Acta de Correcciones al Documento de Trabajo de Grado

Santiago de Cali, 18 de julio de 2024

Autor: Javier Augusto Murcia Rodríguez

Título del Trabajo de Grado: Desarrollo de un prototipo de Super App multiplataforma para una institución de salud de alta complejidad

Director: Juan Carlos Martínez Arias

Como indica el artículo 2.13 de las Directrices para Trabajo de Grado de Maestría, he verificado que el estudiante indicado arriba ha implementado todas las correcciones que los Jurados del Proyecto de Trabajo de Grado definieron que se efectuaran, como consta en el Acta de Evaluación correspondiente.

Firma del Director del Trabajo de Grado

Santiago de Cali, 16 de Mayo de 2024

Ingeniera:
Luisa Fernanda Rincón Pérez
Directora Maestría Ingeniería de Software
Facultad de Ingeniería y Ciencias
Pontificia Universidad Javeriana - Cali

Con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el Trabajo de Grado y posteriormente optar por el título de Magíster en Ingeniería de Software, nos permitimos presentar a su consideración el proyecto de Trabajo de Grado denominado Desarrollo de un prototipo de Super App multiplataforma para una institución de salud de alta complejidad, el cual será realizado por el estudiante Javier Augusto Murcia Rodriguez con código 02145701106 perteneciente al énfasis en Software, bajo la dirección del profesor Juan Carlos Martínez Arias.

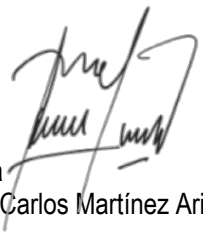
El suscrito director del Trabajo de Grado autoriza para que se proceda a hacer la evaluación de este Proyecto ante el Tribunal que para el efecto se designe, toda vez que ha revisado cuidadosamente el documento y avala que ya se encuentra listo para ser presentado oficialmente.

Atentamente,

Javier Murcia

Firma
Javier Augusto Murcia Rodriguez

C.C. 1.130.682.656 de Cali



Firma
Juan Carlos Martínez Arias

C.C. 7.549.319 de Armenia

FICHA RESUMEN
TRABAJO DE GRADO DE MAESTRÍA

TITULO: "Desarrollo de un prototipo de Super App multiplataforma para una institución de salud de alta complejidad."

1. ÉNFASIS: Ingeniería de Software
2. TIPO DE PROYECTO: Aplicado
3. ÁREA DE TRABAJO: Software
4. ESTUDIANTE: Javier Augusto Murcia Rodriguez
5. CORREO ELECTRÓNICO: ingjaviermurcia@gmail.com
6. DIRECCIÓN Y TELÉFONO: Carrera 85D # 48 - 56 Apto 407 / 3194960012
7. DIRECTOR: MSc. Juan Carlos Martínez Arias
8. VINCULACIÓN DEL DIRECTOR (en la universidad): Planta
9. CORREO ELECTRÓNICO DEL DIRECTOR: juancmartinez@javerianacali.edu.co
10. GRUPO O EMPRESA QUE LO AVALA: Fundación Valle del Lili
11. PALABRAS CLAVE (al menos 5): Apps, Mobile, Super Apps, Requerimientos, Pruebas, Salud, Arquitectura.
12. FECHA DE INICIO (Desarrollo del proyecto): 1/08/2023
13. FECHA FINAL (Desarrollo del proyecto): 16/05/2024
14. RESUMEN: Este proyecto abordó el desarrollo de una Super App para la Fundación Valle del Lili, institución de salud de alta complejidad que enfrenta numerosas solicitudes para el desarrollo de aplicaciones móviles, destinadas a mejorar la comunicación y la atención al paciente. Se diseñó y desarrolló una Super App enfocada en los atributos de calidad de integrabilidad, modularidad, seguridad siendo compatible con Android y iOS, que inicio como un prototipo y evolucionó para su lanzamiento en un entorno productivo, convirtiéndose en una iniciativa prioritaria. El software resolvió la problemática inicial e introdujo mejoras significativas como la unificación de contraseñas, actualización centralizada de datos de contacto de los pacientes, buscando optimizar la experiencia del usuario y la eficiencia operativa de la institución. Durante el desarrollo, se documentaron los requerimientos del software, se implementaron medidas de seguridad robustas, y se estructuró y ejecutó un plan de pruebas y validación. La metodología de arquitectura de software adoptada permitió una mejora sistemática de la solución, estableciéndose como estándar para futuros desarrollos de software en la institución. La integración exitosa de la primera Micro App demostró la capacidad técnica de la Super App. Gracias al uso de tecnologías como WebView, que facilitaron la integración de las URLs públicas de las Micro Apps, fue posible su ejecución dentro y fuera del entorno de la Super App, permitiendo el acceso desde cualquier dispositivo, buscando optimizar el mantenimiento y consumo de recursos. Además, este documento pretende servir como guía y modelo teórico para la institución y otros interesados, destacando que las decisiones adoptadas fueron definidas de manera rigurosa y estricta, basándose tanto en la teoría, como en decisiones tomadas en conjunto por diversas áreas de la institución. Este enfoque colaborativo y fundamentado aseguró que cada paso del proyecto estuviera alineado con las necesidades prácticas y los principios teóricos de la ingeniería de software, buscando que esta sea una solución robusta y sostenible para la institución.

Resumen

Este proyecto abordó el desarrollo de una Super App para la Fundación Valle del Lili, institución de salud de alta complejidad que enfrenta numerosas solicitudes para el desarrollo de aplicaciones móviles, destinadas a mejorar la comunicación y la atención al paciente. Se diseñó y desarrolló una Super App enfocada en los atributos de calidad de integrabilidad, modularidad, seguridad siendo compatible con Android y iOS, que inicio como un prototipo y evolucionó para su lanzamiento en un entorno productivo, convirtiéndose en una iniciativa prioritaria. El software resolvió la problemática inicial e introdujo mejoras significativas como la unificación de contraseñas, actualización centralizada de datos de contacto de los pacientes, buscando optimizar la experiencia del usuario y la eficiencia operativa de la institución.

