9 No.1 (2025): Journal Scientific MQRInvestigar ISSN: 2588–0659
https://doi.org/10.56048/MQR20225.9.1.2025.e184

Backend and architecture development for patient management at UDIPSAI

Desarrollo de backend y arquitectura para gestión de pacientes en UDIPSAI

Autores:

Mora-Cabrera, David Esteban
UNIVERSIDAD CATÓLICA DE CUENCA
Egresado de la Unidad Académica de Informática Ciencias de la Computación, e Innovación Tecnológica
Cuenca-Ecuador

david.mora.32@est.ucacue.edu.ec

https://orcid.org/0009-0009-1270-8907

Sañay-Sañay, Segundo Isael
UNIVERSIDAD CATÓLICA DE CUENCA
Docente Tutor del área
Cuenca – Ecuador

ssanay@ucacue.edu.ec

https://orcid.org/0000-0003-4661-318X

Fechas de recepción: 18-ENE-2025 aceptación: 18-FEB-2025 publicación: 15-MAR-2025

https://orcid.org/0000-0002-8695-5005
http://mqrinvestigar.com/

Resumen

La era digital ha transformado y evolucionado procesos de organizaciones públicas y privadas, existen entidades que no cuentan con muchos de estos beneficios, lo que dificulta el manejo de datos; comprometiendo su eficiencia. Ante esta problemática, el proyecto orienta la automatización de los procesos en UDIPSAI para la gestión de pacientes en un sistema que optimiza el flujo de trabajo y garantiza la seguridad de sus datos.

En el proyecto se aplicó el marco de trabajo ágil Scrum que permitió entregas incrementales como reuniones constantes con las partes interesadas. En cuanto a la codificación del backend se usó el lenguaje de programación Java con el framework Spring Boot teniendo como base el patrón de diseño Modelo - Vista - Controlador + Servicio (MVC+S), y con una arquitectura cliente - servidor.

Por otro lado, la elicitación de requisitos se realizó con el estándar de calidad FURPS+. Mientras que las pruebas del producto se realizaron con Postman y herramientas de navegadores. Finalmente, el despliegue de la aplicación se realizó en linux Ubuntu Server en el que se levantó el servidor web Apache Tomcat.

La automatización de este proceso se vio reflejado en el flujo de trabajo administrativo, mejorando la integridad de la información sensible, así como la satisfacción del 76.67%, 86.6%, 88.8% en seguridad, eficiencia y experiencia por parte de sus especialistas.

Palabras clave: Automatización de procesos; Seguridad de datos; Despliegue en servidores Linux; Experiencia del usuario (UX), MVC+S

Abstract

The digital era transformed the public and private organizations, there are entities that doesn't count with some of these benefits, which make the use of the data more difficult and less efficient. With this problem, this project gives orientation for the process on UDIPSAI for the pacientes on a sistem tha optimize the work and gives a security guarantee for the database.

This project applied the Scrum capable work that allowed the delivering of consistent reunions with the interested parts. About the backend codification it used Java program with Spring Boot framework with the design of Model- Sight- control+ service (MVC+S) and with a client- server architecture. In another ways the elicit of requirements were made with the FURPS+ quality. Meanwhile the product evidence were realized on Postman and navigation tools. Finally the app deployment were realized on Linux Ubuntu Server in which the Apache Tomcat server were installed.

The automatization of these process were reflected on the administrative workflow, improving the sensible information integrity, as the satisfaction of the 76.67%, 86.8%, 88.8% on security, efficiency and experience of the specialist.

Keywords: Process automation; database security; Linux server deployment; user experience (UX) MVC+S

Introducción

La coordinación eficiente de la información es un desafío vital para las instituciones que trabajan con personas vulnerables como es el caso de la UDIPSAI, que ofrece servicios especializados a más de 4,250 pacientes, los cuales en su mayoría representan niños e igualmente adolescentes con necesidades educativas especiales y sensibles. Su labor compromete áreas como la psicología educativa, psicología clínica, terapia de lenguaje, odontología y trabajo social, teniendo como objetivo principal promover la construcción integral de cada uno de sus pacientes.

A pesar del impacto positivo de la UDIPSAI la dependencia de archivos físicos, así como fichas individuales generan inconvenientes como la duplicidad de información, retrasos a nivel operacional y en algunos casos la revictimización de pacientes al solicitar de manera repetida información sensible.

La implementación de un sistema de gestión de pacientes en la UDIPSAI responde principalmente a sus problemas a nivel operacional, ya que, al momento no cuentan con un sistema sólido para el correcto manejo de sus datos lo cual desencadena en duplicidad de datos, falta de integridad de la información, fallas a nivel operacional y mala experiencia para pacientes. La justificación de este desarrollo se fundamenta en 3 principales aspectos que se desglosan a continuación.

Primero, a nivel operacional el cambio de un proceso que es completamente manual por uno digital representa una mejora significativa en cuanto al tiempo empleado para realizar tareas cotidianas, como la gestión de pacientes y de especialistas permitiendo así la eliminación de documentación física con difícil acceso.

Segundo, la Seguridad de la información, así como su privacidad es un tema muy delicado, ya que en muchos casos se trata de menores de edad cuya información resulta ser bastante sensible, y al contar con un sistema físico de estos registros toda esta información es comprometida.

Tercero, la experiencia de los usuarios de UDIPSAI puede ser negativo por el conflicto principal es la revictimización de los pacientes que manejan información sensible al estar constantemente experimentando la repetición constante de dicha información, para ello el uso del sistema propuesto al tener la información de cada uno de los pacientes opta por mejorar la calidad de la atención en UDIPSAI, además de aumentar considerablemente la comodidad y la confianza de sus pacientes.

El proyecto se centra en el desarrollo del backend con una arquitectura sólida para el producto de software de la UDIPSAI, que asegure un manejo centralizado y seguro de la información, optimice la calidad del servicio y reduzca la revictimización de los pacientes. Entre los objetivos específicos ha sido identificar los requisitos funcionales y no funcionales del backend que permitan la integración y eficiencia en el Sistema de Gestión de pacientes de UDIPSAI. Implementar una arquitectura escalable utilizando Spring Boot y el patrón MVC+S para procesar datos con alta seguridad y rendimiento. Desarrollar servicios RESTful que faciliten el registro, consulta y actualización de información por parte de los especialistas, garantizando la integridad y privacidad de los datos del paciente.

Estado del arte

¿Qué es la discapacidad?

Según menciona (Seoane, 2010), en su artículo “QUÉ ES UNA PERSONA CON DISCAPACIDAD” la discapacidad de manera conceptual y conforme a criterios científicos se define como una problemática en la salud causada por una deficiencia de manera individual, teniendo como punto de partida la biología.

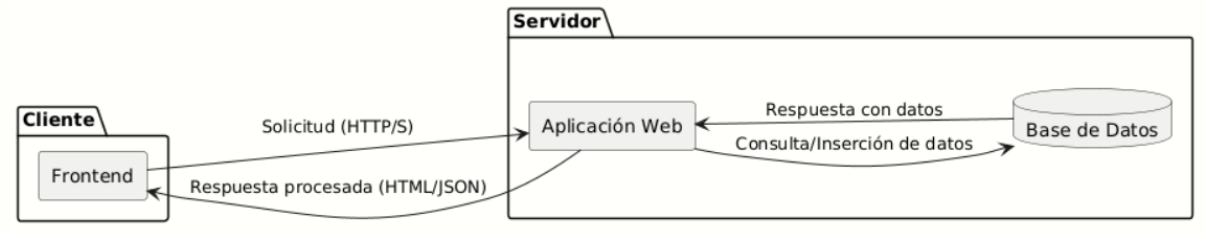
Arquitectura Cliente-Servidor

La importancia de la selección de una arquitectura de software representa un valor clave en cualquier proyecto de software ya que como menciona (Agudelo et al., 2021) “Una arquitectura de software correctamente diseñada es clave para que las organizaciones puedan avanzar y concentrarse en su función misional”.

La arquitectura cliente-servidor cuenta con dos componentes claves, el servidor que utiliza los recursos para gestionar las peticiones de los usuarios permitiendo una conectividad simultanea sin conflictos, y el cliente solicita esas peticiones(Lituma-Sarmiento & Vizñay-Durán, 2023).

