Revista de investigación de sistemas e informática
16(2): 85-101 (2023)
DOI: https://doi.org/10.15381/risi.v16i2.27130

ISSN: 1815-0268 (Impreso) / ISSN: 1816-3823 (Electrónico)
Facultad de Ingeniería de Sistemas e Informática - UNMSM

Aplicativo móvil para la gestión médica en pacientes de la clínica universitaria de una universidad de Lima Metropolitana

Mobile application for the medical management of patients at the university clinic of a university in Metropolitan Lima

Jose Bryan Calle Soraluz
https://orcid.org/0009-0006-4618-808X
jose.callesoraluz@unmsm.edu.pe

Luiggi Steep Pasache Lopera
https://orcid.org/ 0009-0000-9732-7488
luiggi.pasache@unmsm.edu.pe

Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Lima, Perú

Ivan Carlo Petrlik Azabache
https://orcid.org/0000-0002-1201-2143
ipetrlika@unmsm.edu.pe

Universidad Nacional Mayor de San Marcos, Lima, Perú

Recibido: 25/10/2023 - Aceptado: 25/11/2023 - Publicado: 30/12/2023

RESUMEN

El presente artículo tiene como finalidad realizar una mejora en el sistema actual de citas de la clínica universitaria de una universidad de Lima Metropolitana a través del desarrollo de una aplicación móvil. La problemática de esta investigación son las deficiencias en la gestión médica para los pacientes, lo que causa descoordinación, falta de comunicación efectiva, pérdida de tiempo y afectación de la calidad de la atención médica y la satisfacción del paciente. El diseño de la investigación fue cualitativo y el tipo fue descriptivo, el resultado que por medio de la comparación entre la satisfacción con el aplicativo y sin el aplicativo de la gestión médica de los pacientes de la clínica universitaria. Se logró una tasa de rendimiento efectiva de 384 ms. en las peticiones y una desviación estándar de 200 ms. Con el análisis de rendimiento de 100 usuarios y el tiempo de respuesta en paralelo fue de menos de 1 s. Se concluye una mejor percepción por parte de los pacientes en utilizar una plataforma rápida y eficiente.

Palabras clave: Tecnología móvil, citas clínicas, rendimiento, tiempo de respuesta, Kotlin, Jetpack Compose.

ABSTRACT

The purpose of this article is to make an improvement in the current appointment system of the clinic of the Universidad Nacional Mayor de San Marcos through the development of a mobile application. The problem of this research is the deficiencies in medical management for patients, which causes lack of coordination, lack of effective communication, loss of time and affects the quality of medical care and patient satisfaction. The research design was qualitative and the type was descriptive, the result that through the comparison between the satisfaction with the application and without the application of the medical management of the patients of the university clinic. Changed an effective throughput rate of 384 ms. in requests and a standard deviation of 200 ms. With the performance analysis of 100 users and the response time in parallel was less than 1 s. It is concluded a better perception on the part of the patients in using a fast and efficient platform.

Keywords: mobile technology, clinical appointments, performance, response time, Kotlin, Jetpack Compose.

I. INTRODUCCIÓN

La calidad de la atención médica es un factor crucial en la satisfacción y bienestar de los pacientes. Según Research2Guidance. (2017), los pacientes de una institución de salud son los que tienen mayor beneficio en utilizar aplicaciones móviles orientadas a la gestión de citas médicas. Esto se contrasta con el beneficio al personal médico que es de un 30%, en este sentido, la Clínica universitaria de una universidad de Lima Metropolitana, como institución comprometida con la excelencia académica y el cuidado de la salud, reconoce la importancia de realizar una mejora continua en la atención que brinda hacia sus pacientes. Conscientes de los desafíos y las demandas actuales en el campo de la medicina, es decir se plantean estrategias innovadoras que optimicen los procesos médicos y promuevan una atención más eficiente, personalizada y accesible. En este contexto, el presente estudio se centra en la implementación de un Aplicativo Móvil para la Gestión Médica, con el objetivo de potenciar la calidad asistencial en la Clínica Universitaria, asegurando una experiencia integral y satisfactoria para cada paciente que busca atención médica.

Lindsay & Rader (2022) Indican que cada consulta en la clínica toma un intervalo de tiempo de 94,29 minutos, lo que representa un incremento del tiempo en el que se debería tomar la consulta en 157,15%. Lo que nos muestra el aumento significativo de tiempo perdido al momento de realizar la consulta en las clínicas.

Seguidamente Becker, et. al. (2017) manifiestan que las políticas que inciden en el mejoramiento de la calidad en la consulta clínica no se cumplen debido a que no hay una planificación constante de este tipo de actividades generando una deficiente evaluación de los centros de salud.

Espinoza, et. al. (2021) afirma que el sistema de salud peruano es deficiente y ante esa problemática podemos desmembrar ciertos puntos en el que recae esta ineficiencia. Indica que en los establecimientos de salud historias clínicas organizadas en estantes con la probabilidad que se pierdan en el accionar diario por el personal administrativo. Así mismo, en los hospitales no hay una correcta gestión de citas médicas. Una persona se tiene que acercar al establecimiento de salud para gestionar una cita y volver otro día para ser atendida. El mismo caso para gestionar consultas médicas acerca del estado de salud del paciente.

Todos los aspectos mencionados anteriormente, generan un malestar en los pacientes del sistema de salud público por lo que buscan una alternativa privada mucho más eficiente en el que algunas cuentan como un sistema integral como un Sistema de Planificación de Recursos Empresariales (ERP) que permite a todo el personal involucrado en una clínica gestionar sus procesos en una aplicación escritorio, web y móvil. Para identificar con exactitud los inconvenientes se vio conveniente realizar un árbol de problemas (Figura 1).

Figura 1.
Árbol de Problemas del trabajo en investigación.

La figura 1, indica las deficiencias en el servicio que proporciona la clínica universitaria de una universidad de Lima Metropolitana en la gestión médica para los pacientes, lo que resulta en descoordinación entre el paciente y área administrativa, demora en la búsqueda de información del paciente y el descontento de los pacientes.

Es así como una institución pública como la Clínica universitaria de una universidad de Lima Metropolitana no puede ser ajena de la digitalización y es por ello por lo que el presente trabajo de investigación se complementará con un aplicativo móvil que brinde a la comunidad universitaria de San Marcos y el público en general una opción eficiente e innovadora el uso de una herramienta digital para la gestionar las atenciones médicas con esta institución.

Esta problemática puede generar consecuencias negativas tanto para los pacientes como para la clínica universitaria. Para los pacientes, puede resultar en frustraciones debido a horarios imprecisos de las citas médicas, falta de comunicación efectiva con el área administrativa y falta de acceso oportuno a su información médica. Para la clínica, puede implicar una gestión ineficiente de las citas médicas y las historias clínicas, pérdida de tiempo en la búsqueda de información del paciente y descontento de los pacientes, lo que puede afectar la calidad de la atención médica y la satisfacción del paciente.

Siguiendo esta línea, la Organización Mundial de la Salud (OMS) indica que estas soluciones tecnológicas ayudan a mejorar el nivel de vida del paciente logrando así un bienestar y menos engorroso. Seguidamente Souza et. al. (2022) indica en referencia al aplicativo utilizado en su investigación que “Es una herramienta que presenta información confiable para ser utilizada en la atención”, además en consecuencia Zhu & Cyr (2018) concluyeron que “la implementación de aplicaciones móviles para la gestión de citas médicas no solo beneficia a los pacientes, sino que también puede tener un impacto positivo en el personal médico y en el sistema de salud en general”. Es decir, que la implementación de aplicaciones móviles para la gestión de citas médicas no solo beneficia a los pacientes, sino que también puede tener un impacto positivo en el personal médico y en el sistema de salud en general. Aunque el porcentaje de beneficio puede ser menor para el personal médico en comparación con los pacientes, aún existen varias ventajas para ellos.

Velásquez (2018), pone en énfasis que es importante que los ejecutores de los centros de salud tomen decisiones adecuadas con la finalidad de que los pacientes tengan una buena experiencia en el nosocomio. Para ello se debe recabar información en formato encuesta sobre la satisfacción de los clientes con el servicio.

