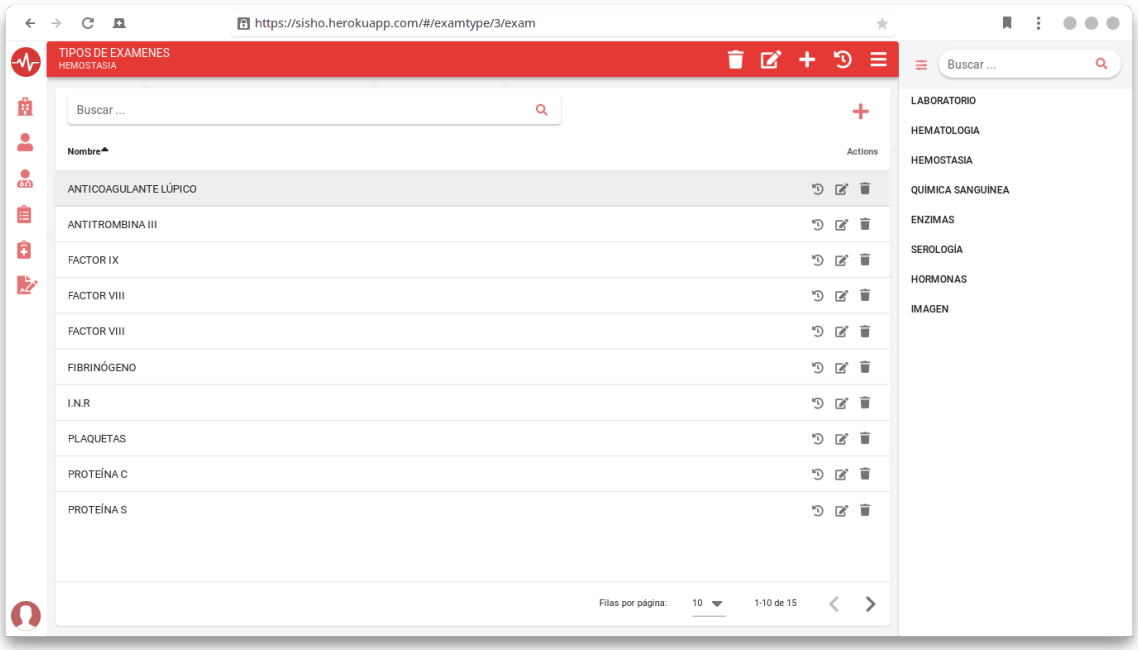
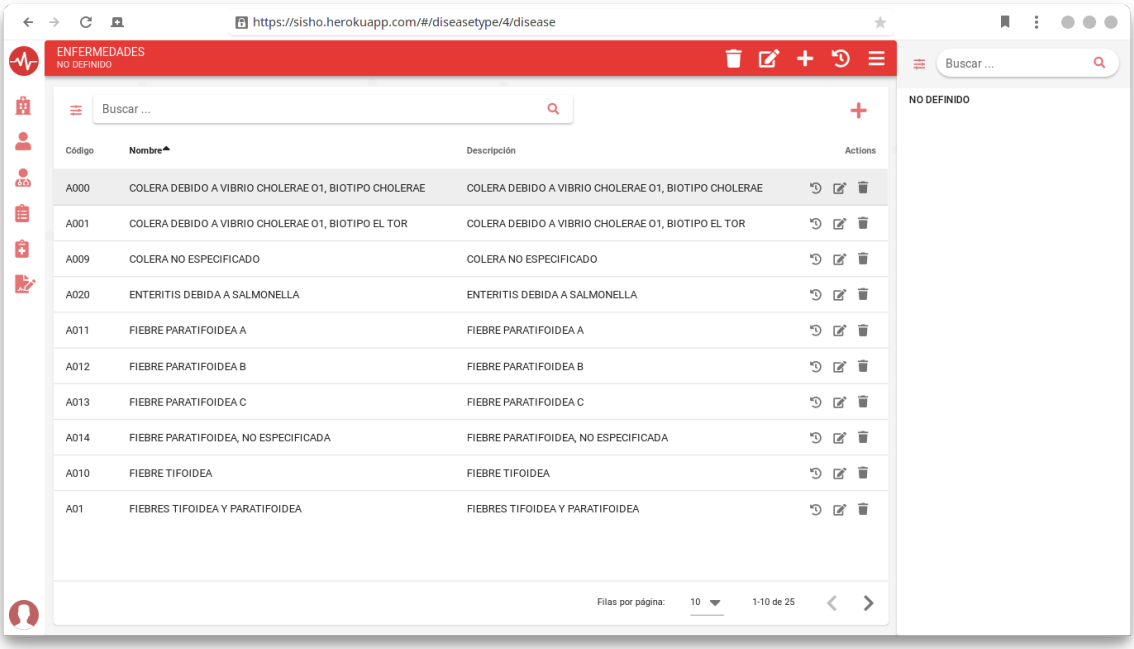