Figura 1

Diagrama de interacción entre cliente-servidor.- El cliente representa al usuario que interactúa con el servidor por medio de un navegador al realizar una solicitud HTTP, el servidor procesa la lógica de la solicitud del cliente y retorna una respuesta a dicha solicitud.



Fuente: Elaboración propia

La división de la arquitectura cliente-servidor (Figura 1) es crucial pues facilita un incremento en la eficiencia, dado que cada uno de sus elementos tiene una responsabilidad única, promoviendo de esta manera la mantenibilidad y la escalabilidad (Moyano et al., 2020).

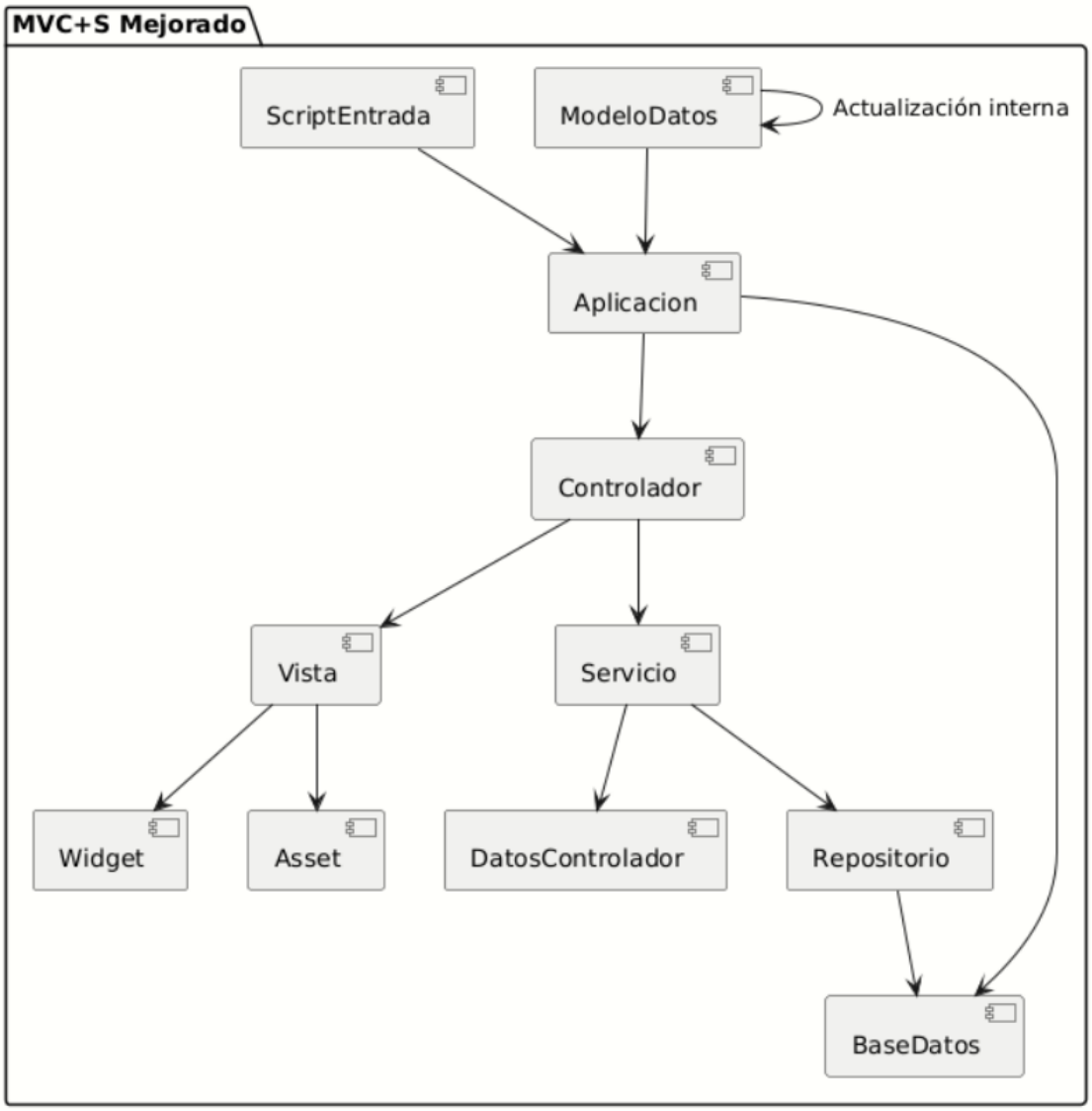
MVC+S

El patrón de diseño MVC+S es una extensión del modelo MVC tradicional utilizado comúnmente para la separación de responsabilidades, en este modelo existen 4 capas: Modelo para la gestión de base de datos, Servicio para la lógica de negocios, Controlador que actúa como intermediario entre backend y frontend y la vista para la presentación de la información a los usuarios, esta estructura modular fomenta la mantenibilidad y escalabilidad del producto de software (Sañay et al., 2019).

Figura 2

Estructura MVC+S.- Representación del flujo de trabajo que se realiza en MVC+S (Sañay et al., 2019), en donde el Modelo se encarga de estructurar los datos de la aplicación, el

controlador actúa como un intermediario entre el modelo y la vista, la vista representa la interfaz gráfica y la extensión de servicio que contiene la lógica de negocio.



Fuente: Elaboración Propia

Desarrollo Backend

El desarrollo backend, está enfocado a la implementación de la lógica de un servidor, la gestión de la base de datos, así como su conexión con el frontend integrando así una aplicación completa. según (Sañay et al., 2019), la función del backend es la de procesamiento de solicitudes por parte de los clientes o usuarios, así como devolver las respuestas correctas a dichas solicitudes todo esto ejecutando la lógica de negocio.

Servicios RESTful

Los Servicios RESTful permiten la comunicación entre sistemas ya que define un conjunto de restricciones para crear servicios web los cuales interactúan con recursos por medio de URLs y HTTP (GET, POST, PUT, DELETE) (Gómez et al., 2020). En la actualidad los servicios RESTful son altamente usados debido a su eficiencia, algunos frameworks como Spring Boot implementan estos servicios debido a sus altas capacidades en el manejo de solicitudes HTTP (Liu et al., 2022).

Herramientas de desarrollo

Spring Boot simplifica la creación de aplicaciones corporativas sólidas y escalables así lo menciona (Gómez et al., 2020), facilita a los programadores la creación rápida de una aplicación a través de la disminución de las alteraciones manuales y la incorporación automática de las dependencias requeridas.

Spring Security forma parte del ecosistema Spring, diseñado con el objetivo de proporcionar filtros de seguridad como autenticación, autorización, y protección contra brechas en la seguridad, es comúnmente usado en aplicaciones web con servicios RESTful (Joshi, 2024). Spring Security no forma parte directa del framework Spring Boot, pero este último facilita su integración.

MySQL es un sistema gestor de bases de datos relacionales, comúnmente utilizado en la elaboración de aplicaciones web. (León Soberón, 2020), MySQL sobresale por su confiabilidad, rendimiento y escalabilidad, lo que lo convierte en la opción ideal para gestionar grandes cantidades de datos.

Nginx en entornos de producción es una herramienta sólida para las aplicaciones frontend. al actuar como un servidor web permite almacenar aplicaciones desarrolladas en frameworks como React, Angular optimizando sus tiempos en carga y recursos. (Kithulwatta et al., 2022).

Apache Tomcat apoya Java Servlet, JavaServer Pages (JSP) y Java (EL) en su lenguaje de expresión. De acuerdo con (*Spring Boot :: Spring Boot*, s. f.), Tomcat funciona como el contenedor de servlets predefinido para aplicaciones creadas con Spring Boot, lo que simplifica la implementación y ejecución de aplicaciones web fundamentadas en Java.

La herramienta Cloudflare Tunnel forma parte del ecosistema de Cloudflare Zero Trust, establece conexiones seguras entre los usuarios y el servidor evitando su exposición directa en internet, para ello crea un canal cifrado entre los servidores internos y la red global de Cloudflare (Mohammady, 2024).