Para abordar el desafío de mejorar la gestión médica en la Clínica Universitaria, se ha desarrollado un Aplicativo Móvil integral y personalizado que se ha diseñado específicamente para cubrir las necesidades de los pacientes y los profesionales de la salud. Esta solución tecnológica ha sido creada con un enfoque centrado en el usuario, incorporando características intuitivas y de fácil navegación para garantizar una experiencia óptima. El aplicativo móvil permite a los pacientes acceder a su historial médico de forma segura y conveniente, programar citas, y revisar el historial de citas visitadas. Además, para los profesionales de la salud, el aplicativo facilita la gestión de citas, la asignación de tratamientos, el seguimiento de pacientes y la visualización de datos clínicos en tiempo real, lo que agiliza los procesos médicos y mejora la toma de decisiones

II. ANTECEDENTES

A continuación, en la presente investigación se va a presentar los antecedentes:

Según Zhu & Cyr (2018), se investigó el uso de aplicaciones móviles de salud en la mejora de experiencia del paciente, sobre cómo el paciente de China interactúa y ve beneficioso el uso de un aplicativo móvil para gestionar sus citas médicas. Para ello se recabó información para el estudio usando un cuestionario a 300 pacientes logrando una satisfacción de los mismos ya que esta solución disminuye significativamente trámites burocráticos que suelen ser engorrosos en otros establecimientos más pequeños, además se demostró que el uso de aplicaciones móviles de salud aumentó las puntuaciones positivas de experiencia ambulatoria en un 17,7%.

Siguiendo esa línea, la investigación nigeriana de Ajayi, et. al. (2019) desarrollaron una app móvil que permite la reprogramación de citas médicas y que ha permitido reducir trámites, estrés haciendo que no solo mitigue los problemas que existen en el país africano sino también en otras partes del mundo. Los autores enfatizaron la metodología para realizar el proyecto que consistió en la recopilación de datos de información, el modelado de los diagramas UML, el diseño e implementación que se desarrolla en una aplicación Android. Así mismo, adopta una arquitectura básica en el que el usuario interactúa con el modo de autenticación, libro de citas y consultas médicas, y el médico además de

estas casuísticas también decide si aceptar o no la cita médica del paciente.

Según Martínez, et.al. (2021), se estableció una metodología de desarrollo de la aplicación para un sistema de atención médica (MAS) en México, estableciendo como primer paso una estrategia de benchmarking como estudio. En esa misma línea, los autores establecieron investigaciones sobre la percepción médica y la disponibilidad de los potenciales usuarios. Una vez que se aprobaron estos pre-requisitos del proyecto, se aprobaron los requerimientos y se eligió un diseño e implementación del sistema, esto en relación con el proceso correcto de desarrollo de software. Los autores realizaron una conexión directa a la base de datos, haciendo que la aplicación sea monolítica. A través del desarrollo de este aplicativo, lograron mayor usabilidad y ahorro de tiempo y dinero de parte de los pacientes, ya que la consulta es de 50 pesos, lo que reduce más de 75% del gasto invertido con los métodos tradicionales y sin perderla jornada laboral, con la creación de una aplicación usable a nivel medio, según el 85% de los entrevistados.

Según Higita (2020) desarrolló un prototipo de aplicación móvil que permite a los estudiantes de Uniminuto Bello realizar simulacros preparatorios para las pruebas de estado Saber Pro y TyT, utilizando Android Studio y el lenguaje Kotlin, este proyecto se basó en el modelo Mobile-D que se divide en 5 fases exploración, iniciación, producción, estabilización y pruebas del sistema. Para la aquitectura en el backend utilizaron una arquitectura en microservicios que permite el acceso a la información almacenada en diferentes servidores mediante una API, mediante archivos PHP. Concluyeron en la importancia del levantamiento de los requerimientos para identificar cada una de las actividades que se realizaron, además comentaron las tecnologías que necesitaban aprender para lograr los requerimientos propuestos.

Según Reinoso et. al. (2022) desarrolla 2 aplicaciones multiplataforma utilizando Ionic, y SCRUM como metodología ágil de su proyecto. Estas aplicaciones, una para pacientes y otra para profesionales de la salud, utilizan un servidor RASA para procesar las entradas de usuario y responder según estas, además utiliza un servidor desarrollado en Spring con una arquitectura en microservicios que permiten la flexibilidad, escalabilidad e independencia de cada componente del sistema.

Según Alvarez et. al. (2020) investigó la creación de una aplicación móvil para incrementar la recurrencia visitas de personas al médico que actuar como un puente entre el paciente y médicos, para evaluar sus resultados realizaron entrevistas a diversos médicos y pacientes, para conocer el impacto del aplicativo en la mejora de la experiencia de creación de citas médicas. Lograron describir el público objetivo del aplicativo, concluyendo que les interesa mucho cuidar su salud y la de sus familias, y prefieren usar una aplicación móvil para facilitar la gestión de su atención médica. Buscan soluciones rápidas, confiables y seguras. Los médicos, por su parte, buscan atraer nuevos pacientes y sienten que falta un sistema integrado. Sin embargo, pueden tener dificultades para adaptarse a nuevos sistemas informáticos.

III. MATERIALES Y MÉTODOS

Nuestra muestra poblacional consiste en todos aquellos pacientes de la Clínica universitaria de una universidad de Lima Metropolitana que requieren servicios médicos y podrían beneficiarse del uso del aplicativo móvil para la gestión médica. Se tomaron en cuenta 100 estudiantes de la UNMSM y un análisis de rendimiento con la misma cantidad de usuarios.

Para la generación de resultados, en primer lugar, se utilizará la herramienta JMeter que según Gomes et. al. (2019) es una herramienta útil para crear tests de rendimiento para aplicaciones gracias a su herramienta “HTTPS Test Script Recorder” que nos permite analizar tests de rendimiento y de esa forma poder evaluar el rendimiento del aplicativo creado.

Además, se describirán las tecnologías utilizadas para la creación del aplicativo móvil y de los servicios que esta va a utilizar.

Para la creación del sistema de gestión médica, se han desarrollado dos aplicaciones, una móvil y un web service, que permiten acceder a los servicios que ofrece la clínica universitaria en la Universidad Nacional Mayor de San Marcos.

Sobre el lenguaje de programación Kotlin Luca et. al. (2020) indica que “Una de las propiedades más citadas del lenguaje Kotlin es que hace que el código sea más fácil de entender y, por lo tanto, más fácil de mantener”, además según Esakia et. al. (2020) este lenguaje de programación presenta ventajas sobre Java, otro lenguaje utilizado para desarrollo android, como la inferencia de tipos, funciones de orden superior, extensiones, clases de datos y soporte para corutinas, razones por la cual hemos optado por utilizar este lenguaje para el desarrollo de la aplicación móvil objeto de la presente investigación, además se utilizará como framework

de desarrollo Jetpack Compose, debido que es recomendado por Kunneth (2022) en su libro, gracias a que simplifica el mantenimiento del aplicativo móvil, proporcionando una forma declarativa de crear interfaces de usuario, utilizando únicamente el lenguaje de programación Kotlin.

Sobre .NET 7, Llerena et. al. (2022) indica que esta nueva versión de .NET ya no limita ejecutarse en el sistema operativo Windows, sino que también puede ejecutarse y ser desarrollado en IOS y Linux, lo que nos permitirá trabajar con servidores de tipo Linux, por lo que, para los servicios REST, comunicación entre la aplicación y la base de datos SQL, se utiliza el lenguaje de programación C# usando el framework .NET 7 como lo aplica Duque (2016).

La comunicación entre estos sistemas se realizará usando Retrofit, ya que es un estándar utilizado mayormente en kotlin para el desarrollo de aplicaciones móviles, que permitirá la gestión de los endpoints.