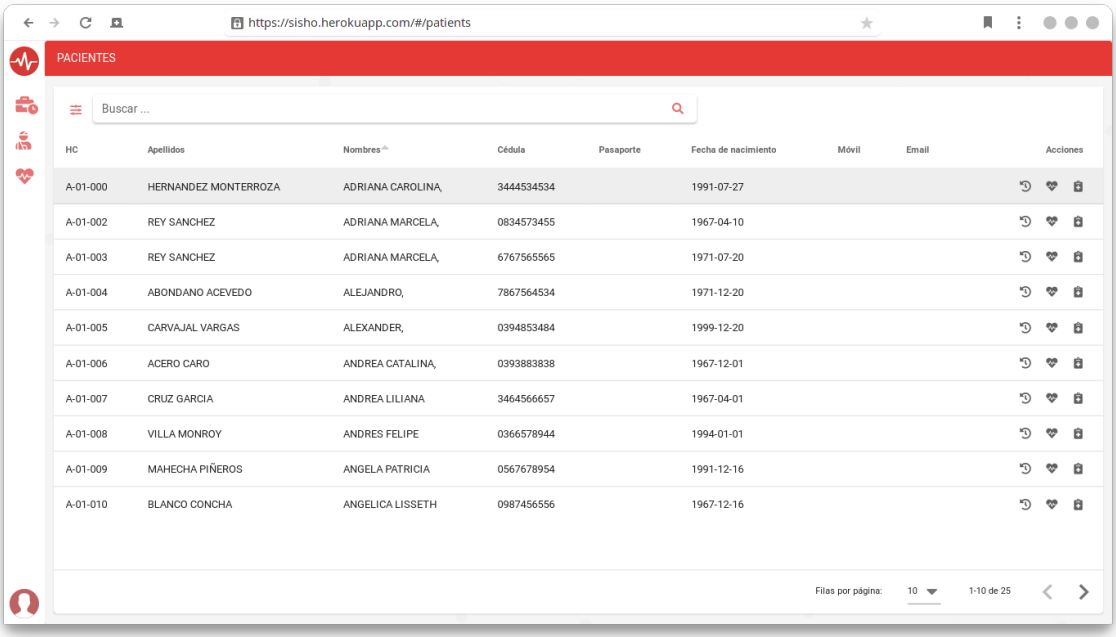
Figura 32: Página de registro de enfermedades para el diagnóstico de un paciente.



5.6. Aplicación web SPA para el rol de analista

El usuario con rol de analista posee la tarea de registrar y actualizar la información del paciente.