Metodología

Metodología de la investigación

El enfoque aplicado al producto de software se fundamentó en dos metodologías: el modelo FURPS+ para la calidad y análisis de requerimientos, y el marco de trabajo ágil SCRUM para la gestión del proceso de desarrollo.

FURPS+

Es un estándar que evalúa la calidad del software en base a sus criterios de: Funcionalidad, Usabilidad, Fiabilidad, Rendimiento, Mantenibilidad y el "+" representa factores adicionales. Su uso demuestra la importancia del análisis de los requerimientos funcionales y no funcionales para obtener una visión completa del software garantizando su éxito (Puspita et al., 2024). En este sentido (Sandoval & Ismael, 2022), en su trabajo mencionan el uso de la metodología FRUPS para determinar si su producto final cumple con estándares de calidad.

SCRUM

Marco de trabajo ágil SCRUM diseñado para solventar problemas complejos permitiendo la organización del equipo en base a sus necesidades. Está enfocado principalmente a la entrega de valor de manera iterativa e incremental, es popular ya que permite una adaptación rápida a los requisitos cambiantes y una mejora en la colaboración de los equipos (Sutherland, 2020).

SCRUM, resulta ser adecuado para proyectos web los cuales están en constante adaptación al cambio, en comparación con otras metodologías como XP (Bautista-Villegas, 2022).

Análisis de requisitos.

La ingeniería de requisitos y su análisis conforma el punto de partida crítico del proceso de desarrollo del software, pues se definen las funcionalidades, reglas de negocio que el producto final debe cumplir para abarcar todas las necesidades del usuario u organización [Primeras iteraciones de SCRUM].

La recopilación de los requisitos se realizó por medio de reuniones semanales con las partes interesadas [stakeholders] permitiendo identificar de manera concreta las necesidades dentro de UDIPSAI, en base a la toma de requisitos se realizó el análisis con el uso del estándar FURPS+ asegurando la calidad y satisfacción de los usuarios sobre el producto final.

Considerando el enfoque principal del sistema, tras aplicar la metodología FURPS+ se determinó que los requisitos no funcionales (Anexo 1) representan un impacto alto para la construcción del backend y la arquitectura, en donde se priorizan aspectos como la seguridad, rendimiento, escalabilidad y mantenibilidad.

Desarrollo de la Arquitectura y Backend con Scrum.

La construcción del backend y la arquitectura del producto de software, se ejecutó en 10 Sprints de dos semanas [5 meses]. La Tabla 1. presenta la segmentación de las actividades a lo largo de los Sprints:

Tabla 1

Tareas de los sprint.- Representación de las tareas realizadas en cada Sprint a lo largo de todo el proceso de desarrollo, enfocados en cumplir con los requisitos no funcionales (Anexo 1), incluyendo también temas como la capacitación y despliegue en un entorno de producción para el software final.

	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5	Sprint 6	Sprint 7	Sprint 8	Sprint 9	Sprint 10
Historias de Usuario	X									

Diseño inicial de arquitectura	X	X								
Implementación del backend y frontend iniciales			X							
Integración de servicios básicos (API, DB)				X						
Estructuración y modularización del backend (Modulos Pacientes, Fichas)					X					
Desarrollo de autentificación y autorizaciones						X				
Incorporación de fichas especializadas							X			
Optimización de la arquitectura para escalabilidad								X		
Pruebas y validación de la arquitectura									X	
Creación de material audiovisual									X	
Despliegue de arquitectura										X
Capacitación										X

Fuente: Elaboración propia

En el diseño de la arquitectura, se planteó como estructura básica el patrón arquitectural <<cliente – servidor>> que se observa la figura 1, realizada en el primer y segundo Sprint, hasta llegar a una solución refinada y completa que se observa en la figura 4, la cual fue escalando a mayor nivel en función de las reuniones y análisis realizado con los directivos de la institución.

En los sprint 3 y 4, con el fin de lograr el diseño planteado en la figura 4, se realizó un análisis mediante entrevistas y reuniones [stakeholders de la UDIPSAI], en donde se concluyó la

necesidad de estructurar el backend bajo un esquema modular basado en el uso de controladores [requerimientos no funcionales]; por la necesidad de gestionar diferentes tipos de solicitudes: servicios para implementar la lógica de negocio, repositorios para el manejo correcto y optimización de la información, un modelo para definir la estructura de las tablas dentro de la base de datos (...).

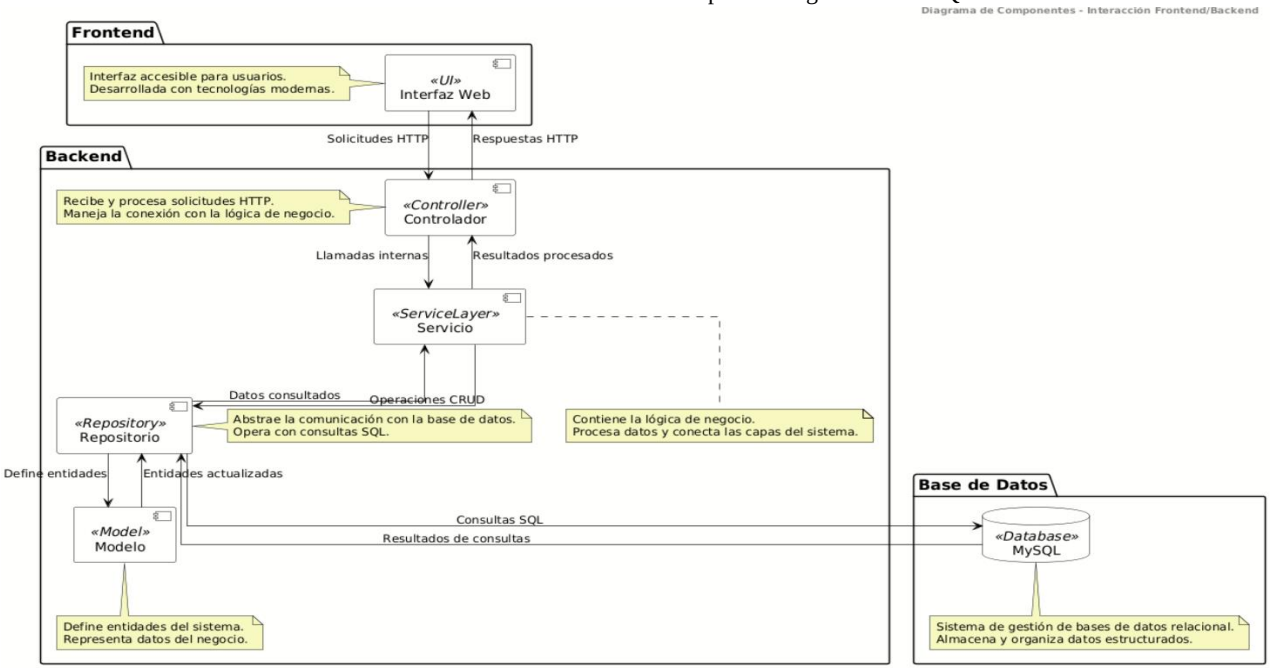
La construcción del código [backEnd], se optó por el uso del lenguaje Java con el framework Spring Boot debido a su popularidad, comunidad activa y compatibilidad con servicios RESTful, que simplifica el proceso de creación de servicios web escalables.

Por otro lado, para el manejo de la persistencia de los datos se utilizó MySql como motor de base de datos, el uso de esta herramienta se basó principalmente en el rendimiento en cuanto a peticiones CRUD al igual que la gestión y manejo de imágenes.

La arquitectura que se muestra en la figura 4, muestra la importancia de los paquetes: Controlador, Servicio, Repositorio y Modelo implementados, así como su interacción inicial con el frontend. Cada módulo tiene una separación con responsabilidades específicas.

Figura 3

Arquitectura cliente-servidor.- Con las primeras interacciones entre el frontend y el backend, en donde se muestra la estructura inicial de la arquitectura y cómo se interconectan sus módulos, representando su funcionalidad e incorporación con la base de datos.