A. Arquitectura de la aplicación

Como se observa en la figura 2, la aplicación se desarrolla bajó el paradigma de servicios es por ello que la solución se comunica a un App Service proporcionado por Microsoft (2023), el cual se llama (ClínicaUniversitaria), cuando el usuario interactúa y envía o recibe datos por parte de la aplicación. Este App Service, se comunica con la base de datos (BDClinica) en Azure. Asimismo, también se comunica con un servicio de alojamientos de datos multimedia (ClinicaStorage) BlobStorage en Azure administrado por Microsoft (2023) y para las notificaciones se comunicará con Firebase para que el usuario pueda ser notificado ante cualquier eventualidad.

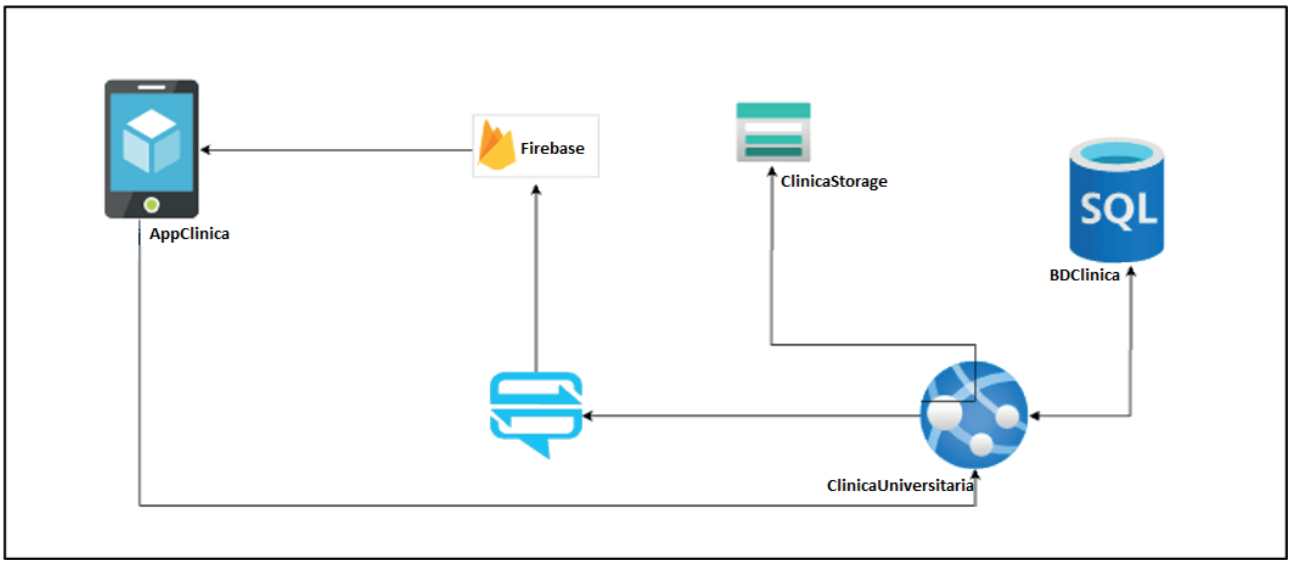
B. Patrón de diseño

La solución comprenderá varios componentes comunicados entre sí usando el principio del patrón Model View ViewModel (MVVM), observada en la figura 3, que según Zhang et. al. (2023) es una mejor alternativa para la manipulación constante de los datos que se muestran en interfaces de usuario, debido a que otras arquitecturas como Modelo Vista controlador (MVC), reducen el desempeño del aplicativo, por lo que usando esta arquitectura podemos tener un mejor desempeño que utilizando otras arquitecturas más clásicas, tales como MVC, que según Pantoja (2023), incrementa el tiempo de desarrollo que en una aplicación e incrementa la cantidad de clases necesarias.

En primer lugar, el usuario interactúa con un Activity o Fragment, este elemento se comunica internamente con un ViewModel que permitirá la comunicación de datos y la actividad interna utilizando la clase “MutableState” propia de JetPack Compose. El Repository permitirá la gestión de los datos que pueden ser de dos maneras: utilizando una base de datos local, almacenada en caché en el celular, usando Room - SQLite y también conectando la aplicación al App Service mediante el servicio REST usando Retrofit como intermediario.

Figura 2

Arquitectura del Software

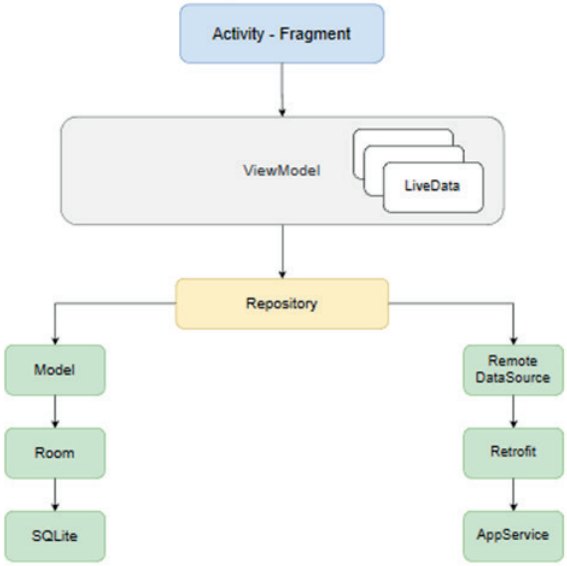


A continuación, en la figura 3, se presenta el patrón de diseño utilizado para el aplicativo móvil.

Como podemos observar en la figura 3, tenemos un gráfico de lo que representa la arquitectura escogida MVVM, esta arquitectura según Kouraklis (2016) se encuentra dentro de la categoría de arquitecturas de 3 capas, donde tenemos una capa de acceso a datos, otra de lógica de negocio y presentación, que se representan en la imagen por los repositorios, viewModel y los activity/fragments respectivamente.

Figura 3.

Patrón de diseño de la aplicación



C. Prototipado

A continuación, presentamos los módulos del sistema o aplicación móvil de la siguiente manera:

1. Módulo de acceso

En la figura 4 muestra la pantalla de inicio de sesión, donde el usuario debe ingresar su dni y contraseña para acceder al sistema de gestión médica después de la validación.

La figura 5 muestra los campos requeridos para que un paciente se registre en el sistema y pueda acceder a los servicios de la clínica universitaria de una universidad de Lima Metropolitana.

En esta vista se puede observar las últimas noticias de la clínica universitaria, además de poder visualizar los servicios que esta ofrece.

La figura 6 presenta la vista de inicio del aplicativo móvil para la plataforma médica de la clínica universitaria, donde se podrán apreciar las Noticias más recientes y los Servicios que ofrece la clínica.

Según la figura 7, se describen las citas pendientes del usuario, además de añadir información extra sobre la cita que puede ser útil al usuario momentos antes de ingresar al consultorio médico.

Figura 4.

Login de la plataforma móvil.

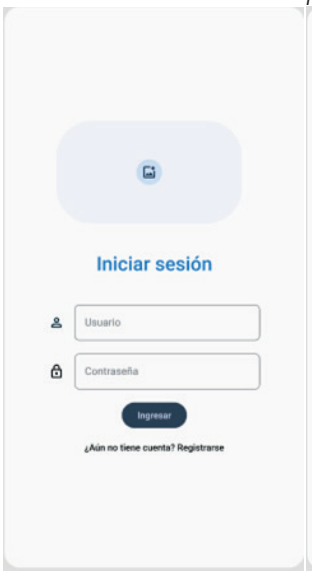


Figura 5.

Registro de usuario en la plataforma móvil.

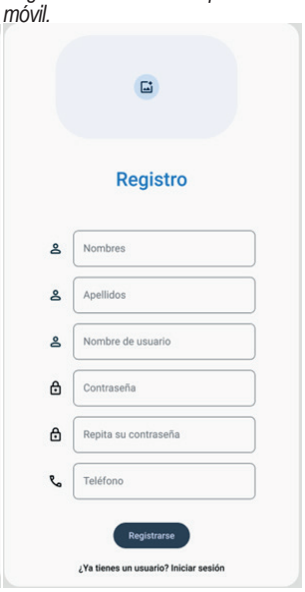


Figura 6.