Durante el desarrollo, se documentaron los requerimientos del software, se implementaron medidas de seguridad robustas, y se estructuró y ejecutó un plan de pruebas y validación. La metodología de arquitectura de software adoptada permitió una mejora sistemática de la solución, estableciéndose como estándar para futuros desarrollos de software en la institución. La integración exitosa de la primera Micro App demostró la capacidad técnica de la Super App. Gracias al uso de tecnologías como WebView, que facilitaron la integración de las URLs públicas de las Micro Apps, fue posible su ejecución dentro y fuera del entorno de la Super App, permitiendo el acceso desde cualquier dispositivo, buscando optimizar el mantenimiento y consumo de recursos.

Además, este documento pretende servir como guía y modelo teórico para la institución y otros interesados, destacando que las decisiones adoptadas fueron definidas de manera rigurosa y estricta, basándose tanto en la teoría, como en decisiones tomadas en conjunto por diversas áreas de la institución. Este enfoque colaborativo y fundamentado aseguró que cada paso del proyecto estuviera alineado con las necesidades prácticas y los principios teóricos de la ingeniería de software, buscando que esta sea una solución robusta y sostenible para la institución.

Palabras Clave Apps, Móvil, Super Apps, Requerimientos, Pruebas, Salud.

Abstract

This project addressed the development of a Super App for the Valle del Lili Foundation, a high-complexity health institution facing numerous requests for the development of mobile applications aimed at improving communication and patient care. A Super App focused on quality attributes such as integrability, modularity, and security, compatible with Android and iOS, was designed and developed. It started as a prototype and evolved for deployment in a productive environment, becoming a priority initiative. The software resolved the initial problems and introduced significant improvements such as password unification, centralized updating of patient contact data, seeking to optimize user experience and operational efficiency of the institution.

During the development, software requirements were documented, robust security measures were implemented, and a structured testing and validation plan was executed. The adopted software architecture methodology allowed for systematic improvement of the solution, establishing it as a standard for future software developments within the institution. The successful integration of the first Micro App demonstrated the technical capability of the Super App. Thanks to technologies such as WebView, which facilitated the integration of public URLs of the Micro Apps, execution was possible both inside and outside the Super App environment, allowing access from any device and aiming to optimize maintenance and resource consumption.

Furthermore, this document aims to serve as a guide and theoretical model for the institution and other stakeholders, highlighting that the adopted decisions were defined rigorously and strictly, based on both theory and joint decisions made by various areas of the institution. This collaborative and grounded approach ensured that each step of the project was aligned with the practical needs and theoretical principles of software engineering, seeking to make this a robust and sustainable solution for the institution.

Keywords Apps, Mobile, Super Apps, Requirements, Tests, Health

Pontificia Universidad
JAVERIANA
Cali

Desarrollo de un prototipo de Super App multiplataforma para una institución de salud de alta complejidad

Javier Augusto Murcia Rodriguez

Proyecto presentado como requisito para optar al título de:
Magister en Ingeniería de Software

Director:
MSc Juan Carlos Martínez Arias

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Departamento de Electrónica y Ciencias de la Computación
Cali, Colombia
17 de julio de 2024

Agradecimientos

Quiero expresar mi sincero agradecimiento a mi familia por su apoyo constante. En particular, resalto mi profunda gratitud hacia mi esposa, Jessica, quien ha sido una fuente de inspiración y apoyo incondicional en cada paso de este camino. Extiendo igualmente mi gratitud a todos los profesores y compañeros, así como a las personas que de alguna manera colaboraron en este proceso. Un agradecimiento especial a mi director de proyecto, el profesor Juan Carlos, por su invaluable orientación y apoyo a lo largo de este proyecto.

Índice

Índice de figuras

Índice de tablas

1. Introducción

La industria del software se caracteriza por su rápida y constante evolución, lo que plantea desafíos significativos para las industrias que dependen de estas tecnologías. Cada día se crean, actualizan o descontinúan aplicaciones de software, y las industrias deben mantenerse al día y adaptarse a estos cambios para conservar su competitividad y minimizar la deuda técnica. Además, los usuarios de estas aplicaciones frecuentemente evalúan cómo las nuevas tecnologías pueden integrarse en sus procesos diarios facilitando la ejecución de sus actividades.

En la industria de la salud, estos desafíos son particularmente complejos, pues, aunque el desarrollo de software no es la actividad principal de las instituciones prestadoras de servicios de salud; estas deben garantizar la inclusión de innovación tecnológica en sus procesos y que cualquier nuevo desarrollo se realice con precaución buscando minimizar los riesgos, cumplir con las regulaciones legales, teniendo en cuenta las restricciones presupuestarias y de desarrollo.

La Fundación Valle del Lili, una institución de salud de alta complejidad enfrenta una demanda significativa de aplicaciones móviles diseñadas para mejorar los procesos y la atención al paciente. En este proyecto, se establecieron objetivos claros para abordar las necesidades variadas de los servicios de la institución. Se centró en analizar y responder de manera progresiva a estas necesidades mediante el desarrollo de una Super App modular, escalable y segura, compatible con iOS y Android. Esta aplicación no solo atiende las demandas actuales, sino que está diseñada para adaptarse a las futuras necesidades de la institución. La meta principal fue garantizar la integrabilidad, modularidad, escalabilidad y seguridad utilizando diversas herramientas de ingeniería de software, logrando una solución que se integra perfectamente con las operaciones de la institución. Además, la Super App, con sus características de rigurosidad teórica para la toma de decisiones durante su diseño y desarrollo, proporciona una solución transversal que podría ser aplicada en otros sectores enfrentando desafíos similares.