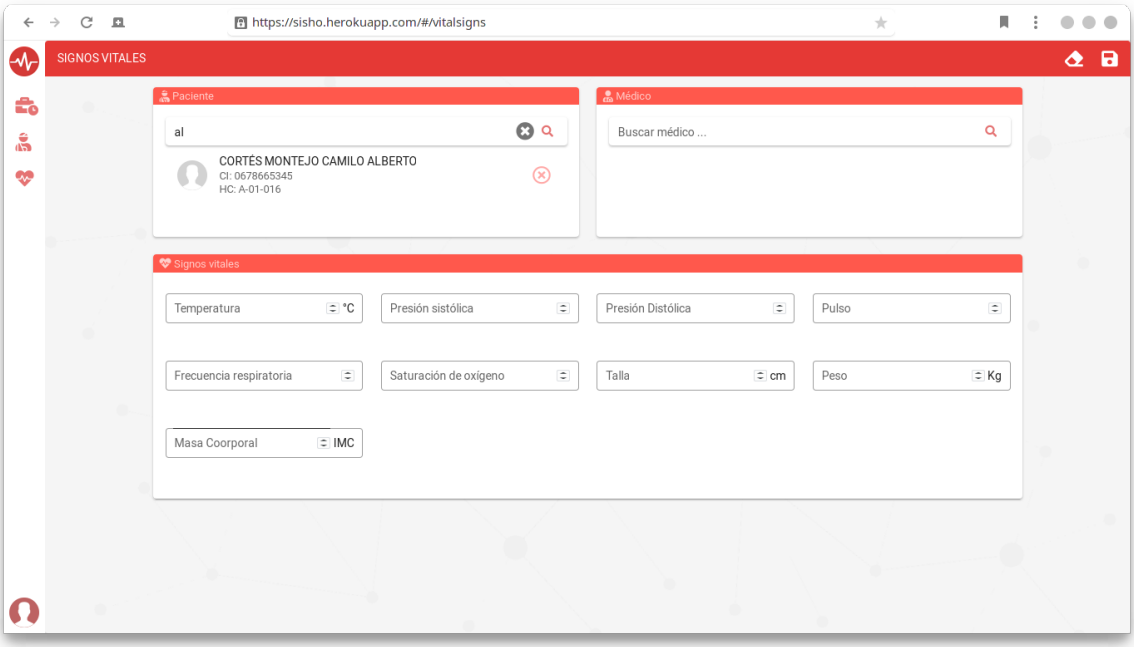
Figura 33: Página de registro de los pacientes.



5.7. Aplicación web SPA para el rol de enfermería

El usuario con rol de enfermería posee la tarea de registrar los signos vitales del paciente.

Figura 34: Página de registro de los signos vitales.



5.8. Aplicación web SPA para el rol de médico

El médico posee acceso a la informacion de los pacientes y al registro de la historia clinica.

Figura 35: Mediante una serie de pasos obligatorios el médico puede registrar la ficha del paciente.

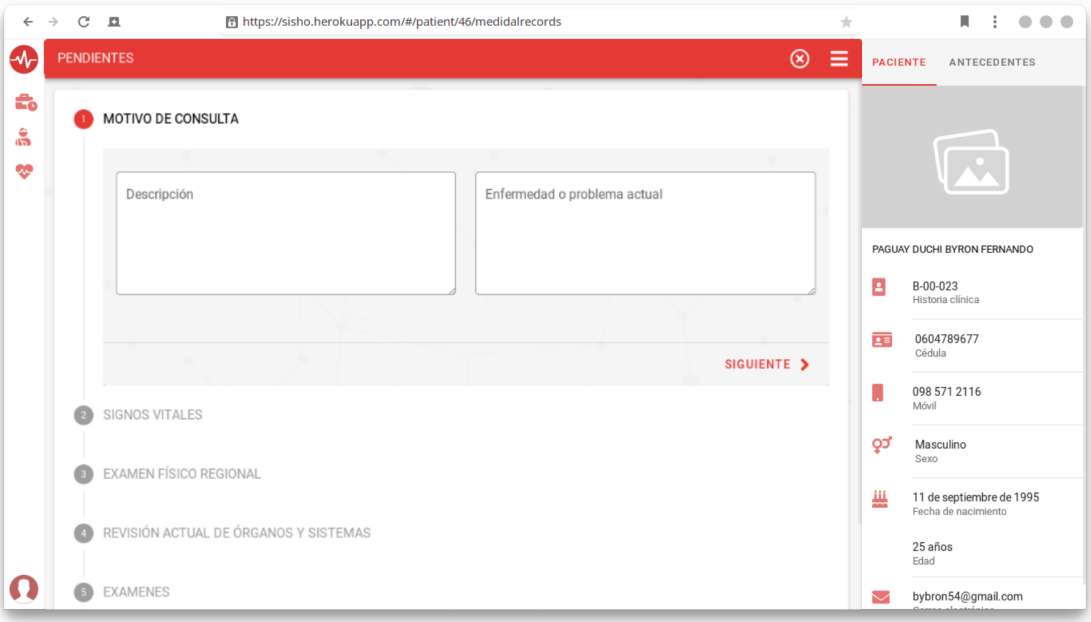


Figura 36: Página que muestar la información de una ficha médica.