Vista de inicio del aplicativo móvil.

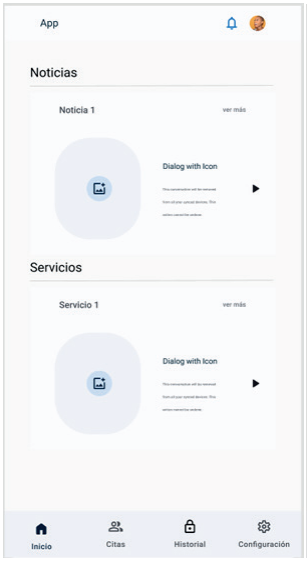
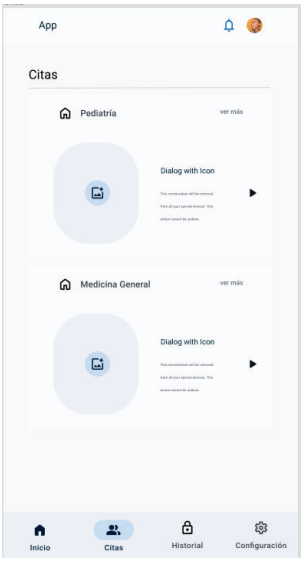


Figura 7.

Vista de citas del aplicativo móvil.



D. Estructura del Proyecto

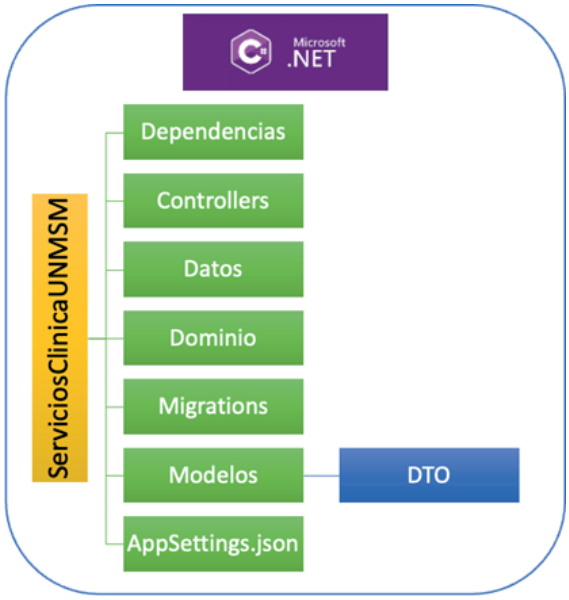
a. Backend

Como se explicó en el inciso IVa. la arquitectura del proyecto está orquestada en un ambiente de servicios en que la aplicación se conectará a un

backend desarrollado en C# .Net 7 y permitirá el envío y recepción de peticiones del usuario. Como primer paso se tiene la estructura de los paquetes del servicio (fig. 8) en la cual se pasará a detallar cada una de estas carpetas.

Figura 8.

Estructura de paquetes del servicio BackEnd.



A continuación, según la figura 8 vamos a detallar los elementos de esta figura:

- **Dependencias**

  En este package se almacenan los archivos que permitirán la conexión a plugins de nuestro servicio. Particularmente .Net 7 tiene un repositorio de servicios de Microsoft y de terceros llamado Paquetes Nuget. Estos objetos permiten diversas acciones como la conexión a un servidor SQL Azure, el uso de Entity Framework para ser el intermediario entre la capa de datos y la capa lógica de nuestro servicio, entre otros.

- **Controllers**

  .Net se familiariza con el patrón de diseño Modelo Vista Controlador (MVC), el cual permite una separación por capas de la estructura de nuestro proyecto. Una de estas capas es Controller. Este package recopila las clases controladoras, asigna rutas para direccionar los endpoints que realizan diversas operaciones CRUD (fig. 9) y otras más avanzadas. Para cada módulo del proyecto se asigna un controlador y este controlador abarca una o más APIs por la que la aplicación se va a conectar.

- **Datos**

  Este paquete contiene una clase permite la unión de todos los modelos que hemos considerado en el proyecto.

- **Dominio**

  La carpeta de Dominio almacena archivos de los atributos que va a considerar las diversas APIs de la aplicación. Estos atributos se extraen de la base de datos y albergan modelos de petición y respuesta de la aplicación. En la Figura 10. Se muestran una serie de atributos en formato JSON para la consulta de noticias. Esto permitirá que la aplicación pueda obtener y mostrar la lista de noticias.

- **Migrations**

  Los migrations corresponden a una serie de archivos que muestran las actualizaciones que ha tenido nuestra base de datos. Por temas de auditoría siempre se almacena bajo parámetros de fecha, hora y una descripción según el cambio subido. (Figura 11)

- **Modelos**

  Es en esta carpeta donde detallamos los atributos que van a tener nuestros modelos como llave primaria, campos requeridos, tipos de datos, etc. Es una técnica moderna implementada en Entity Framework llamada Code First en la que ya no nos preocupamos por realizar el modelo en un Script SQL sino en el mismo proyecto del servicio, permite actualizaciones continuas sin necesidad de realizar código extra.

- **AppSettings.json**

  Es el archivo de configuración general del proyecto. Es aquí donde se coloca la cadena de conexión a nuestro servicio de Azure SQL Server entre otros servicios.

b. Aplicación Móvil

La arquitectura del proyecto está desarrollada sobre el patrón Model View ViewModel (MVVM), que permitirá comunicarse con las vistas, logrando así reducir el acoplamiento y permite además aumentar la calidad de código.

Así como explica el diagrama de la figura 12, el proyecto tiene una estructura de paquetes organizados en carpetas, que se detallarán a continuación.

Figura 9.

Estructura de Endpoint que permite el registro de un usuario.

```
//REGISTRO DE UN USUARIO
[HttpPost("CrearUsuario")]
[ProducesResponseType(StatusCodes.Status201Created)]
[ProducesResponseType(StatusCodes.Status400BadRequest)]
[ProducesResponseType(StatusCodes.Status500InternalServerError)]
public ActionResult<UsuarioDto> CrearUsuario([FromBody] UsuarioDto usuarioDto)
{
    if(!ModelState.IsValid)
    {
        return BadRequest(ModelState);
    }
    if(_db.Usuarios.FirstOrDefault(v=>v.correo.ToLower() == usuarioDto.correo.ToLower())!=null)
    {
        ModelState.AddModelError("CorreoExiste", "El usuario con ese correo ya existe");
        return BadRequest(ModelState);
    }
    if(usuarioDto == null)
    {
        return BadRequest(ModelState);
    }
    if (usuarioDto.id_usuario > 0)
    {
        return StatusCode(StatusCodes.Status500InternalServerError);
    }
    Usuario modelo = new()
    {
        nombre = usuarioDto.nombre,
        apellidoPaterno = usuarioDto.apellidoPaterno,
        apellidoMaterno = usuarioDto.apellidoMaterno,
        correo = usuarioDto.correo,
        contrasenia = usuarioDto.contrasenia,
        fechaCreacion = usuarioDto.fechaCreacion,
        fechaActualizacion = usuarioDto.fechaActualizacion,
        imagen = usuarioDto.imagen,
        estado = usuarioDto.estado
    };
    _db.Usuarios.Add(modelo);
    _db.SaveChanges();
    return NoContent();
}
```

Figura 10.

Estructura de Endpoint que permite el registro de un usuario.