2. Definición del problema

2.1. Planteamiento del problema

Actualmente, la necesidad de realizar desarrollos de software es alta, lo que se ve reflejado en la alta demanda de profesionales que suplan esta necesidad, esto, se puede comprobar en el artículo de (Hylton et al., 2022), donde se pronosticó una alta demanda para estos profesionales en la próxima década. Sin embargo, existen algunos fenómenos como la pandemia que han aumentado aún más la necesidad de profesionales en el campo de las tecnologías de la información (TI), otorgando una mayor relevancia frente a otras profesiones.

De acuerdo con este estudio, se espera que la demanda de desarrollos de software y por consiguiente de las ocupaciones relacionadas con la informática experimenten un crecimiento del 13,4 % durante la década actual (2020-2030), lo cual es 5,7 % más rápido que el promedio del 7,7 % para todas las ocupaciones. Considerando que el desarrollo de software puede ser ejecutado o consumido desde múltiples plataformas, como, por ejemplo, el desarrollo de software de aplicaciones móviles; este tipo de desarrollo de software ha presentado un aumento en su demanda de manera particular, pues como lo indica Malavolta et al. (2015) hace casi una década que "los dispositivos móviles, por su carácter utilitario, facilidad de uso y accesibilidad, se han convertido en el medio más popular e indispensable para cubrir las necesidades humanas en los últimos años". Los dispositivos móviles han cobrado mayor relevancia en la actualidad y se han convertido en una necesidad básica para las personas, siendo una herramienta muy valiosa para interactuar de manera rápida y efectiva mediante las aplicaciones de software que han sido desarrolladas e instaladas en los dispositivos.

La Fundación Valle del Lili, una institución prestadora de servicios de salud de alta complejidad, se enfrenta a una demanda creciente por parte de sus servicios de aplicaciones móviles, que buscan mejorar los procesos y la atención al paciente. A pesar de contar con más de 125 servicios, la institución no dispone con una aplicación móvil o un medio tecnológico que permita manejar y gestionar esta creciente demanda de soluciones de una manera organizada, escalable y sostenible. Dado que el desarrollo

de software no es su actividad principal y los recursos son limitados, no era viable desarrollar una aplicación móvil para cada servicio, ni era práctico para los pacientes tener que instalar una aplicación diferente para cada uno.

Frente a la problemática planteada, se propuso una solución: crear una aplicación móvil tipo Super App para la Fundación Valle del Lili. Dicha aplicación posee la habilidad de consolidar diversas Apps / Micro Apps en un único entorno, ofreciendo una respuesta fexible y adecuada a las necesidades de los variados servicios de la institución, dado que se pueden añadir de forma gradual a la plataforma. Además, esta herramienta brinda a los pacientes una interacción directa y eficiente con la institución mediante una sola interfaz gráfica.

2.2. Formulación del problema

En instituciones de salud como la Fundación Valle del Lili, cuyo enfoque principal no es el desarrollo de aplicaciones móviles, se enfrentan a la necesidad de contar con una gran cantidad de estas. Esto plantea varias preguntas:

- ¿Qué enfoque o metodología se debería adoptar para poder obtener, analizar, especificar, validar y administrar los diferentes requerimientos de las aplicaciones móviles para los distintos servicios?
- ¿Sería posible desarrollar una única aplicación móvil que sea escalable y modular, de manera que, aunque internamente esté compuesta por múltiples aplicaciones, para el paciente se presente como una única interfaz gráfica?
- En caso de que sea posible desarrollar una aplicación con las características mencionadas anteriormente, ¿cómo se lograría la integración entre las diferentes aplicaciones, dando prioridad al atributo de calidad de seguridad debido a la importancia de la información que se manejaría? Además, ¿sería posible realizar pruebas y qué tipos de pruebas se podrían aplicar?

3. Objetivos del proyecto

3.1. Objetivo General

Desarrollar un prototipo de software para dispositivos móviles que sea modular y multiplataforma (iOS - Android) tipo Super App, para una institución de salud de alta complejidad que garantice modularidad, integración y seguridad en la gestión de la información.

3.2. Objetivos específicos

1. Analizar y documentar los requerimientos específicos de software considerados como prioritarios por la institución de salud de alta complejidad, con el fin de entender las funciones y características necesarias del prototipo de software a desarrollar.

2. Diseñar y desarrollar un prototipo de software modular y multiplataforma (iOS - Android) que se adapte a las necesidades priorizadas por la institución de salud y que permita la integración gradual de los diferentes desarrollos requeridos por los servicios, enfocado a Super Apps.

3. Definir y documentar los componentes tecnológicos y la arquitectura de integración que permitan la incorporación efectiva y segura de diversas aplicaciones (Apps / Micro Apps) al prototipo de Super App propuesto. La seguridad será uno de los atributos de calidad primordiales debido a la relevancia de la información que se manejará.

4. Establecer un plan de pruebas y validación para garantizar que el prototipo de software cumple con los requerimientos identificados por la institución de salud de alta complejidad.

4. Marco teórico de referencia y antecedentes

4.1. Bases Teóricas

4.1.1. Requerimiento en ingeniería

Una afirmación que identifica una característica o restricción operativa, funcional o de diseño de un producto o proceso, que es inequívoca, comprobable o medible, y necesaria para la aceptabilidad del producto o proceso (según los consumidores o las pautas internas de aseguramiento de la calidad) (IEEE, 1999).

4.1.2. Ingeniería de requerimientos

Es un subconjunto de la ingeniería de sistemas que se ocupa de descubrir, desarrollar, rastrear, analizar, calificar, comunicar y gestionar los requisitos que definen el sistema en sucesivos niveles de abstracción (Dick et al., 2017).

4.1.3. Institución prestadora de salud (IPS) de alta complejidad

Las instituciones prestadoras de salud son entidades oficiales, mixtas, privadas, comunitarias y solidarias, organizadas para la prestación de los servicios de salud a los afiliados del Sistema General de Seguridad Social en Salud dentro de las entidades promotoras de salud o fuera de ellas. El Estado podrá establecer mecanismos para el fomento de estas organizaciones y abrir líneas de crédito para la organización de grupos de práctica profesional y para las instituciones prestadoras de servicios de tipo comunitario y solidario (de la Función Pública, 1993). Sé denomina alta complejidad a nivel de atención III y IV: urgencias y servicios especializados, hospitalización de mayor complejidad, y atención especializada (de salud Colombia, 2023).