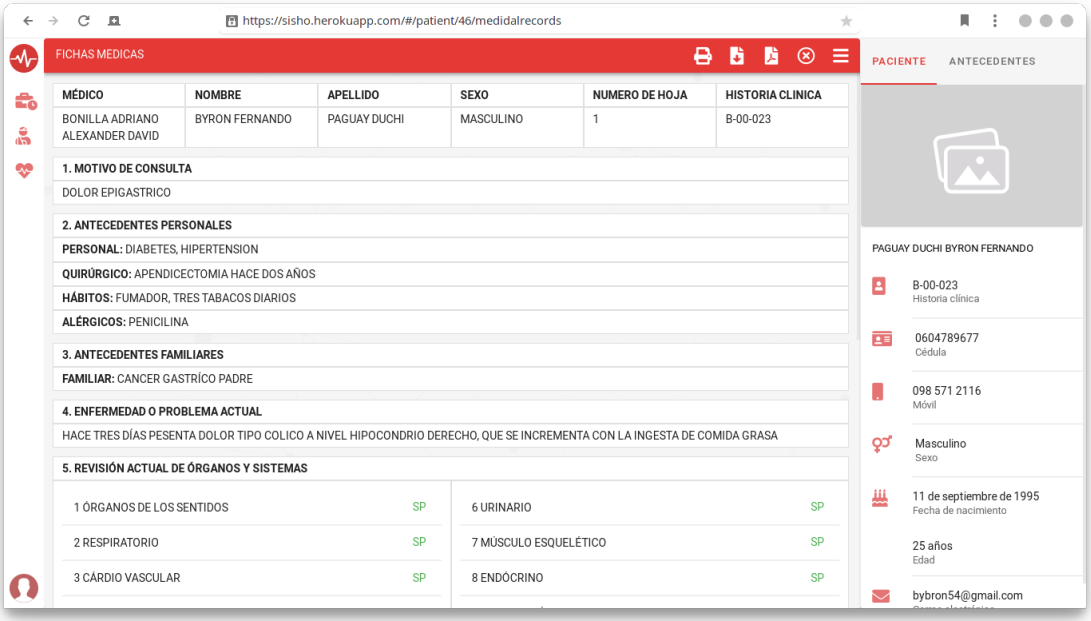
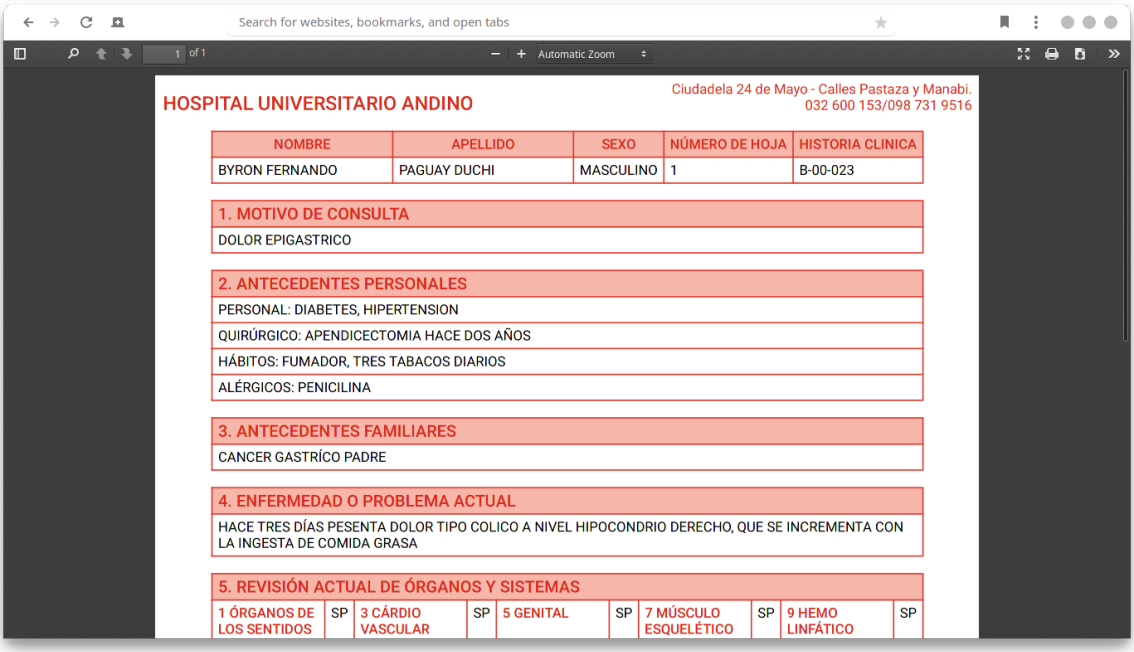


Figura 37: El sistema puede generar un reporte de la ficha médica del paciente.