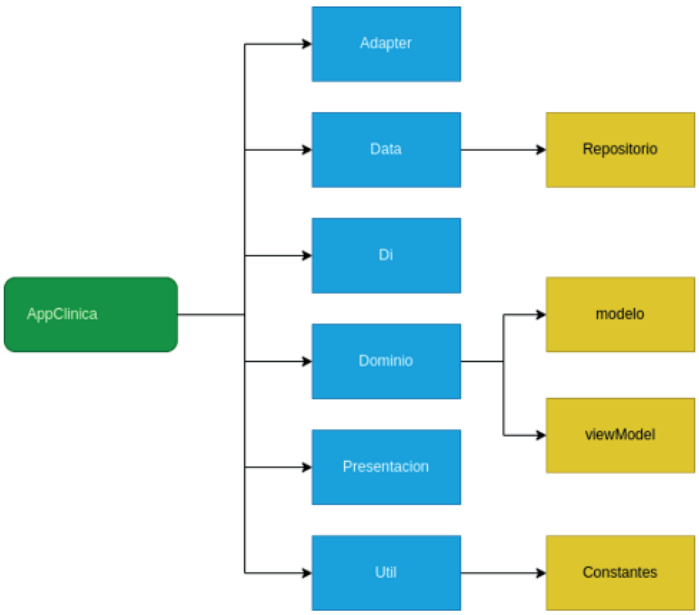
```
public class Noticias
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int id_noticia { get; set; }
    [Required]
    [StringLength(500)]
    public string url_noticia { get; set; }
    [StringLength(500)]
    public string img_noticia { get; set; }
    [Required]
    public int estado { get; set; }
}
```

Figura 11.

Lista de archivos de cambios en la estructura de las tablas de la base de datos.

Figura 12.

Estructura de paquetes del Aplicativo Android.



- **Adapter**

  El paquete "Adapter" se encarga de la comunicación entre los elementos externos y la capa de dominio de la aplicación. Contiene adaptadores que traducen los datos y las solicitudes provenientes de fuentes externas (como una base de datos, una API o la interfaz de usuario) a un formato que pueda ser utilizado por la capa de dominio. También se encarga de convertir las respuestas del dominio en un formato adecuado para ser mostrado o almacenado externamente.

- **Data**

- Este paquete es responsable de la manipulación y acceso a los datos de la aplicación. Aquí se encuentran las implementaciones de las interfaces definidas en la capa de dominio para la obtención, almacenamiento y manipulación de datos. Puede incluir clases para interactuar con bases de datos, servicios web, sistemas de archivos u otras fuentes de datos.

  - **Repositorio**

    Este paquete se utiliza para gestionar la capa de acceso a datos de la aplicación. Esta capa es responsable de interactuar con las fuentes de datos externas, como bases de datos, servicios web o sistemas de archivos.

- **Di**

- El paquete "DI" o Dependency Injection" se refiere a la gestión de la inyección de dependencias en la aplicación. Aquí se encuentran las configuraciones y las clases relacionadas con la creación y la provisión de instancias de objetos a lo largo de la aplicación. La inyección de dependencias es un patrón de diseño que permite desacoplar las clases y facilita la prueba unitaria y la modularidad de la aplicación.

- **Dominio**

- El paquete "Dominio" contiene el núcleo de la lógica de negocio de la aplicación. Aquí se encuentran las entidades, los casos de uso (use cases) y las interfaces que definen la funcionalidad principal de la aplicación. Este paquete no debe depender de ninguna capa externa y debe ser independiente de la tecnología utilizada. Representa las reglas y los procesos esenciales que definen el propósito de la aplicación.

  - **Modelo**

    Este paquete es utilizado para definir y representar los conceptos centrales y las reglas de negocio de la aplicación. Esta capa se centra en el núcleo funcional de la aplicación y encapsula la lógica de negocio, independientemente de

cualquier tecnología o infraestructura específica.

- **ViewModel**

  Se encarga de gestionar los estados de la aplicación y presentarlos a la vista, que se encuentra en el paquete “Presentación” presentado a continuación.

- **Presentación**
- El paquete "Presentación" es responsable de la interfaz de usuario y la presentación de los datos al usuario final. Aquí se encuentran las clases relacionadas con la interacción con el usuario, como las vistas, los controladores, los presentadores y los modelos de vista. Se encarga de mostrar los datos del dominio al usuario y de capturar las acciones o eventos del usuario para comunicarnos a la capa de dominio.
- **Útil**
- El paquete "Útil" contiene clases o funciones de utilidad que pueden ser utilizadas en diferentes partes de la aplicación. Estas clases proporcionan funcionalidades genéricas o ayudan a realizar tareas comunes de manera más sencilla. Pueden incluir funciones matemáticas, funciones para el manejo de fechas y horas, clases para la manipulación de archivos, datos que se utilizarán regularmente en toda la aplicación, etc.

2. Flujos importantes del aplicativo

Inicio de sesión

El diagrama mostrado anteriormente (figura 13), indica el flujo simplificado del aplicativo desde que se inicia la aplicación, hasta que se termina de completar la validación de datos y el inicio de sesión correspondiente.

Obtener datos

El diagrama mostrado anteriormente (figura 14), indica el flujo del aplicativo desde que se ejecuta

Figura 13.

Diagrama de secuencia muestra de Inicio de sesión.

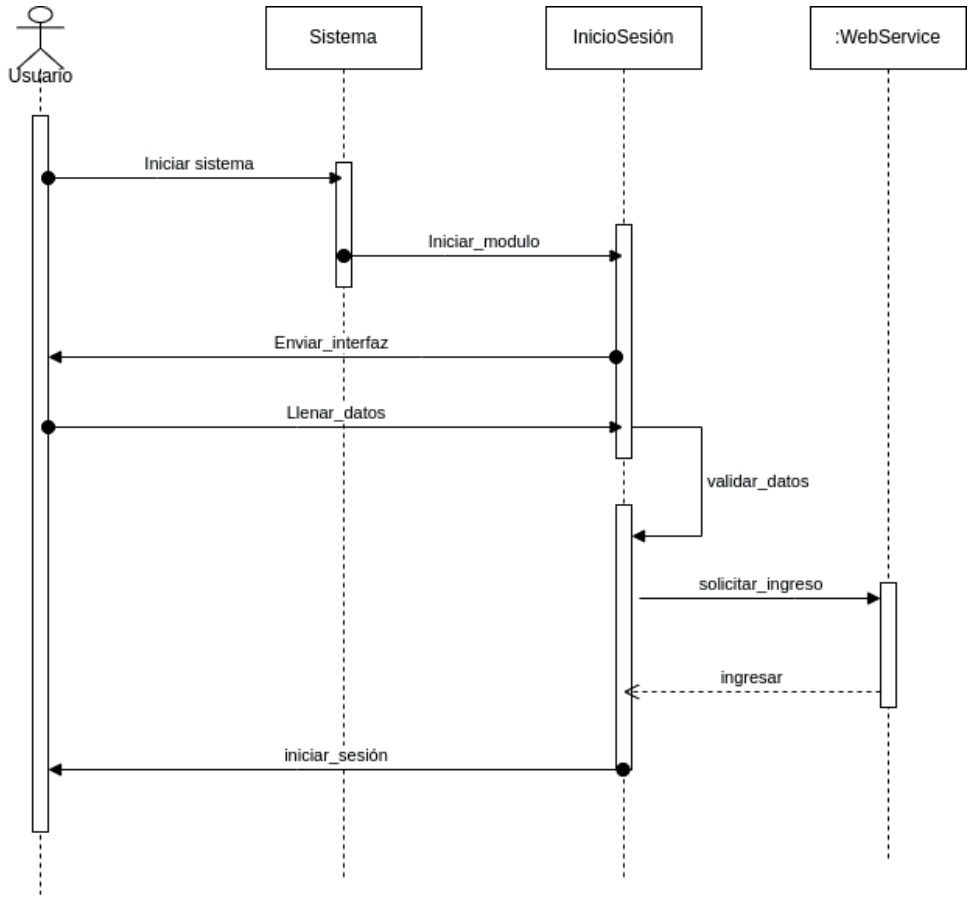
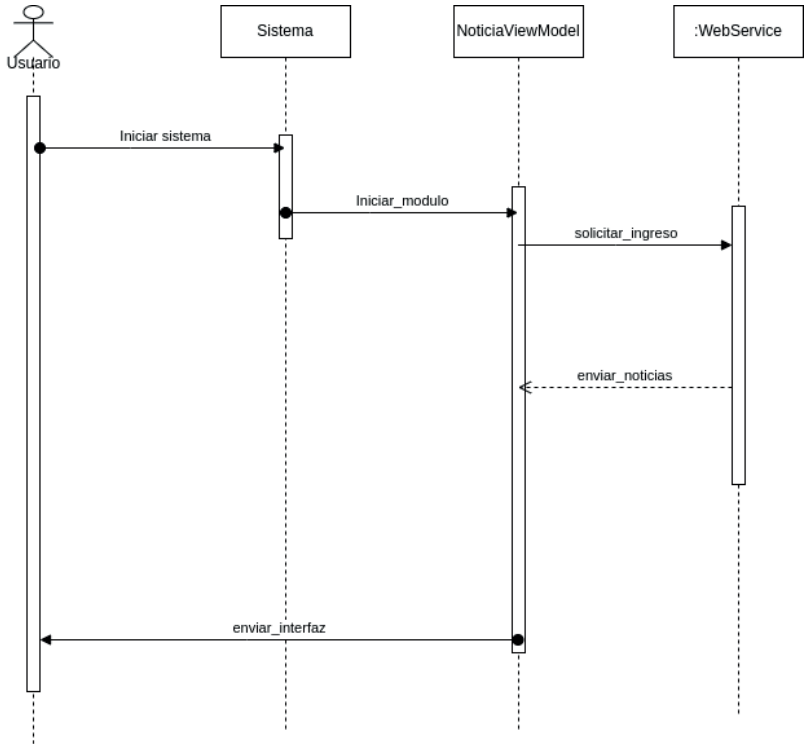