4.1.4. Prototipo de Software:

El prototipado puede ser empleado, y ayuda a identificar lagunas e incomprensiones en la definición de los requisitos. El prototipo es una versión temprana funcional

del sistema, se utiliza para dar a los usuarios una idea de cómo será el sistema funcional, y su evaluación por parte de los interesados ayuda a aclarar los requisitos. El prototipo puede ser desechado al final del prototipado, o puede ser reutilizado en el desarrollo del sistema. El prototipado implica segun O'regan (2017):

- Definir los objetivos del prototipo.
- Decidir que requisitos funcionales serán prototipados.
- Desarrollar el prototipo.
- Evaluar el prototipo.

4.1.5. Aplicaciones móviles multiplataforma o cross-platform apps

Cuando los desarrolladores de aplicaciones móviles escriben código nativo, interactúan con el sistema operativo (SO) a través de interfaces de programación de aplicaciones (APIs) expuestas por el propio SO o por marcos de desarrollo adicionales específicos de la plataforma (a los que juntos denominamos APIs del sistema). Normalmente, cada plataforma tiene su propio conjunto de APIs del sistema, a las que se accede a través de lenguajes de programación específicos de la plataforma. Por lo tanto, las aplicaciones destinadas a múltiples plataformas generalmente necesitan ser desarrolladas por separado para cada plataforma diferente. Los marcos de desarrollo multiplataforma (MDMP) abordan este problema encapsulando las APIs del sistema para diferentes plataformas en sus propias APIs, que son consistentes entre las diferentes plataformas y se pueden acceder a través de un solo lenguaje de programación. De esta manera, utilizando los MDMP, los desarrolladores pueden escribir su código solo una vez y tener aplicaciones funcionales para ambas plataformas móviles principales (iOS y Android). Dado que los MDMP encapsulan muchas de las APIs del sistema comúnmente utilizadas, se ha demostrado que reducen los costos de desarrollo y mantenimiento de los proyectos (Mascetti et al., 2021).

4.1.6. Super Apps

Dada la relevancia del concepto de "Super App" en el contexto de este proyecto, resulta fundamental comprender y definir correctamente este término. Por lo tanto, se ha optado por considerar las siguientes definiciones

- El término "Super Apps" fue introducido en 2010 por Mike Lazaridis, fundador de BlackBerry. Lazaridis definió una "Super App" como "un ecosistema cerrado de numerosas aplicaciones", las cuales son utilizadas a diario debido a su conveniencia y a la experiencia integrada, contextualizada y eficiente que proporcionan. Las Super Apps actúan como marketplaces que ofrecen una amplia variedad de productos y servicios. Consolidan las funcionalidades de diversas aplicaciones individuales, integrándolas en una única aplicación que sirve como paraguas para numerosos servicios. Las súper aplicaciones mejoran la experiencia del usuario debido a que poseen una cantidad de datos del cliente sin precedentes, lo que les permite ofrecer experiencias personalizadas y tratos especiales. Estas eliminan la necesidad de iniciar sesión en diferentes aplicaciones y, además, permiten ahorrar espacio en el teléfono (Baquero, 2021).

- A medida que las empresas de tecnología se han vuelto más omnipresentes, su incursión en las líneas tradicionales de negocio también se ha vuelto inevitable. En este contexto, ha surgido el término Super-apps, es decir, aplicaciones móviles que en el mismo entorno buscan satisfacer diferentes necesidades diarias de los consumidores sin requerir que descarguen otra aplicación. En otras palabras, las Super-apps desempeñan el papel de un mercado o ecosistema que alberga en sí mismo diferentes tipos de soluciones, servicios y experiencias que tradicionalmente solo se encontrarían en una aplicación específicamente diseñada para ello. Aunque el término Super-apps comenzó a desarrollarse principalmente en Asia con gigantes como WeChat y Alipay, hoy en día más empresas buscan ofrecer estas soluciones todo-en-uno para satisfacer diversas necesidades del cliente. Estas soluciones ofrecidas suelen abarcar desde el comercio electrónico hasta la entrega de bienes, servicios financieros y redes sociales. Además, estas aplicaciones móviles apuntan a un gran número de necesidades cotidianas de

los consumidores, y en la esfera financiera ofrecen servicios tradicionalmente disponibles a través de los bancos (Roa et al., 2021).

4.1.7. Arquitectura de software y arquitectura de integración

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución (IEEE, 2000).

La arquitectura de integración es una arquitectura de software que facilita la integración de múltiples componentes de TI. Esta arquitectura cambia con los avances en la utilidad multiplataforma y otros paradigmas de desarrollo para nuevos tipos de operaciones digitales. En algunos sentidos, la integración trata de "descomponer silos" y ayudar a diferentes programas de software a comunicarse. La incrustación de una aplicación en un contexto más amplio puede requerir herramientas especializadas, como interfaces de programación de aplicaciones (API), hechas específicamente para permitir este tipo de integración. Las arquitecturas basadas en la nube y otros tipos de nuevas opciones se están volviendo populares para su uso en arquitecturas de integración. Utilizando herramientas como API, middleware y otros recursos, los ingenieros ensamblan arquitecturas viables que integran con éxito sus muchas partes. Cuando se aplica a los negocios, esto a menudo se llama integración de aplicaciones empresariales y se realiza para apoyar objetivos comerciales clave. Otra forma de pensar en la arquitectura de integración es como el "esqueleto" de un sistema de TI o, como lo llaman algunos expertos, la "plomería" de un sistema. Existen diferentes métodos, como la integración punto a punto, y diferentes "topologías" para la integración que marcan una diferencia en el diseño (Techopedia, 2023), ademas, segun Microsoft (2023) el propósito de la integración es conectar aplicaciones, datos, servicios y dispositivos, a menudo de formas complejas. A través de la integración, las organizaciones unen flujos de trabajo para que sean coherentes y escalables. Las empresas conectan aplicaciones, datos y procesos de manera rápida, eficiente y automatizada. Las conexiones pueden ejecutarse entre sistemas locales, en la nube y en sistemas periféricos. Pueden reunir tecnologías empresariales, de socios, de ter-

ceros y heredadas. En el caso de los datos, la integración ofrece soluciones para recopilar y procesar información de múltiples fuentes, en múltiples formatos. Para integrar aplicaciones, a veces son adecuadas las llamadas directas a API. Pero a veces las tecnologías necesitan comunicarse de forma asíncrona, a través de mensajería o eventos. Todos los procesos de integración necesitan orquestación, una forma sencilla de definir y ejecutar la lógica del flujo de trabajo.