HOSPITAL UNIVERSITARIO ANDINO

Ciudadela 24 de Mayo - Calles Pastaza y Manabi.
032 600 153/098 731 9516

NOMBRE	APELLIDO	SEXO	NÚMERO DE HOJA	HISTORIA CLINICA
BYRON FERNANDO	PAGUAY DUCHI	MASCULINO	1	B-00-023

1. MOTIVO DE CONSULTA

DOLOR EPIGASTRICO

2. ANTECEDENTES PERSONALES

PERSONAL: DIABETES, HIPERTENSION

QUIRÚRGICO: APENDICECTOMIA HACE DOS AÑOS

HÁBITOS: FUMADOR, TRES TABACOS DIARIOS

ALÉRGICOS: PENICILINA

3. ANTECEDENTES FAMILIARES

CANCER GASTRÍCO PADRE

4. ENFERMEDAD O PROBLEMA ACTUAL

HACE TRES DÍAS PESENTA DOLOR TIPO COLICO A NIVEL HIPOCONDRIO DERECHO, QUE SE INCREMENTA CON LA INGESTA DE COMIDA GRASA

5. REVISIÓN ACTUAL DE ÓRGANOS Y SISTEMAS

1 ÓRGANOS DE LOS SENTIDOS	SP	3 CÁRDIO VASCULAR	SP	5 GENITAL	SP	7 MÚSCULO ESQUELÉTICO	SP	9 HEMO LINFÁTICO	SP

5.9. Pruebas de rendimiento con Apache JMeter

Para las pruebas de rendimiento se procedió con la preparación de los datos en formato csv, configurarlos y utilizarlos con Apache JMeter.

Figura 38: Preparación de datos para las pruebas de rendimiento.

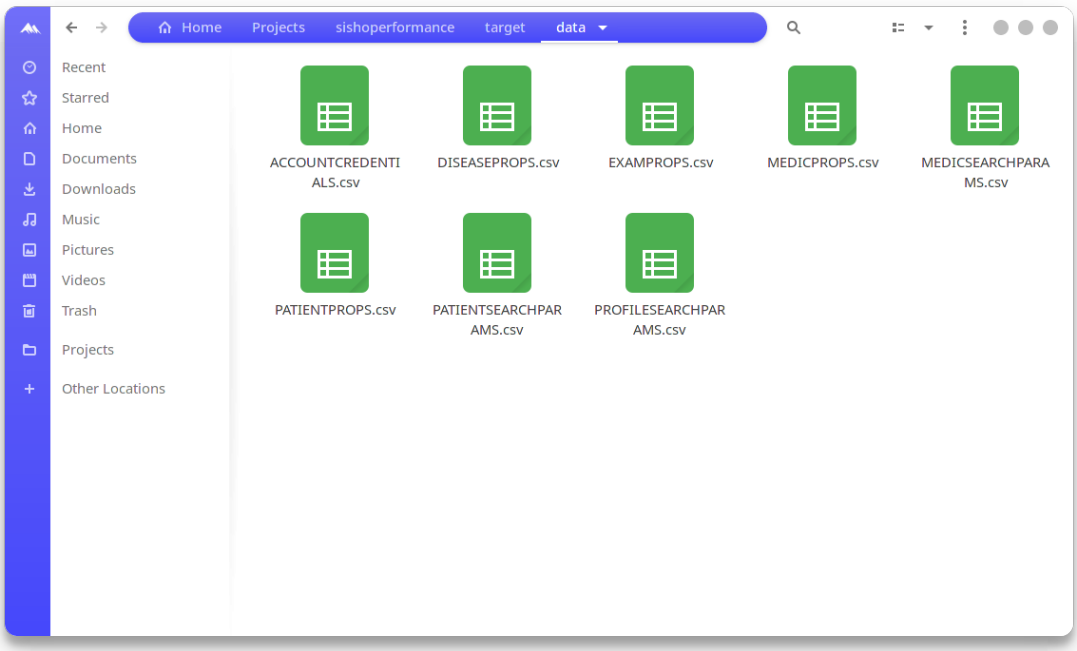


Figura 39: Rendimiento del acceso a la cuenta.