Figura 14.

Diagrama de secuencia obtención de datos



la aplicación, hasta que se termina de completar la obtención de datos, como se puede observar en el gráfico, se tiene una llamada asíncrona a la API, para obtener los datos requeridos de esta forma no se detiene el proceso de ejecución de la aplicación.

3. Técnicas utilizadas

Corrutinas

Una forma de solucionar el problema de la sincronización y condiciones de la carrera son las rutinas. Estas fueron creadas con el propósito de esperar a que el Backend termine de procesar todas las peticiones que realizan los usuarios dentro del aplicativo.

Como se puede evidenciar en la figura 15, que se realiza una corrutina en el alcance del ViewModel de las noticias. Esta corrutina se encarga de realizar la petición explicada con anterioridad al servidor para la obtención de noticias que serán mostradas al usuario.

Inyección de dependencias

La Inyección de Dependencias, según Villalobos & Díaz (2022), es un patrón de diseño de software que permite la reutilización de una clase sin necesidad de instanciarla sino enviar como parámetro el objeto de la clase lo que permite que nuestro código sea débilmente acoplado. El acoplamiento de un código es el grado de mantenibilidad y ello nos permitirá que el cambio de un módulo no impacte a otros.

Podemos clasificar la inyección de dependencias en 3 categorías (Gutierrez, Verduzco & Farías, 2015):

1. Inyección por interfaz: Consiste en definir una interfaz y realizar la inyección desde una clase a esta interfaz. En la figura 16 se puede observar la aplicación de la inyección por interfaz al hacer un llamado a la interfaz “ClinicaRepositorio” mediante el parámetro de la clase AgregarNuevoUsuario. Nótese que la inyección es aplicada gracias a la etiqueta “@Inject”.

2. Inyección por constructor: Se utiliza la palabra clave “constructor” para poder inyectar dependencia entre una clase a otra.

3. Inyección por setter: Se utiliza para poder inyectar dependencia entre un método de una clase hacia una clase. En el lenguaje programación Java se hace uso de la palabra reservada “setter”.

Figura 15.

Ejemplo de código de petición con rutinas.

```
class NoticiaViewModel : ViewModel() {

    Luiggi Pasache
    var state by mutableStateOf(emptyList<Noticia>())
        private set

    Luiggi Pasache
    init {
        viewModelScope.launch { this: CoroutineScope
            obtenerNoticias()
        }
    }

    Luiggi Pasache
    private suspend fun obtenerNoticias() {
        try {
            val noticias :List<Noticia> = ApiClinicaUNMSM.obtenerApi().obtenerNoticias()
            state = noticias
        } catch (e: Exception) {
            //TODO: Manejar excepcion
        }
    }
}
```

Figura 16.

Ejemplo de aplicación de la inyección por interfaz.

```
class AgregarNuevoUsuario @Inject constructor(
    private val clinicaUNMSMRepositorio: ClinicaUNMSMRepositorio
) {
    suspend operator fun invoke(registroUsuario: RegistroUsuario){
        try {
            val reg = clinicaUNMSMRepositorio.registrarUsuario(registroUsuario)
            Log.i( tag: "CU1NewUser",  msg: "Valido: $reg.toString()")
        }catch (e: Exception){
            Log.i( tag: "CU1NewUser",  msg: "Error: $e.toString()")
        }
    }
}
```

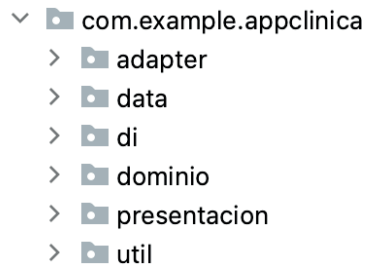
Arquitectura limpia

La Arquitectura Limpia fue publicada por Robert Martin en el ambiente de la Ingeniería de Software basada en componentes adoptando así los cimientos que según Martin (2017) esta arquitectura ayuda a estructurar de manera eficiente un proyecto software de tal forma que se separe la lógica de negocio con la esencia del desarrollo adoptando así una estructura en capas, además según Arias et. al. (2021) esta arquitectura ayudó a mejorar la analizabilidad, estabilidad, testeabilidad y cambiabilidad en la aplicación analizada en su artículo, por lo que mejora la calidad interna del software. Relacionado la aplicación esta arquitectura para el desarrollo Android, Tung (2017) indica que mejora la calidad de código de una aplicación móvil y reduce la complejidad.

Las capas de esta arquitectura se han estructurado según se muestra en la figura 17.

Figura 17.

Estructura de capas del proyecto Android.



Tal como se indicó anteriormente, en la figura 17 se puede observar la estructura de carpetas que sigue el código del aplicativo móvil para la clínica.

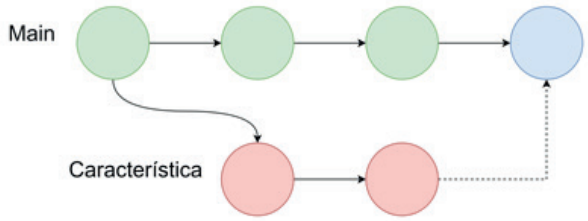
4. Versionado

En el transcurso de la presente investigación hemos decidido optar por un sistema de control de versiones altamente eficiente y confiable, como es el caso de Git, junto con la plataforma en línea Github. Con Git, podemos realizar un seguimiento preciso de todas las actualizaciones, revisiones y ajustes realizados en los archivos de nuestro proyecto. Además, GitHub nos ofrece la capacidad de almacenar y compartir nuestro repositorio de código en la nube, lo que facilita la colaboración entre los miembros del equipo de investigación. Con la funcionalidad de ramificación y fusión de Git, podemos trabajar en diferentes características para la aplicación en paralelo y luego combinarlos sin problemas. Además, GitHub proporciona una interfaz intuitiva para realizar un seguimiento de las discusiones, realizar sugerencias y realizar un seguimiento de problemas o errores encontrados en el proyecto. En resumen, la combinación de Git y GitHub ha demostrado ser una herramienta esencial en nuestra investigación, proporcionándoles un mayor control, colaboración y organización en el desarrollo de nuestro proyecto.

Además, tal como se observa en la figura 18, se utilizaron las siguientes convenciones de git para la realización de nuevas características:

Figura 18.

Estrategia de merge y ramas en el proyecto.



- **Nombres de ramas:** La rama principal tiene el nombre Main, mientras que las ramas secundarias se nombran según las características que se iban añadiendo
- **Estrategia de merge:** Se realiza merge cuando los commits que se iban a fusionar eran pocos, squash cuando estos commits necesitaban ser resumidos para comunicar de una mejor manera los cambios. Además, se utilizan pull request para asegurarnos que todos los integrantes del equipo revisen los cambios que se añadirán a la rama principal.