Fuente: Elaboración Propia

La interacción entre los módulos [diagrama de secuencia que se muestra en la figura 5] inicia con el cliente [UI: interfaz de web o frontend] en donde, el usuario interactúa con el sistema mediante petición de servicios al backend por medio de solicitudes HTTP, el cual procesa y responde las solicitudes; para ello se realiza un proceso a través de diferentes módulos organizados en paquetes [Controlador, Servicio, Repositorio y Modelo], mismos que se describen a continuación:

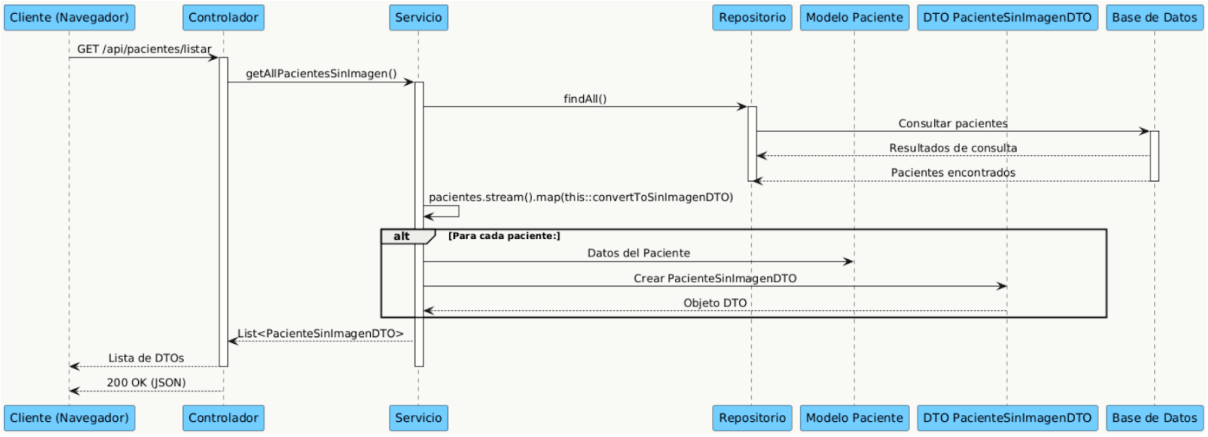
- **Controlador. -** Recepta las peticiones de los usuarios con solicitudes HTTP [GET, POST, PUT, DELETE], mediante endpoints RESTful realizando una validación de la información receptada antes de ser enviada a la lógica implementada en el Servicio. Finalmente se encarga de modificar las respuestas del servicio para enviarlas en un formato requerido por el usuario.
- **Servicio. -** Contiene la lógica del negocio, actuando como un agente conector entre el Controlador y el Repositorio, siendo su función procesar las solicitudes que envía

el Controlador y aplicar los métodos requeridos, con esto el Servicio realiza llamadas al Repositorio para poder acceder a los datos que se encuentran almacenados dentro de la base de datos y finalmente se envía la información hacia el Controlador.

- **Repositorio. –** Recibe la información del Servicio, y prepara las consultas a la base de datos sobre las tablas mediante métodos y consultas para, interactuar con las entidades, mejorando de esta forma el rendimiento del sistema.
- **Modelo. –** Aquí, se define la estructura de las tablas de la base de datos, para mapearlas directamente se usa anotaciones [@Entity, @Table y @Column] que son utilizadas por el Repositorio para construir o recuperar entidades.

Figura 4

El diagrama de secuencia inicial.- El cliente inicia el flujo al enviar una solicitud HTTP la cual pasa directamente al controlador que direcciona esta solicitud al servicio para procesar la lógica, realizando llamadas tanto al modelo como al repositorio para la validación de los datos.



Fuente: Elaboración propia

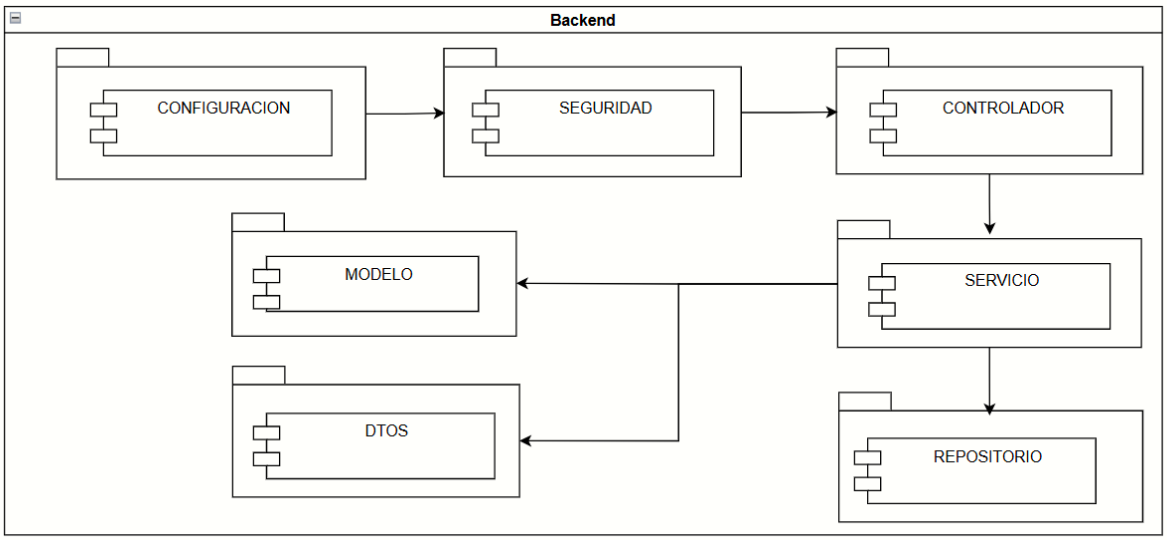
En el Sprint cinco que se muestra en la figura 5 y 6, se agregó el módulo DTO a la estructura inicial (figura 4.) mismo que transfiere los datos de las capas [Servicio, Repositorio] limitando la exposición de información sensible, incluye los atributos necesarios para ciertas

solicitudes, además; es utilizado como un convertidor de las entidades en objetos simplificados antes de ser enviados al usuario.

En el sprint seis, se avanza en la construcción de la arquitectura y el backend, aquí se implementó el módulo de autenticación de usuarios [JWT: JSON Web Token] asegurando un diseño de un flujo seguro de endpoints dentro del sistema, permitiendo que los usuarios [quienes dicen que son] accedan de manera correcta a los servicios del backend, además se agregan dos módulos adicionales [Configuración y Seguridad] que se muestra en la figura 6, concluyendo la estructura del backend [modelo arquitectural MVC+S de la figura 2 y 6], solventando la seguridad del producto de software.

Figura 5

Representación de la estructura modular del backend.- Donde se siguió el patrón de diseño MVC+S, añadiendo los módulos de configuración y seguridad para el manejo de autenticación asegurando un flujo seguro dentro del sistema.