4.1.8. Pruebas de software

La prueba de software no sólo se utiliza para encontrar errores o fallos en el software, sino también para mejorar la calidad del mismo. Ofrece una solución rentable a todos los problemas y proporciona un producto mejorado al cliente en términos de funcionalidad, precisión, seguridad de datos, etc. Es un proceso iterativo que continúa incluso después de que se ha completado el producto.

El principal objetivo de las pruebas de software es verificar y validar la integridad del producto. Este debe cumplir con los requisitos técnicos establecidos. Los fallos técnicos deben ser reportados adecuadamente por el tester, quien también debe asegurarse de que el software esté libre de fallos antes de su lanzamiento al mercado. Los testers tienen que generar casos de prueba de alta calidad y emitir informes correctos de problemas.

La prueba de software ahorra mucho esfuerzo, dinero y tiempo a las organizaciones que desarrollan y venden el producto. Además, ofrece optimización empresarial. Confirma que una aplicación o producto es capaz de operar correctamente y sin problemas en todas las condiciones requeridas, en todos los tipos diferentes de sistemas operativos y en los navegadores web de manera adecuada y correcta. Esto mejora la experiencia del usuario y la satisfacción del cliente. Todos estos hechos contribuyen a la generación de ingresos, por lo que aportan beneficios a la organización.

En términos sencillos, para asegurarse de que el software lanzado es seguro y funciona como se esperaba, se introdujo el concepto de prueba de software. Es importante porque los fallos, errores y defectos del software pueden potencialmente llevar a costos excesivos y consecuencias peligrosas.

La prueba de software es una tarea intelectual altamente creativa que los testers deben realizar. Hay principios que todos los testers deben conocer mientras realizan pruebas de software. Se encuentran involucrados diferentes problemas en las pruebas de software, como la escalabilidad, usabilidad, seguridad, etc (Taley, 2020).

4.1.9. React

React es una biblioteca para crear interfaces de usuario tanto para aplicaciones web como nativas. Permite a los desarrolladores construir interfaces a partir de componentes individuales, como Thumbnail, LikeButton y Video, y combinarlos en pantallas, páginas y aplicaciones completas. Estos componentes son funciones de JavaScript y pueden incluir código y marcado, utilizando una sintaxis llamada JSX. Esta extensión de sintaxis de JavaScript, popularizada por React, facilita la creación, mantenimiento y eliminación de componentes. React no solo se utiliza para construir páginas completas, sino que también se puede agregar a páginas HTML existentes para renderizar componentes interactivos. Además, React funciona como una arquitectura, permitiendo a los desarrolladores construir aplicaciones web y nativas con las mismas habilidades, y aprovechando las fortalezas únicas de cada plataforma. Más que una biblioteca o arquitectura, React representa una comunidad donde desarrolladores y diseñadores de diversos orígenes crean interfaces de usuario juntos. Además de lo anterior que es resumido de la página de los desarrolladores de la librería ellos mismos describen React como, React es una herramienta poderosa y versátil para el desarrollo de interfaces de usuario, capaz de abordar tanto aplicaciones web como nativas con un enfoque modular y colaborativo (Meta Open Source, 2023).

4.1.10. Webpack

Webpack es un empaquetador de módulos estático diseñado para aplicaciones JavaScript modernas. Este procesa la aplicación construyendo un gráfico de dependencias que inicia desde uno o más puntos de entrada. A partir de aquí, combina cada módulo necesario en uno o más paquetes, conocidos como bundles, que sirven como activos estáticos desde los cuales se distribuye tu contenido. Webpack trans-

forma y empaqueta una variedad de recursos y archivos, como JavaScript, HTML, CSS e imágenes, tomando módulos con dependencias para generar archivos estáticos que los representen. Esto resulta particularmente valioso en aplicaciones grandes y complejas que utilizan muchos módulos JavaScript y otros recursos, ya que Webpack los organiza eficientemente para optimizar su carga en un navegador (Webpack JS, 2023a).

4.1.11. Re.Pack

Re.Pack es un conjunto de herramientas de código abierto que utiliza Webpack 5 y el sistema de plugins de React Native CLI para permitir a los desarrolladores empaquetar aplicaciones React Native utilizando características de Webpack. Hace que Webpack entienda las bases de código de React Native y produzca paquetes que pueden ser ejecutados por aplicaciones React Native.

El conjunto de herramientas proporciona a los desarrolladores el ecosistema completo de plugins y cargadores de Webpack que pueden ser utilizados en la construcción de aplicaciones móviles avanzadas que se benefician de la división de código, incluyendo: Super apps, Mini apps, Aplicaciones modulares, Aplicaciones con rendimiento deficiente y Aplicaciones que requieren soluciones personalizadas avanzadas (Callstack, 2023a).

4.1.12. Module Federation

Module Federation es una funcionalidad de Webpack que permite construir aplicaciones unificadas a partir de múltiples compilaciones separadas. Estas compilaciones actúan como contenedores y pueden exponer y consumir código entre sí. Se diferencia entre módulos locales (parte de la compilación actual) y módulos remotos (cargados en tiempo de ejecución desde un contenedor remoto). La operación de cargar módulos remotos se considera asíncrona, y se realiza a través de operaciones de carga de segmentos, como llamadas import().