RESULTADOS

Para conocer el impacto real del aplicativo para mejorar la calidad del servicio ofrecido por la clínica.

Con base en el análisis de los datos recopilados durante el período de estudio, los resultados revelaron una clara mejora en la calidad del servicio de la clínica universitaria tras la implementación del aplicativo móvil. A continuación, se presentan los principales hallazgos y conclusiones obtenidos:

Mejora en la accesibilidad: El aplicativo móvil permitió a los pacientes acceder de manera más rápida y sencilla a los servicios de la clínica, eliminando la necesidad de hacer largas filas o esperar largos tiempos de espera. Los usuarios pudieron programar citas, consultar resultados de exámenes, acceder a su historial médico y recibir recordatorios de citas, entre otras funciones, a través de la aplicación.

Optimización de la gestión médica: El aplicativo móvil facilitó la gestión de la información médica de los pacientes, permitiendo a los profesionales de salud acceder a los datos relevantes de manera instantánea y actualizada. Esto agiliza los procesos de diagnóstico, tratamiento y seguimiento de los pacientes, mejorando la eficiencia y la precisión en la atención médica.

Mayor satisfacción de los pacientes: Según los datos recopilados a través de encuestas de satisfacción, se observó un incremento significativo en la satisfacción de los pacientes con el servicio brindado por la clínica después de la implementación del aplicativo móvil. Los pacientes destacaron la comodidad, la rapidez y la eficacia del sistema, así como la mejora en la comunicación con su médico y la posibilidad de tener un mayor control sobre su propia salud.

Reducción de errores y omisiones: El uso del aplicativo móvil contribuyó a minimizar errores y omisiones en el manejo de la información médica de los pacientes. La digitalización de los datos permitió una mejor organización, almacenamiento y acceso a la información, disminuyendo la posibilidad de confusiones o pérdida de datos importantes.

Optimización de consultas a los servicios: El uso del software Postman nos ayuda a testear los endpoints. En la figura 19 se observa que el tiempo de respuesta fue de 567 ms. con un peso del JSON de 508 Bytes y obteniendo como cuerpo de respuesta los atributos de la petición.

Además de la validación anterior, los endpoints se probaron con la herramienta JMeter, que nos permitió realizar las pruebas de rendimiento, a continuación, se presentarán los resultados entregados por la herramienta.

Como podemos observar en la figura 20, tenemos los registros de las peticiones realizadas con JMeter que muestran el tiempo de ejecución de cada petición, que resumiendo las métricas obtenemos.

Esta tabla 1, muestra un tiempo de respuesta medio de 384 ms para las 800 peticiones realizadas por los 100 usuarios, y una desviación estándar del tiempo de respuesta de 200 ms.

Como podemos observar en la figura 21, el tiempo de respuesta para los 100 usuarios empieza alto alrededor de los 700 ms para el caso de uso del listado de noticias, y que luego se disminuye notablemente para mantenerse junto a las demás

Figura 19.

Testing de API de validación de usuario por correo.

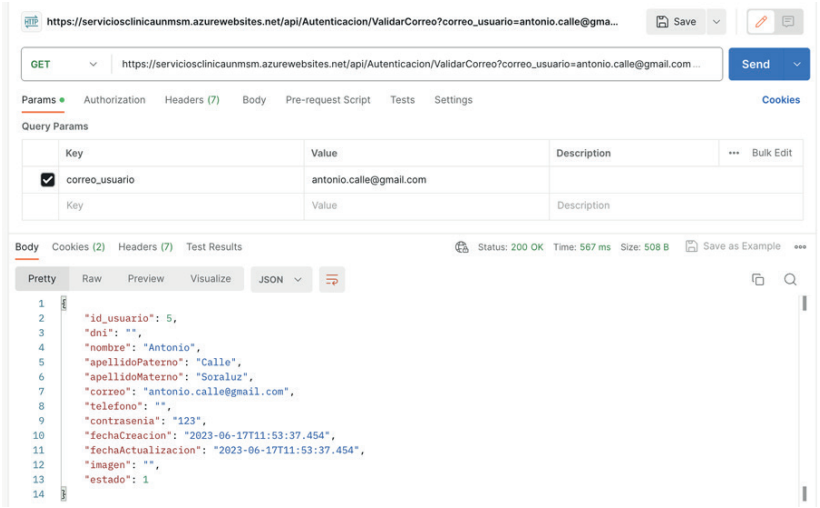


Figura 20.

Muestra de peticiones http a la API con JMeter.

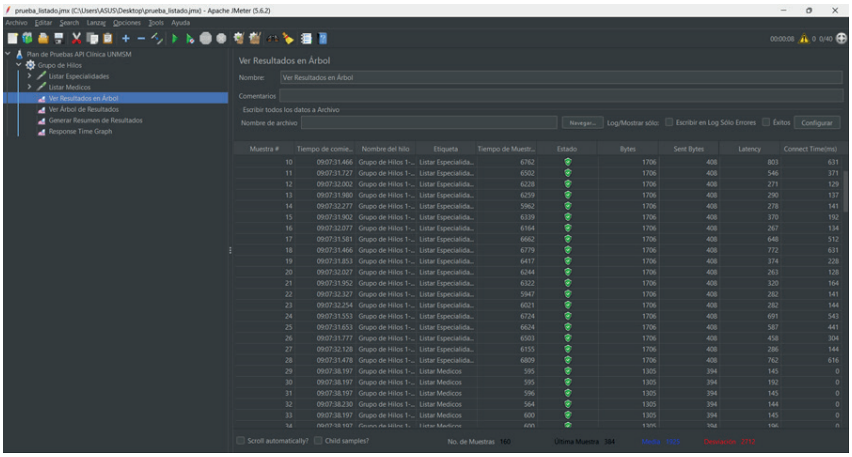


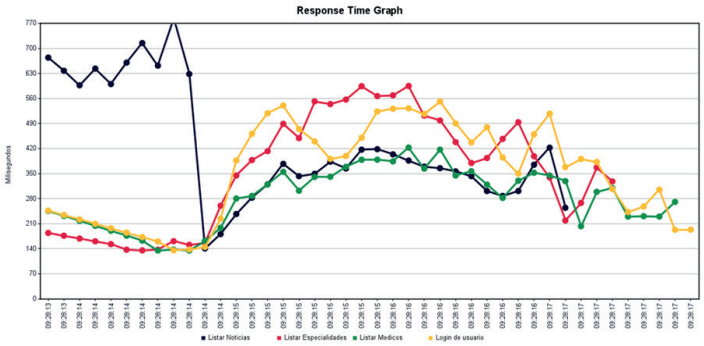
Tabla 01.

Métricas sobre peticiones http realizadas.

Métrica	Valor
Nro. de muestras	800
Media de tiempo de respuesta	384
Desviación estándar de tiempo de respuesta	200

Figura 21.

Gráfico de tiempo de respuesta de peticiones http con JMeter.



peticiones en valores normales entre 540 ms y 280 ms, lo que evidencia un tiempo de respuesta adecuado para la cantidad de peticiones recibidas.

CONCLUSIONES

En base a los resultados obtenidos, se puede concluir que la implementación del aplicativo móvil en la clínica universitaria ha tenido un impacto positivo en varios aspectos relacionados con la calidad del servicio y la experiencia de los pacientes.

La satisfacción de los pacientes ha experimentado un aumento significativo después de la implementación del aplicativo móvil. Los usuarios han destacado la comodidad, rapidez y eficacia del sistema, así como la mejora en la comunicación con su médico y la posibilidad de tener un mayor control sobre su propia salud. Estos aspectos han contribuido a una mayor confianza y satisfacción general con el servicio brindado por la clínica.

Por último, el uso del aplicativo móvil ha ayudado a reducir errores y omisiones en el manejo de la información médica de los pacientes. La digitalización de los datos ha permitido una mejor organización, almacenamiento y acceso a la información, disminuyendo la posibilidad de confusiones o pérdida de datos importantes. Esto ha contribuido a una mayor seguridad y precisión en los registros médicos.