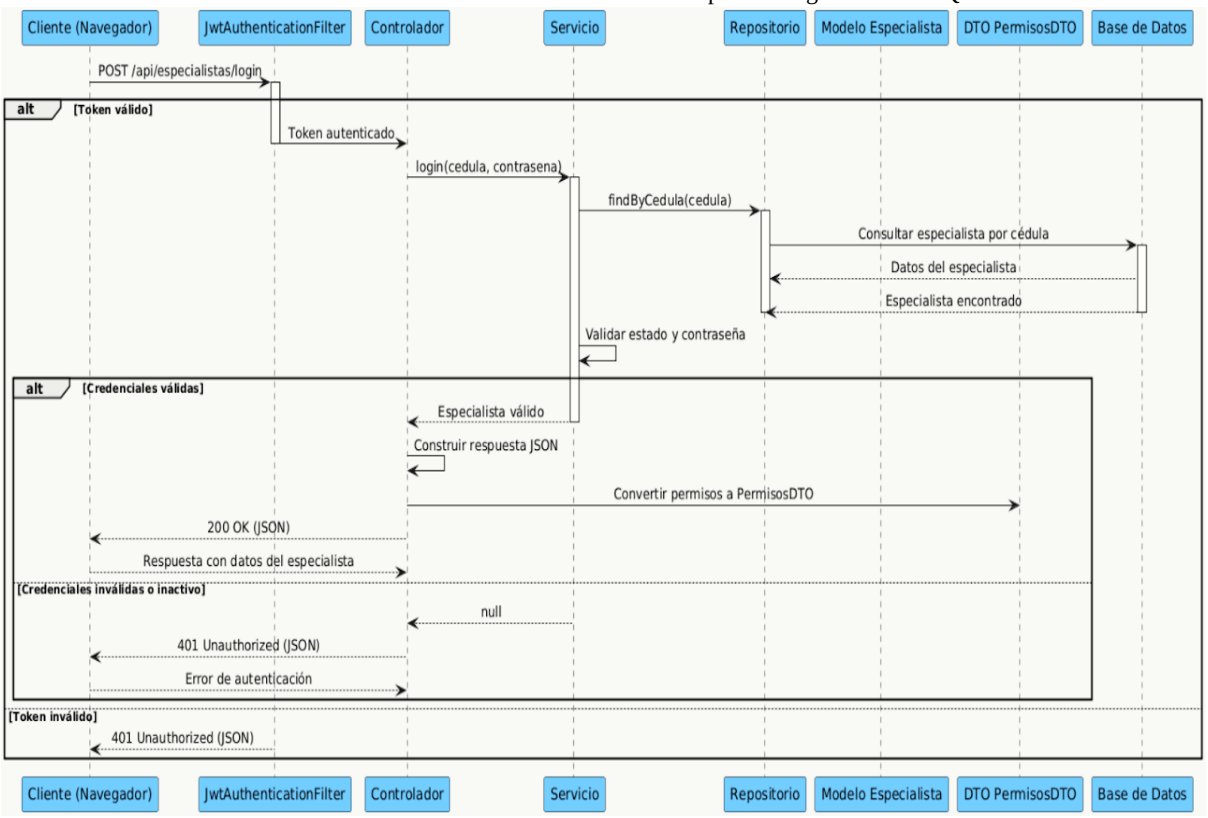
Fuente: Elaboración propia

En el diagrama de secuencia que se observa en la figura 7, se analiza el flujo de autenticación de usuarios, que comienza con el envío de credenciales mediante una solicitud HTTP al endpoint de login [Controlador], en secuencia el backend redirecciona la petición al filtro de autenticación [Seguridad] que se conecta con la base de datos [Repositorio] verificando si el usuario existe y que sus credenciales sean válidas, de ser así el filtro genera un token JWT [Seguridad] que contiene la información codificada del usuario y en caso de no ser válida la autenticación el sistema retorna una respuesta 401 Unauthorized con el mensaje de error.
En el módulo de Configuración se establecen las normas de seguridad generales dentro del sistema haciendo uso de Spring Security, siendo su función principal configurar las políticas de acceso al sistema para cada solicitud que llega al backend.

El módulo de Seguridad [filtro JWT] es el encargado de validar el token para cada solicitud al sistema, haciendo uso del encabezado definido, el cual lo compara con el valor recibido, si es válido crea un objeto con la autenticación con un usuario y rol, en caso de recibir un token invalido detiene solicitudes en curso y es invocado cada vez que se realiza una petición.

Figura 6

Diagrama de flujo del proceso de autenticación con JWT.- Usuario envía sus credenciales para solicitar la autenticación al servidor, el cual valida retornando una respuesta exitosa al usuario y en caso de no validar las credenciales muestra un error 401.



Fuente: Elaboración propia

Despliegue de arquitectura y backend

En el sprint 7 y 8 se realizó el refinamiento y optimización de la arquitectura, a continuación, se preparó un ambiente de producción [décimo Sprint] (figura 8) que sigue la arquitectura de despliegue del producto sobre un servidor con Ubuntu Server, en el cual se realizó la instalación y configuración de los siguientes servicios:

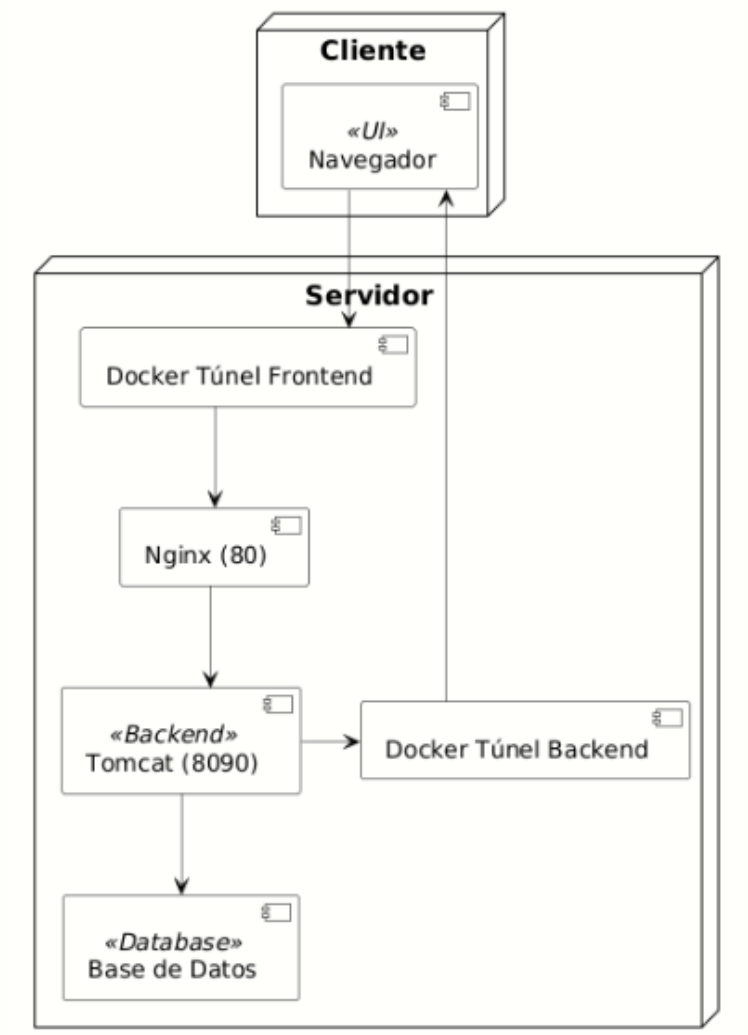
- **Docker Túnel de Cloudflare Frontend. -** Que actúa como un proxy entre el cliente y el servidor para la protección de los datos transmitidos, utilizado para gestionar el acceso al frontend.
- **Docker Tunel de Cloudflare Backend. –** Que se encarga de encapsular y administrar la comunicación del backend.

- **Servidor Nginx. -** Que gestiona y redirige las solicitudes HTTP para la comunicación con el backend.
- **Servidor Tomcat. -** Contiene el backend desarrollado en Java con Spring Boot manejando todas las peticiones.
- **Base de datos (MySQL). -** Almacena la información de UDIPSAI, es utilizado por el servidor Tomcat para el acceso a los datos y para la gestión de las entidades del sistema.

La arquitectura final que se muestra en la figura 8, está conformada por un dominio [udipsai/work] de acceso remoto a usuarios, los incrementos y mejoras aplicadas (Figura 8) en comparación con su diseño inicial (Figura 1) es una solución completa que satisface los requerimientos funcionales y no funcionales (Anexo 1) con la integración de la seguridad, modularidad, escalabilidad y mantenibilidad a lo largo del tiempo [FURPS+].

Figura 7

Diagrama de despliegue a nivel físico.- Representa cómo se realizó la construcción de la arquitectura dentro del servidor linux y cómo funciona la comunicación entre todos sus módulos para brindar una respuesta al usuario final.



Fuente: Elaboración propia

Resultados

Con la implementación e implantación del producto de software en UDIPSAI, los tiempos de operación disminuyeron.