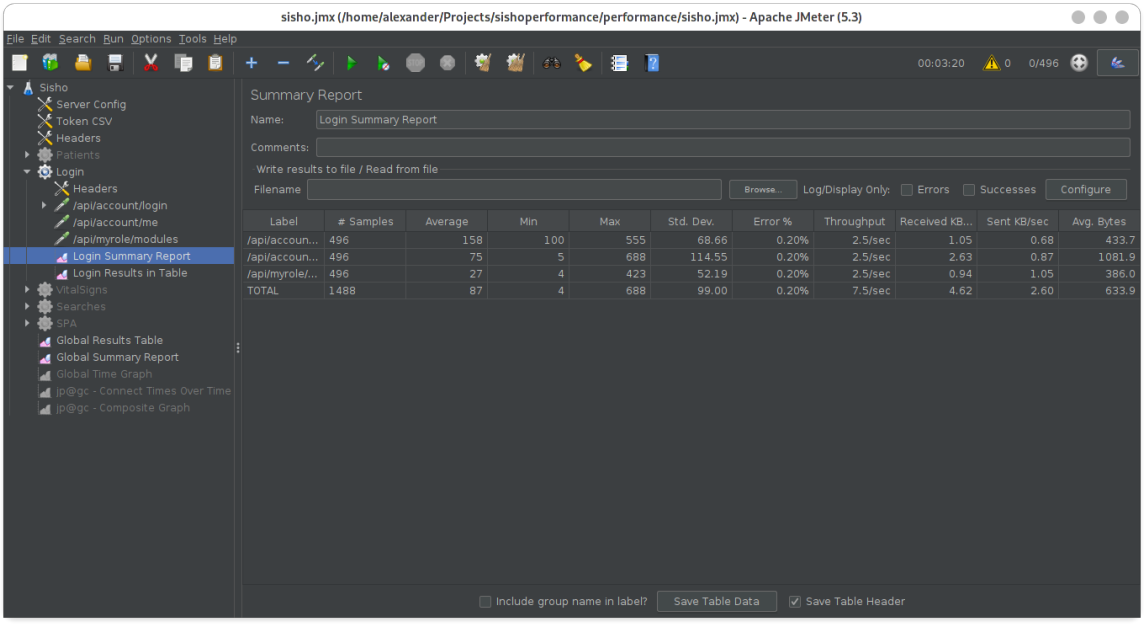


Figura 40: Rendimiento del modulo de pacientes.