En resumen, los resultados de esta investigación demuestran que la implementación del aplicativo móvil en la clínica universitaria ha tenido un impacto positivo en la accesibilidad, la gestión médica, la satisfacción de los pacientes y la reducción de errores. Estos hallazgos respaldan la efectividad de la tecnología móvil en el ámbito de la atención médica y sugieren que su adopción puede generar mejoras significativas en la calidad del servicio y la experiencia del paciente.

REFERENCIAS

[1] Alvarez, C., León, G., Lopez , M., Lucar , Y., & Suárez, E. (2020). Aplicativo móvil para reservaciones de citas médicas – MI DOC. Extraido de: https://repositorioacademico.upc.edu.pe/bitstream/handle/10757/654858/AlvarezV_C.pdf?sequence=3&isAllowed=y

[2] Ajayi, O., Akinrujomu, O, Daso, O., & Akinniyi, O. (2019). A mobile based medical appointment and consultation (MMAC) system. International Journal of Computer Science and Mobile Computing, 8(5), 208-218. Recuperado de: https://www.researchgate.net/publication/341669761_A_MOBILE_BASED_MEDICAL_APPOINTMENT_AND_CONSULTATION_MMAC_SYSTEM

[3] Arias, J., Reyna, B., & Mamani, G. (2021). “Repercusión de arquitectura limpia y la norma ISO/IEC 25010 en la mantenibilidad de aplicativos Android.” TecnoLógicas, 24(52), e2104. https://doi.org/10.22430/22565337.2104

[4] Becker, L., Loch, M., & Rodrigo, R. (2017). Barreiras percebidas por diretores de saúde para tomada de decisão baseada em evidências. Revista panamericana de salud publica [Pan American journal of public health], 41, 1. https://doi.org/10.26633/rpsp.2017.147

[5] Esakia, A. (2020). “Transitioning to teaching android with kotlin and jetpack components.” SIGCSE 2020 - Proceedings of the 51st ACM T echnical Symposium on Computer Science Education. S.l.: s.n., DOI 10.1145/3328778.3372603

[6] Espinoza, E., Gil, W., Agurto, E. (2021). Principales problemas en la gestión de establecimientos de salud en el Perú. Rev. Cubana de Salud Pública. DOI: https://scielosp.org/pdf/rcsp/2020.v46n4/e2146/es

[7] Gomes A., Demion B., Mouawad P. (2019), Master Apache JMeter - From Load Testing to DevOps. Packt Publishing Ltd. Extraído de: https://books.google.com.pe/books?id=D_amDwAAQBAJ&printsec=copyright&redir_esc=y#v=onepage&q&f=false

[8] Gutiérrez, C., Verduzco, J., Mendoza, N. (2015). Inyección de Dependencias en el Lenguaje de Programación Go. RIDE Revista Iberoamericana para la Investigación y el Desarrollo Educativo, vol. 5, núm. 10

[9] Higuita, J. (2020). "Prototipo de Aplicación móvil de simulacros preparatorios de las pruebas de estado Saber Pro y TyT -preSaber". Disponible en: https://repository.uniminuto.edu/bitstream/10656/15893/1/T.TI_HiguitaJaramilloJuanCarlos_2020.pdf

[10] Kouraklis, J. (2016). MVVM in Delphi: Architecting and building model view ViewModel applications (1a ed.). APress.

[11] Kunneth, T. (2022). Android UI Development with Jetpack Compose: Bring declarative and native UIs to life quickly and easily on Android using Jetpack Compose. Packt Publishing.

[12] Lindsay F., Rader M. (2022). Using Lean Healthcare Techniques to Reduce Appointment Times. DOI: https://doi.org/10.1016/j.nurpra.2022.11.018

[13] Llerena, A., Viscaino, F., Culque, W. (2022). "Desarrollo de software con Net Core". Revista Universidad Y Sociedad, 14(2), 85–89. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2218-36202022000200085

[14] Luca, A., Coppola, R., Malnati, G., Torchiano, M. (2020). Effectiveness of Kotlin vs. Java in android app development tasks, Information and Software Technology. Volume 127, 2020, 106374, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2020.106374

[15] Martin, R. (2017). Clean architecture: A craftsman's guide to software structure and design. Prentice Hall.

[16] Martínez, M., Muñóz, J., Pérez, R., Ramos, S. (2020). MAS: SISTEMA DE ATENCIÓN MÉDICA. Beneficios para estudiantes de la materia Ingeniería de Software y la comunidad con una aplicación móvil de servicios médicos. Revista Iberoamericana para la Investigación y el Desarrollo Educativo Recuperado de https://www.ride.org.mx/index.php/RIDE/article/view/773/2657

[17] Microsoft (2023). Información general de App Service.Recuperado de: https://learn.microsoft.com/es-es/azure/app-service/overview

[18] Montes, A. (2018), "Clean architecture para mejorar el desarrollo de aplicaciones móviles en la empresa GMD", Trabajo de grado, Univ. Nac. Mayor de San Marcos. https://hdl.handle.net/20.500.12672/10218

[19] Pantoja, B. (2023). "El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing". Acta Nova, 2(4), 493–507. http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S1683-07892004000100005

[20] Reinoso, A. & Zhirzhan, C. (2023). Desarrollo De Una Aplicación Móvil Para El Agendamiento De Citas De Consultas Médicas Utilizando Técnicas De Procesamiento De Lenguaje Natural Aplicadas A Un Asistente Virtual. Cuenca, Ecuador. Recuperado de https://dspace.ups.edu.ec/bitstream/123456789/22064/1/UPS-CT009620.pdf

[21] Research2Guidance. (2017). MOBILE HEALTH MARKET REPORT 2013-2017: The Commercialization of mhealth Applications. Berlin.

[22] Souza, F., Santos, W, Dantas, J., Sousa, H., Moreira, O., & Silva, R. (2022). Desenvolvimento de aplicativo móvel para o acompanhamento pré-natal e validação de conteúdo. Acta Paulista de Enfermagem, 35. https://doi.org/10.37689/acta-ape/2022ao01861

[23] Tung, B. (2017). "Reactive Programming and Clean Architecture in Android Development," (Bachelor of Engineerin), Helsinki Metropolia University of Applied Sciences https://www.theseus.fi/handle/10024/126982

[24] Velásquez, A. (2018). Investigación en políticas y sistemas de salud para la gestión basada en evidencias. Rev Peru Med Exp Salud Pública. 2018;35(3):371-2. DOI: 10.17843/rpmesp.2018.353.3978 https://doi.org/10.17843/rpmesp.2018.353.3978

[25] Villalobos, D., Díaz, C. (2022). Modelo arquitectural de servicio al cliente basado en tecnologías cloud para estéticas veterinarias implementado en una aplicación móvil. Caso de estudio: Servicio al Cliente en la Estética Veterinaria "Huellitas y Pelitos. Universidad Distrital Francisco José de Caldas. Obtenido de: https://repository.udistrital.

edu.co/bitstream/handle/11349/30438/DiazSerranoCamiloAndres2022.pdf?sequence=1&isAllowed=y

[26] Zhang, Z. Q., Wang, F., Xiao, B. J., Ma, J., & Yang, F. (2023). A new remote web-based MDSplus data visualization system for EAST. Fusion Engineering and Design, 186(113337), 113337. https://doi.org/10.1016/j.fusengdes.2022.113337

[27] Zhu, J., Cyr, A. (2018). The Use of Mobile Health Applications to Improve Patient Experience: Cross-Sectional Study in Chinese Public Hospitals. DOI: https://doi.org/10.2196%2Fmhealth.9145

Financiamiento:

Propio.

Conflictos de interés:

Los autores declaran no tener conflictos de interés.

Contribuciones de autoría:

José Bryan Calle Soraluz: Desarrollo la parte del Backend de la solución y redacto el articulo científico.

Luiggi Steep Pasache Lopera: Desarrollo la parte del Frontend de la solución y redacto el articulo científico.

Ivan Carlo Petrlik Azabache: Asesoro y reviso el articulo científico y la solución móvil y web.