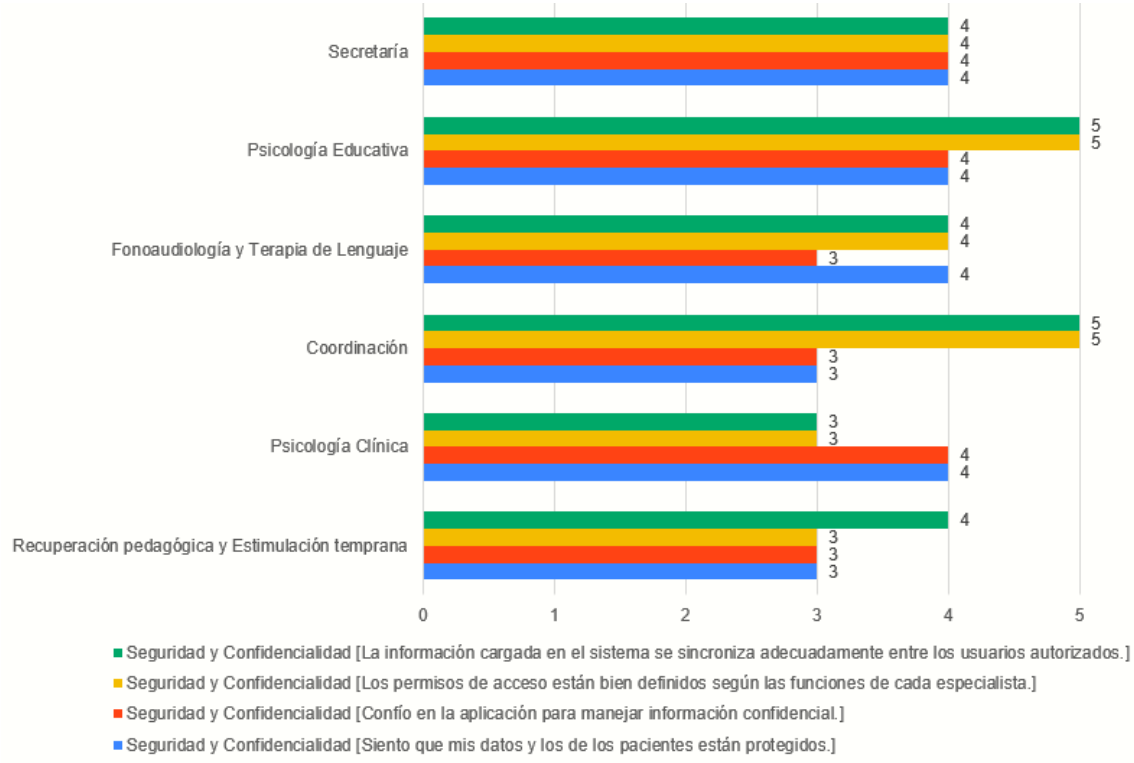
Para corroborar la eficiencia de la arquitectura se aplicó cuestionario (Campoverde-Calle et al., 2024) TAMAPP, posterior a la capacitación de stakeholders de UDIPSAI, con un enfoque por especialidades. La encuesta se valoró con preguntas en una escala de 1 a 5, donde 5 es completamente de acuerdo y 1 completamente en desacuerdo; con orientación a los niveles de: eficiencia, seguridad y confiabilidad, y colaboración y experiencia del usuario.

Seguridad

El promedio de satisfacción en cuanto a seguridad por parte de los especialistas es de 3.88, representando un porcentaje de aceptación del 76,67% (Figura 9).

Figura 8

Resultados de seguridad.- Segmentados por cada una de las especialidades y preguntas realizadas dentro de UDIPSAI, en donde se refleja que la mayor aceptación se encuentra en el área de Psicología Educativa.



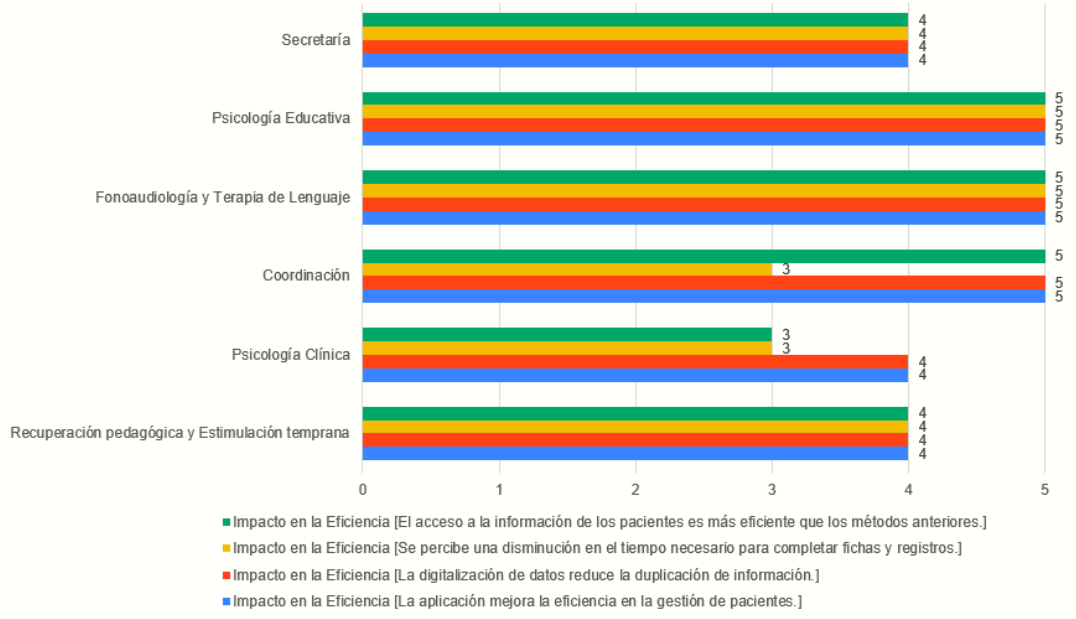
Fuente: Elaboración propia

Eficiencia

La eficiencia del sistema dio un promedio total de satisfacción por parte de los especialistas de 4.33, representando un porcentaje del 86.6% (Figura 10).

Figura 9

Resultados de eficiencia.- Segmentados por cada una de las especialidades y las preguntas realizadas dentro de UDIPSAI, en donde se refleja que la mayor aceptación se encuentra en las áreas de Psicología Educativa y Terapia y lenguaje.



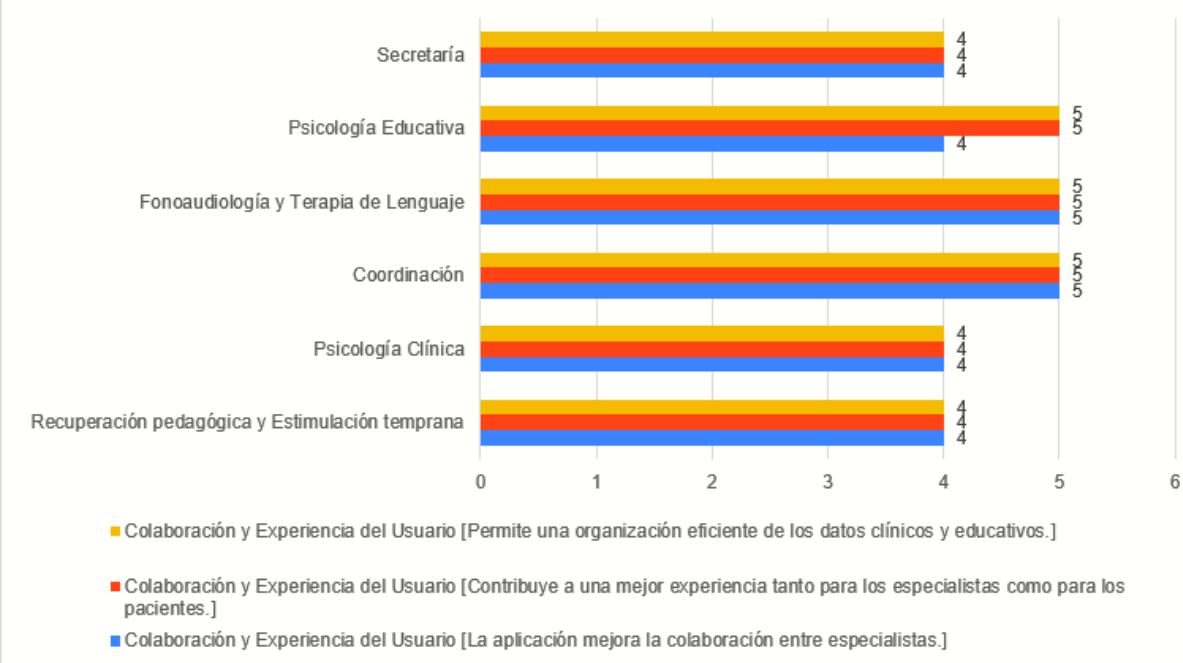
Fuente: Elaboración propia

Experiencia

En la figura 11 se muestra, la experiencia por parte de los usuarios con una satisfacción promedio de 4.44, representando un porcentaje de aceptación del 88.8%.