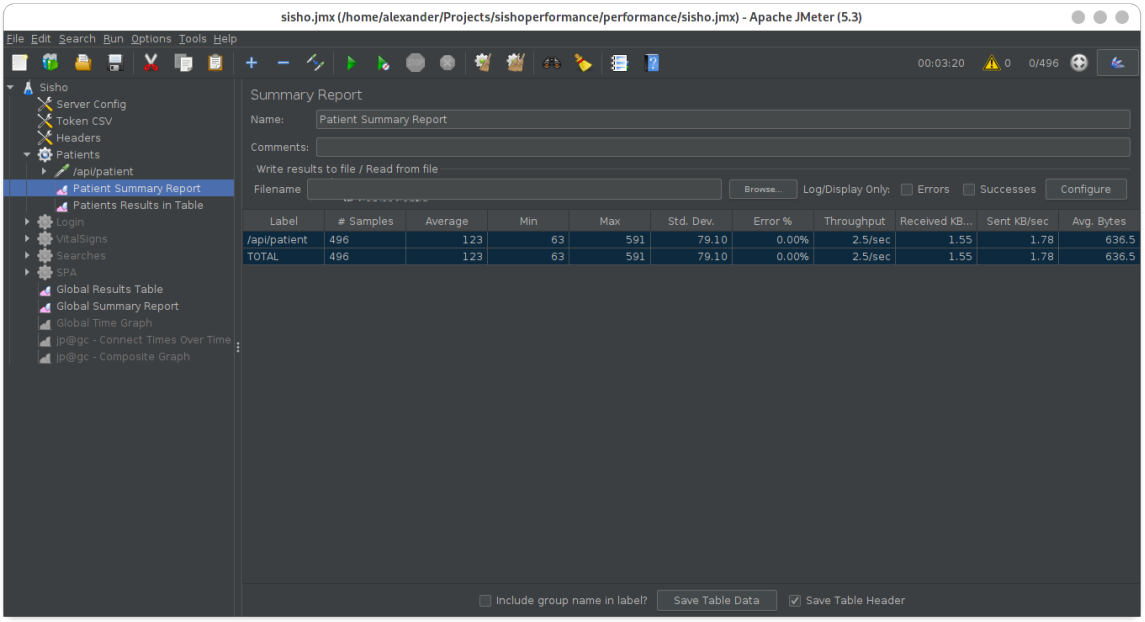


Figura 41: Rendimiento de los algoritmos de busqueda.

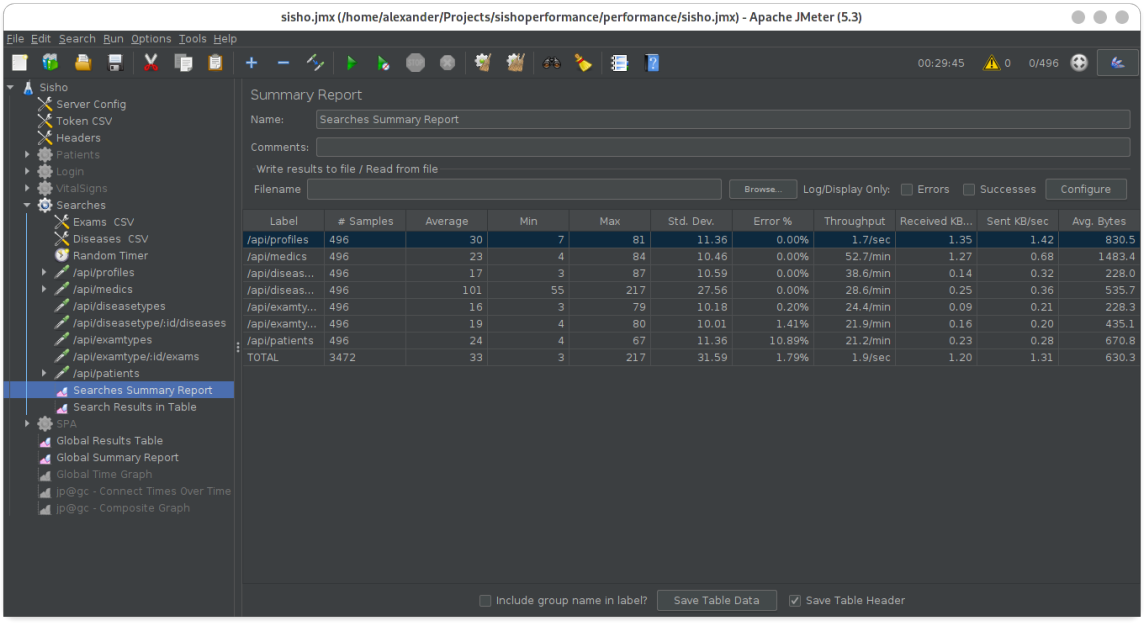


Figura 42: Rendimiento del módulo de la ficha médica.