Webpack crea contenedores a través de una entrada de contenedor, que expone acceso asíncrono a módulos específicos en dos pasos: carga del módulo (asíncrona) y

evaluación del módulo (sincrónica). Las herramientas básicas incluyen ContainerPlugin, que crea una entrada adicional de contenedor con módulos expuestos, y ContainerReferencePlugin, que permite importar módulos remotos de estos contenedores. ModuleFederationPlugin combina estas dos funcionalidades.

Los módulos compartidos en Module Federation son módulos que son tanto reemplazables como provistos como reemplazos a contenedores anidados. Por ejemplo, en aplicaciones de una sola página, cada página puede ser una compilación separada y parte de un shell de aplicación que referencia todas las páginas como módulos remotos. Además, se puede configurar contenedores remotos dinámicos, donde se resuelven promesas en tiempo de ejecución para determinar qué módulos se deben cargar desde un contenedor remoto (Webpack JS, 2023b).

4.1.13. Importancia de la arquitectura de software

Según Bass et al. (2022, Capítulo 2), Se pueden usar o tener en cuenta las siguientes razones para motivar la creación de una nueva arquitectura, o el análisis y evolución de la arquitectura de un sistema:

1. Una arquitectura puede inhibir o habilitar los atributos de calidad principales de un sistema.

2. Las decisiones tomadas en una arquitectura te permiten razonar y gestionar los cambios a medida que el sistema evoluciona.

3. El análisis de una arquitectura permite predecir anticipadamente las cualidades de un sistema.

4. Una arquitectura documentada mejora la comunicación entre las partes interesadas.

5. La arquitectura es portadora de las decisiones de diseño más tempranas, y por lo tanto, más fundamentales y difíciles de cambiar.

6. Una arquitectura define un conjunto de restricciones para la implementación subsiguiente.

7. La arquitectura dicta la estructura de una organización, o viceversa.
8. Una arquitectura puede proporcionar la base para el desarrollo incremental.
9. Una arquitectura es el artefacto clave que permite al arquitecto y al gerente de proyecto razonar sobre el costo y el cronograma.
10. Una arquitectura puede crearse como un modelo transferible y reutilizable que forma el corazón de una línea de productos.
11. El desarrollo basado en arquitectura se centra en el ensamblaje de componentes, en lugar de simplemente en su creación.
12. Al restringir alternativas de diseño, la arquitectura canaliza la creatividad de los desarrolladores, reduciendo la complejidad del diseño y del sistema.
13. Una arquitectura puede ser la base para la capacitación de un nuevo miembro del equipo.

4.1.14. Especificación de requisitos de atributos de calidad y decisiones de diseño arquitectónico

Según lo resumido Bass et al. (2022, Capítulo 3) En la ingeniería de software, se distingue entre requisitos funcionales y requisitos de atributos de calidad. Los requisitos funcionales se satisfacen incluyendo un conjunto apropiado de responsabilidades dentro del diseño, mientras que los requisitos de atributos de calidad se logran a través de las estructuras y comportamientos de la arquitectura. Para capturar y expresar un requisito de atributo de calidad, se recomienda el uso de escenarios de atributos de calidad, que constan de seis partes esenciales:

- **Fuente del estímulo:** El origen del evento que impacta el sistema.
- **Estímulo:** El evento concreto que requiere una respuesta del sistema.
- **Entorno:** El estado del sistema en el momento de recibir el estímulo.

- **Artefacto:** La parte del sistema afectada por el estímulo.
- **Respuesta:** La actividad que el sistema debe llevar a cabo como resultado del estímulo.
- **Medida de la respuesta:** Criterios para evaluar la respuesta del sistema.

Un desafío común en el diseño arquitectónico es la captura inadecuada de estos requisitos. Para superar esto, se emplean tácticas y patrones arquitectónicos. Una táctica arquitectónica es una decisión de diseño que influye en la respuesta del sistema a un atributo de calidad. Se centra en mejorar un atributo de calidad específico.

Por otro lado, un patrón arquitectónico aborda problemas recurrentes de diseño que surgen en contextos específicos y ofrece soluciones arquitectónicas probadas. Los patrones arquitectónicos se pueden entender como conjuntos de tácticas, cada una de las cuales puede abordar múltiples atributos de calidad.

Para comprender las decisiones tomadas en una arquitectura, se puede utilizar una lista de verificación basada en tácticas. Esta técnica de análisis de arquitectura ligera puede proporcionar información sobre las fortalezas y debilidades de la arquitectura en un tiempo muy corto.

En conjunto, los escenarios de atributos de calidad, junto con tácticas y patrones arquitectónicos, forman un marco para el diseño de sistemas que no solo cumplan con sus funciones previstas sino que también posean las cualidades necesarias para satisfacer las expectativas de los usuarios y resistir los desafíos operativos y ambientales. A continuación se puede ver un listado algunos atributos de calidad no solo mencionados en el libro del autor ya citado sino también explicados en caso de querer profundizar:

- Disponibilidad (Availability)
- Desplegabilidad (Deployability)
- Eficiencia energética (Energy Efficiency)
- Integrabilidad (Integrability)

- Modificabilidad (Modifiability)
- Rendimiento (Performance)
- Protección (Safety)
- Seguridad (Security)
- Testabilidad (Testability)
- Usabilidad (Usability)

4.1.15. Requisitos Arquitectónicamente Significativos (ASRs)

En el marco de la arquitectura de sistemas de software, los Requisitos Arquitectónicamente Significativos (ASRs) desempeñan un papel crucial. Según Bass et al. (2022, Capítulo 19), un ASR se define como un requisito que ejerce un impacto profundo y medible en la arquitectura de un sistema de software. La ausencia de un ASR podría resultar en una arquitectura notablemente distinta. Aunque comúnmente asociados con atributos de calidad (QA) como rendimiento, seguridad, modificabilidad, disponibilidad y usabilidad, los ASRs no se limitan a estos y pueden abarcar tanto requisitos funcionales como no funcionales, estos son considerados importantes en dos aspectos:

- **Impacto en la arquitectura:** Los ASRs influyen de manera significativa en la arquitectura del sistema. Esta influencia se manifiesta en la selección de patrones y tácticas arquitectónicas, orientadas específicamente a cumplir con los requisitos de QA. Un requisito de QA particularmente complejo o crítico es más propenso a ser un ASR debido a su potencial para alterar significativamente la estructura arquitectónica del sistema.
- **Identificación de ASRs:** La identificación de ASRs es una tarea que requiere un análisis detallado y una colaboración estrecha con los stakeholders. Los arquitectos deben emprender un esfuerzo significativo para descubrir posibles

ASRs, a menudo iniciando este proceso mediante conversaciones con stakeholders clave para comprender a fondo las necesidades y objetivos del proyecto.