Figura 10

Resultados de experiencia.- Segmentados por cada una de las especialidades, y las preguntas realizadas dentro de UDIPSAI, en donde se refleja que la mayor aceptación se encuentra en las áreas de Coordinación y Terapia y lenguaje



Fuente: Elaboración propia

Discusión

El proceso de desarrollo del backend y arquitectura ha generado un impacto muy notorio en UDIPSAI, ya que, se obtuvo una optimización en el flujo de trabajo al automatizar sus procesos físicos. Los principales beneficios son la reducción en los tiempos de operación, seguridad y experiencia por parte de sus usuarios.

Seguridad y Calidad

La calidad del código fue uno de los retos principales dentro del proyecto, se obtuvo mediante buenas prácticas de programación, como el uso de un patrón de diseño MVC+S que permite la segmentación de responsabilidades, generando un código mantenible y escalable.

Las clases se construyeron con un único objetivo, alineado con el principio de responsabilidad única SRP (Single Responsibility Principle), por lo cual se mantiene una integridad y legibilidad del código. La implementación del manejo de errores [en los endpoints del producto] proporciona respuestas claras, con información relevante en caso de fallos, mejorando la experiencia del usuario y depuración del sistema.

El sistema se configuró con un dominio y un túnel Cloudflare que permiten un acceso protegido a la información. Cualquier interacción entre los clientes y el servidor se realiza por medio de HTTPS, lo cual previene los famosos ataques (man-in-the-middle).

Las peticiones al sistema cuentan con un filtro JWT que permite el acceso por medio de un token personalizado, el cual asigna un rol estático para el ingreso a los recursos.

Pruebas y Desempeño

La eficiencia y el desempeño del software se validó mediante la ejecución de pruebas; como servidor se usó una máquina local y finalmente para la información se usó una base de datos con valores ficticios pero representativos de escenarios reales.

Se realizaron pruebas unitarias para verificar el comportamiento individual de los módulos del sistema, como herramienta se utilizó Postman para la prueba de API.

La eficiencia y los tiempos de respuesta del sistema se evaluaron mediante el uso de herramientas incluidas en los navegadores, midiendo el desempeño luego de la petición realizada.

Las pruebas unitarias hicieron énfasis en la funcionalidad del sistema, cada caso de prueba lleva la nomenclatura TC que significa “Test Case” seguido del número en secuencia, a continuación, se detallan los resultados de las pruebas unitarias:

Tabla 2

Resumen de los resultados de las pruebas unitarias.- Se aplicaron con la herramienta Postman para validar la funcionalidad y calidad de las principales solicitudes del sistema, mostrando un resultado positivo en la mayoría de los casos.

Caso de Prueba	Descripción	Estado
TC001	Login de Usuario	Aprobado
TC002	Usuario correcto, contraseña incorrecta	Aprobado
TC003	Registro normal de paciente	Aprobado
TC004	Registro repetido de paciente	Aprobado
TC005	Registro de paciente con institución desconocida	Aprobado
TC006	Edición normal de paciente	Reprobado

TC007	Edición de paciente con ID desconocido	Reprobado
TC008	Eliminación de paciente con ID correcto	Aprobado
TC009	Eliminación de paciente inexistente	Aprobado
TC010	Registro normal de médico	Aprobado
TC011	Registro de médico con campo de especialidad erróneo	Aprobado
TC012	Edición normal de médico	Aprobado
TC013	Edición de médico con especialidad incorrecta	Aprobado
TC014	Eliminación normal de médico	Aprobado
TC015	Eliminación de médico desconocido	Aprobado
TC016	Registro normal de institución	Aprobado
TC017	Registro repetido de institución	Aprobado
TC018	Edición normal de institución	Aprobado
TC019	Edición de institución con ID erróneo	Aprobado
TC020	Eliminación normal de institución	Aprobado

Fuente: Elaboración propia

Las pruebas de rendimiento midieron los tiempos de respuesta para las tareas pesadas del sistema la a continuación muestra los resultados:

Tabla 3

Resultados de las pruebas de rendimiento.- Se realizó haciendo uso de las herramientas incluidas en los navegadores con el objetivo de medir los tiempos de respuesta, en donde se refleja un resultado positivo para cada prueba.

Pruebas de Tiempos de Respuesta		
Cargar Formulario de Registro de Pacientes	Tiempo de carga 4.74s.	Positivo
Cargar Formulario de Registro de Especialistas	Tiempo de carga 2.99s.	Positivo
Cargar Formulario de Registro de Instituciones	Tiempo de carga 3.04s.	Positivo

Fuente: Elaboración propia

El enfoque de la arquitectura que se implementó es distinto de otro tipo de soluciones como las monolíticas o sistemas menos seguros, principalmente por los beneficios como modularidad y seguridad, lo cual asegura un enfoque muy competente y preparado para enfrentar futuras mejoras.

Conclusiones

La coordinación eficiente de la información es un desafío vital para las instituciones que trabajan con personas vulnerables como es el caso de la UDIPSAI, que ofrece servicios especializados a más de 4,250 pacientes, los cuales en su mayoría representan niños e igualmente adolescentes con necesidades educativas especiales y sensibles. Su labor compromete áreas como la psicología educativa, psicología clínica, terapia de lenguaje, odontología y trabajo social, teniendo como objetivo principal promover la construcción integral de cada uno de sus pacientes.

A pesar del impacto positivo de la UDIPSAI la dependencia de archivos físicos, así como fichas individuales generan inconvenientes como la duplicidad de información, retrasos a nivel operacional y en algunos casos la revictimización de pacientes al solicitar de manera repetida información sensible.

La implementación de un sistema de gestión de pacientes en la UDIPSAI responde principalmente a sus problemas a nivel operacional, ya que, al momento no cuentan con un sistema sólido para el correcto manejo de sus datos lo cual desencadena en duplicidad de datos, falta de integridad de la información, fallas a nivel operacional y mala experiencia para pacientes. La justificación de este desarrollo se fundamenta en 3 principales aspectos que se desglosan a continuación.

Primero, a nivel operacional el cambio de un proceso que es completamente manual por uno digital representa una mejora significativa en cuanto al tiempo empleado para realizar tareas cotidianas, como la gestión de pacientes y de especialistas permitiendo así la eliminación de documentación física con difícil acceso.

Segundo, la Seguridad de la información, así como su privacidad es un tema muy delicado, ya que en muchos casos se trata de menores de edad cuya información resulta ser bastante

sensible, y al contar con un sistema físico de estos registros toda esta información es comprometida.

Tercero, la experiencia de los usuarios de UDIPSAI puede ser negativo por el conflicto principal es la revictimización de los pacientes que manejan información sensible al estar constantemente experimentando la repetición constante de dicha información, para ello el uso del sistema propuesto al tener la información de cada uno de los pacientes opta por mejorar la calidad de la atención en UDIPSAI, además de aumentar considerablemente la comodidad y la confianza de sus pacientes.

El proyecto se centra en el desarrollo del backend con una arquitectura sólida para el producto de software de la UDIPSAI, que asegure un manejo centralizado y seguro de la información, optimice la calidad del servicio y reduzca la revictimización de los pacientes. Entre los objetivos específicos ha sido identificar los requisitos funcionales y no funcionales del backend que permitan la integración y eficiencia en el Sistema de Gestión de pacientes de UDIPSAI. Implementar una arquitectura escalable utilizando Spring Boot y el patrón MVC+S para procesar datos con alta seguridad y rendimiento. Desarrollar servicios RESTful que faciliten el registro, consulta y actualización de información por parte de los especialistas, garantizando la integridad y privacidad de los datos del paciente.