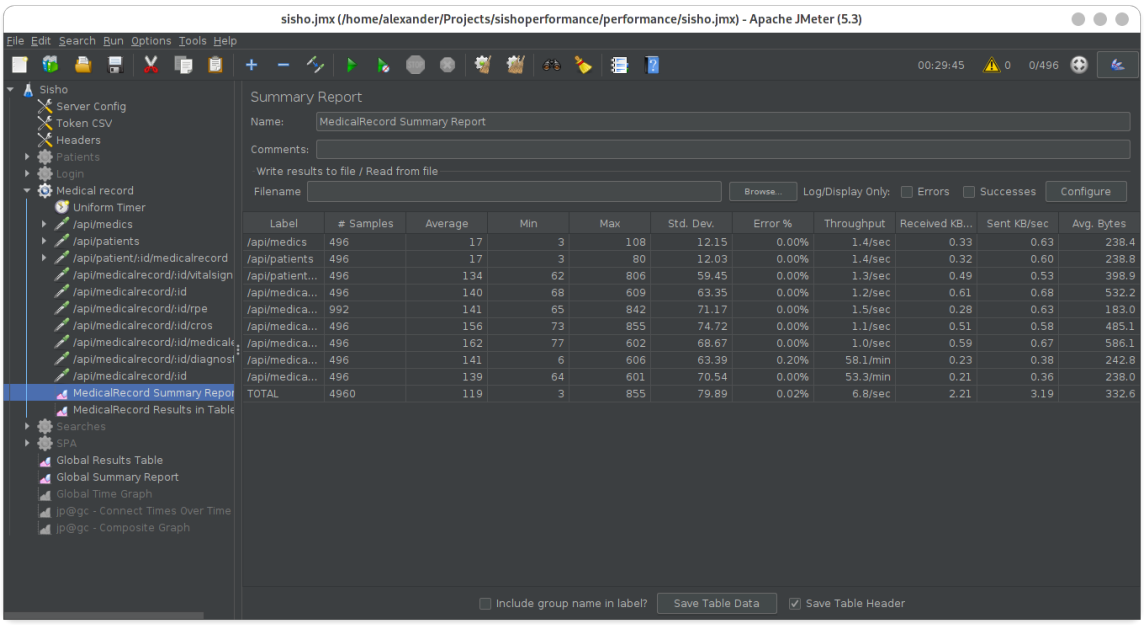
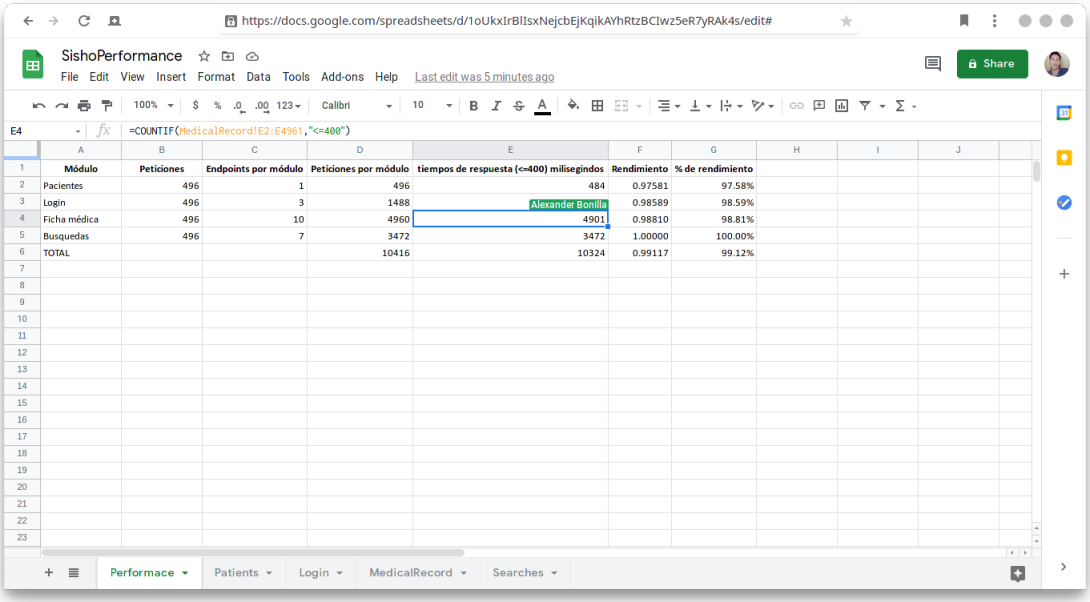


Figura 43: Resultados de rendimiento del servidor y las fórmulas para contar el tiempo de respuesta menor de 400 milisegundos.



Módulo	Peticiones	Endpoints por módulo	Peticiones por módulo	tiempos de respuesta (<=400) milisegindos	Rendimiento	% de rendimiento
Pacientes	496	1	496	484	0.97581	97.58%
Login	496	3	1488	1452	0.98589	98.59%
Ficha médica	496	10	4960	4901	0.98810	98.81%
Busquedas	496	7	3472	3472	1.00000	100.00%
TOTAL			10416	10324	0.99117	99.12%

5.10. Repositorio de código fuente

El código fuente se almacena en un repositorio de GitHub, desde donde se puede descargar e implementarlo.

Figura 44: Código almacenado en GitHub.