Las fuentes posibles para la identificación de ASRs pueden ser:

- **Documentos de requisitos:** Estos documentos son una fuente primordial para la identificación de ASRs, aunque rara vez presentan estos requisitos de manera explícita. Por lo tanto, los arquitectos deben emplear un enfoque analítico para interpretar y extraer ASRs de la documentación disponible.
- **Objetivos de negocio:** Los ASRs frecuentemente derivan de los objetivos comerciales de la organización. Comprender estas metas de negocio es esencial, ya que pueden influir o dictar requisitos que son críticos desde una perspectiva arquitectónica.

4.1.16. ADD (Attribute-Driven Design)

Según Bass et al. (2022, Capítulo 20), el Diseño impulsado por atributos (ADD – Attribute Driven Design) es un método para definir la arquitectura de software, basado en los requerimientos de atributos de calidad del software. Es un proceso recursivo que descompone un sistema o elemento del sistema mediante la aplicación de tácticas y patrones arquitectónicos que satisfacen estos requerimientos. ADD implica varios pasos esenciales para el diseño arquitectónico de software:

1. **Revisión de entradas (Review Inputs):** Antes de comenzar una ronda de diseño, se deben verificar los impulsores arquitectónicos que son las entradas al proceso de diseño. Esto incluye propósito de la ronda de diseño, requerimientos funcionales primarios, escenarios principales de atributos de calidad (QA), cualquier restricción y preocupaciones. Es fundamental tener claridad sobre los objetivos para cada ronda de diseño.
2. **Establecer el objetivo de la iteración seleccionando impulsores (Establish Iteration Goal by Selecting Drivers):** Cada iteración de diseño se centra en lograr un objetivo particular, que típicamente involucra satisfacer un

subconjunto de los impulsores. Por ejemplo, el objetivo de una iteración puede ser crear estructuras a partir de elementos que permitirán lograr un escenario de rendimiento específico o un caso de uso.

3. **Elegir uno o más elementos del sistema para refinar (Choose One or More Elements of the System to Refine):** Satisfacer los impulsores requiere tomar decisiones de diseño arquitectónico, las cuales se manifiestan en una o más estructuras arquitectónicas compuestas de elementos interrelacionados, obtenidos refinando otros elementos identificados en una iteración anterior.

4. **Elegir uno o más conceptos de diseño que satisfagan los impulsores seleccionados (Choose One or More Design Concepts That Satisfy the Selected Drivers):** Seleccionar los conceptos de diseño es probablemente la decisión más difícil en el proceso de diseño, ya que requiere identificar los diferentes conceptos de diseño que podrían utilizarse plausiblemente para lograr el objetivo de la iteración.

5. **Instanciar elementos arquitectónicos, asignar responsabilidades y definir interfaces (Instantiate Architectural Elements, Allocate Responsibilities, and Define Interfaces):** Una vez seleccionados uno o más conceptos de diseño, se deben instanciar elementos a partir de los conceptos de diseño seleccionados, asignar responsabilidades a cada uno y definir interfaces.

6. **Bosquejar vistas y registrar decisiones de diseño (Sketch Views and Record Design Decisions):** Después de realizar las actividades de diseño para la iteración, se deben documentar las vistas preliminares de las estructuras creadas y registrar las decisiones de diseño significativas y sus razones.

7. **Realizar análisis del diseño actual y revisar el objetivo de la iteración y el logro del propósito de diseño (Perform Analysis of Current Design and Review Iteration Goal and Achievement of Design Purpose):** Se debe crear un diseño parcial que aborde el objetivo establecido para la iteración y asegurarse de que esto sea así, lo cual puede incluir la revisión de las vistas y las decisiones de diseño capturadas.

El proceso descrito se despliega a través de fases iterativas y recursivas, garantizando así que la arquitectura del software se adhiera a los atributos de calidad exigidos y a los requerimientos sistémicos. Con cada paso fundamentado en el aprendizaje del anterior, se avanza gradualmente hacia un diseño arquitectónico sólido, en sintonía con las exigencias de los interesados y las metas del sistema. En este contexto, dicho enfoque facilita el desarrollo de una Super App robusta, diseñada específicamente para abordar y resolver la problemática identificada.

4.1.17. AXIOS

Axios es un cliente HTTP basado en promesas para el navegador y Node.js. Ofrece una API isomórfica, permitiendo usar el mismo código en ambos entornos. En el navegador, utiliza XMLHttpRequests y en Node.js, el módulo HTTP nativo. Sus características incluyen interceptación de solicitudes, cancelaciones, manejo automático de JSON, y protección contra XSRF. Axios facilita la serialización de parámetros de consulta y cuerpos de solicitud, y captura el progreso de las transferencias de datos (Axios Team, 2024).

4.1.18. API FETCH de Javascript

La API Fetch es una interfaz nativa de JavaScript para realizar peticiones HTTP de manera sencilla y eficiente. A diferencia de XMLHttpRequest, Fetch utiliza promesas, facilitando el manejo asincrónico y simplificando el código. Permite hacer solicitudes a recursos, manejar respuestas, y configurar opciones de solicitud de forma flexible. Además, está integrada en casi todos los contextos de JavaScript moderno, promoviendo su uso en aplicaciones web para mejorar la interacción con servidores y APIs (Contributors, 2023).