Referencias bibliográficas

Agudelo, O., Sanabria, F. R., & Rodríguez, S. V. (2021). Evaluación de una Arquitectura de Software. *Prospectiva*, *19*(2), Article 2. https://doi.org/10.15665/rp.v19i2.2636

Bautista-Villegas, E. (2022). Metodologías agiles XP y Scrum, empleadas para el desarrollo de páginas web, bajo MVC, con lenguaje PHP y framework Laravel. *Revista Amazonía Digital*, *1*(1), Article 1. https://doi.org/10.55873/rad.v1i1.168

Dhalla, H. K. (2021). A Performance Comparison of RESTful Applications Implemented in Spring Boot Java and MS.NET Core. *Journal of Physics: Conference Series*, *1933*(1), 012041. https://doi.org/10.1088/1742-6596/1933/1/012041

FRONTEND AND BACKEND DEVELOPER DIFFERENCE AND ADVANTAGES | Multidisciplinary Journal of Science and Technology. (s. f.). Recuperado 13 de enero de 2025, de https://mjstjournal.com/index.php/mjst/article/view/1875

Gómez, O. S., Rosero, R. H., & Cortés-Verdín, K. (2020). CRUDyLeaf: A DSL for Generating Spring Boot REST APIs from Entity CRUD Operations. *Cybernetics and Information Technologies*, *20*(3), 3-14. https://doi.org/10.2478/cait-2020-0024

León Soberón, J. J. (2020). *Análisis comparativo de sistemas gestores de bases de datos postgresql y mysql en procesos crud*.

Lituma-Sarmiento, A. F., & Vizñay-Durán, J. K. (2023). Análisis y Diseño de una propuesta de sistema integral de Gestión Empresarial basado en una arquitectura Cliente-Servidor. *MQRInvestigar*, *7*(1), Article 1. https://doi.org/10.56048/MQR20225.7.1.2023.2262-2290

Liu, Y., Li, Y., Deng, G., Liu, Y., Wan, R., Wu, R., Ji, D., Xu, S., & Bao, M. (2022). Morest: Model-based RESTful API testing with execution feedback. *Proceedings of the 44th International Conference on Software Engineering*, 1406-1417. https://doi.org/10.1145/3510003.3510133

Matute-Álvarez, J. I., Tenén-Banegas, A. D., Sañay-Sañay, S. I., & Gaona-Pineda, J. S. (2023). Sistema de realidad virtual para la gestión de la visualización e interacción con los ecosistemas del Bioparque AMARU Cuenca, Ecuador. *MQRInvestigar*, *7*(3), Article 3. https://doi.org/10.56048/MQR20225.7.3.2023.3313-3338

Moyano, J. H., Cenci, K. M., & Ardenghi, J. R. (2020). *Arquitectura Cliente-Servidor de Alto Rendimiento para servicio RTK*. XXVI Congreso Argentino de Ciencias de la Computación (CACIC) (Modalidad virtual, 5 al 9 de octubre de 2020). http://sedici.unlp.edu.ar/handle/10915/114094

Puspita, S. M., Ardhani, A. W., Retnaningrum, D. A., Firmansyah, A. R., & Rolliawati, D. (2024). ANALYSIS OF SOFTWARE QUALITY USING THE FURPS+ MODEL. *JURTEKSI (Jurnal Teknologi dan Sistem Informasi)*, *11*(1), Article 1. https://doi.org/10.33330/jurteksi.v11i1.3233

Sandoval, A., & Ismael, O. (2022). Modelo furps aplicado al análisis de calidad de un software desarrollado con Sencha Ext Js. *ISSN 2525-1333*. http://repositoriocyt.unlam.edu.ar/handle/123456789/1215

Sañay, I. S., Molina, E. B., & Masache, O. C. (2019). La ingeniería del software para el desarrollo industrial en la zona 6 del Austro. *RECIMUNDO*, *3*(1), Article 1. https://doi.org/10.26820/recimundo/3.(1).enero.2019.1625-1643

Seoane, J. A. (2010). *¿QUÉ ES UNA PERSONA CON DISCAPACIDAD?*

Spring Boot: Spring Boot. (s. f.). Recuperado 12 de enero de 2025, de https://docs.spring.io/spring-boot/

Sutherland, K. S. & J. (2020). *La Guía Definitiva de Scrum: Las Reglas del Juego*. https://repositorio.uvm.edu.ve/handle/123456789/59

Joshi, P. K. (2024). Spring Boot Security in Payment Gateway Applications. *International Journal of Computing and Engineering*, *6*(4), 13-21.

Kithulwatta, W. M. C. J. T., Jayasena, K. P. N., Kumara, B. T. G. S., & Rathnayaka, R. M. K. T. (2022). Performance Evaluation of Docker-based Apache and Nginx Web Server. *2022 3rd International Conference for Emerging Technology (INCET)*, 1-6. https://doi.org/10.1109/INCET54531.2022.9824303

Mohammady, H. (2024). *Enhancing the security and privacy of home storage servers in private clouds using zero trust principles* [fi=Ylempi AMK-opinnäytetyö|sv=Högre YH-examensarbete|en=Master's thesis|]. http://www.theseus.fi/handle/10024/876384

Campoverde-Calle, M. S., Sañay-Sañay, S. I., & Cabrera-Duffaut, A. E. (2024). Desarrollo de Metodología de Validación de APPS basada en el Modelo de Aceptación Tecnológica: Caso de estudio "BodyUC". *MQRInvestigar*, *8*(1), Article 1. https://doi.org/10.56048/MQR20225.8.1.2024.3742-3770

Conflicto de intereses:

Los autores declaran que no existe conflicto de interés posible.

Financiamiento:

No existió asistencia financiera de partes externas al presente artículo.

Agradecimiento:

Se extiende un agradecimiento cordial a los profesionales de UDIPSAI, quienes formaron parte crucial para la toma de requisitos dentro del proyecto. Se reconoce a Carlos Valladarez por su compromiso al desarrollar el frontend del sistema. Finalmente un agradecimiento especial a la Lic. Joselyn Urgiles por su apoyo en la revisión de este artículo

Nota:

El artículo no es producto de una publicación anterior.

Anexos

Anexo 1

Requerimientos refinados mediante la metodología FURPS.- Los identificadores toman la inicial en inglés del acrónimo FURPS, siendo FS los requerimientos funcionales del sistema. La prioridad de los requisitos, junto a una estimación de dificultad de dichos requerimientos.

Funcionalidad (F):

F1 - Registro y gestión de pacientes mediante una ficha digitalizada.

F2 - Implementar búsquedas por cédula de identidad, número de ficha o nombre.

F3 - Permitir acceso a información específica de pacientes para especialistas.

F4 - Crear formularios especializados (historia clínica, terapia de lenguaje, fonoaudiología, psicología clínica y educativa).

F5 - Generar y registrar seguimientos de citas con detalles específicos.

F6 - Registrar el historial de cambios en las fichas, con valores anteriores y nuevos.

F7 - Generar archivos PDF a partir de fichas especializadas.

F8 - Digitalizar y almacenar resultados de pruebas en formato PDF.

F9 - Gestionar la información de especialistas (crear, listar, actualizar, eliminar).

F10 - Asignar o eliminar pasantes para especialistas.

F11 - Permitir a los usuarios cambiar sus contraseñas.

F12 - Garantizar la seguridad en el almacenamiento y uso de datos.

Usabilidad (U):

U1 - Diseñar una interfaz intuitiva y accesible.

U2 - Proporcionar documentación clara para facilitar el uso del sistema.

U3 - Garantizar el acceso remoto para especialistas.

Confiabilidad (R):

R1 - Mantener un historial de cambios en las fichas de pacientes.

R2 - Asegurar la disponibilidad del sistema para acceso remoto.

Desempeño (P):

P1 - Optimizar el tiempo de respuesta en consultas y operaciones CRUD.

P2 - Garantizar la eficiencia en almacenamiento y recuperación de datos.

Soporte (S):

S1 - Proporcionar soporte técnico para instalación y configuración.

S2 - Capacitar a los especialistas en el uso del sistema.

S3 - Ofrecer tutoriales en video para resolver dudas.

S4 - Brindar soporte postimplementación durante tres meses.