4.1.19. Zustand

Zustand es una solución de gestión de estado ligera y eficaz para React, basada en hooks, que permite manejar el estado sin necesidad de proveedores. Este enfoque isomórfico permite integraciones simples y un rendimiento óptimo al evitar problemas

comunes como la pérdida de contexto. Los desarrolladores pueden definir y actualizar estados de manera flexible y reutilizable (Zustand Team, 2024).

4.1.20. I18next

I18next es un marco de internacionalización para JavaScript, proporcionando herramientas completas para adaptar aplicaciones a múltiples idiomas. Es nativo de JavaScript y facilita la carga de traducciones, detección del idioma del usuario y configuración flexible. i18next es escalable para proyectos de cualquier tamaño y cuenta con numerosos plugins para integrarse con otros sistemas y mejorar la gestión de traducciones (i18next Team, 2024).

4.1.21. React native paper

React Native Paper es una biblioteca que cumple con los estándares de calidad para desarrollo en React Native, basada en Material Design la cual proporciona una amplia gama de componentes UI que ayudan a que las aplicaciones móviles parezcan y se sientan nativas en iOS y Android. Ofrece soporte completo para temas, adaptaciones a la plataforma, accesibilidad y soporte de idiomas de derecha a izquierda (RTL), facilitando la creación de interfaces de usuario coherentes y accesibles (React Native Paper Team, 2024).

4.1.22. React Native WebView

React Native WebView es un componente que encapsula WebView para aplicaciones React Native, permitiendo mostrar contenido web dentro de apps nativas. Es útil para integrar páginas web completas o widgets de terceros en aplicaciones móviles. El proyecto está alojado en GitHub, donde se puede acceder a su documentación completa, ejemplos de uso y código fuente para su integración y personalización en proyectos de desarrollo móvil (React Native WebView Contributors, 2024).

4.1.23. Bootstrap

Bootstrap es una biblioteca HTML, CSS y JS muy popular para desarrollar sitios responsivos y móviles. Ofrece un sistema de grillas para el diseño, componentes reutilizables y potentes plugins de JavaScript, facilitando el diseño de interfaces modernas y estéticas. Su documentación completa y actualizada permite personalizar y extender el framework según las necesidades específicas del proyecto (Bootstrap Team, 2024a).

4.1.24. Window

La interfaz Window de la API Web proporciona funciones y propiedades relacionadas con la ventana del navegador. Permite la interacción con el entorno de navegación, controlando aspectos como el tamaño de la ventana, la manipulación de eventos y el almacenamiento local. Es un objeto global que representa la ventana que contiene un documento DOM (Contributors, 2024a).

4.1.25. API (Application Programming Interface)

- Una API es una forma de compartir información, especificando qué información puede solicitarse, cómo se realizan las solicitudes y qué se devuelve (datos o un mensaje de error). La API envía una consulta o Llamada(s) y recibe una respuesta o Devolución(es) con datos (IEEE, 2024).

- Una API, o interfaz de programación de aplicaciones, es un conjunto de reglas definidas que permiten a las diferentes aplicaciones comunicarse entre sí. Actúa como una capa intermedia que procesa las transferencias de datos entre sistemas, permitiendo a las empresas abrir los datos y la funcionalidad de sus aplicaciones a desarrolladores externos, socios comerciales y departamentos internos dentro de sus empresas (IBM, 2024b).

4.1.26. React Custom Hooks

Los hooks personalizados permiten reutilizar lógica de estado y efectos en componentes de React (React native desde la version 0.59) sin sacrificar la claridad del código. Los hooks personalizados encapsulan funcionalidades comunes y las hacen reutilizables a través de diferentes componentes, facilitando la gestión del estado y los ciclos de vida de componentes de manera más modular y mantenible. Esta metodología no solo mejora la legibilidad del código, sino que también promueve la eficiencia en el desarrollo de aplicaciones complejas (React Documentation Contributors, 2024).

4.1.27. Pruebas de unidad o unitarias

Una prueba unitaria es un bloque de código que verifica la precisión de un bloque más pequeño y aislado de código de aplicación, normalmente una función o un método. La prueba unitaria está diseñada para verificar que el bloque de código se ejecuta según lo esperado, de acuerdo con la lógica teórica del desarrollador. La prueba unitaria solo interactúa con el bloque de código a través de entradas y salidas (verdaderas o falsas) capturadas afirmadas.

Un solo bloque de código también puede tener un conjunto de pruebas unitarias, conocidas como casos de prueba. Un conjunto completo de casos de prueba cubre todo el comportamiento esperado del bloque de código, pero no siempre es necesario definir el conjunto completo de casos de prueba.

Cuando un bloque de código requiere que otras partes del sistema se ejecuten, no se puede utilizar una prueba unitaria con esos datos externos. La prueba unitaria debe ejecutarse de forma aislada. Es posible que el código requiera otros datos del sistema, como bases de datos, objetos o comunicaciones de red, para funcionar de forma correcta. Si ese es el caso, utilice fragmentos de datos en su lugar. Es más fácil escribir pruebas unitarias para bloques de código pequeños y lógicamente sencillos (Amazon Web Services, 2024).

4.1.28. Pruebas de integración

Las pruebas de integración, según Katalon (2024), son un proceso sistemático destinado a evaluar la eficacia de la interacción entre módulos o componentes de software tras completar las pruebas unitarias. Este enfoque es crucial para identificar problemas en las interfaces y asegurar que los componentes funcionen adecuadamente en conjunto dentro de un sistema más amplio. De igual manera, en la publicación de Katalon (2024) se discuten varios métodos como el Big Bang, donde todos los módulos se integran y prueban simultáneamente, y enfoques incrementales como Bottom-up, que empieza con módulos de nivel más bajo y avanza hacia arriba; Top-down, que procede en dirección contraria; y Sandwich, que combina ambos enfoques para una cobertura más completa de las pruebas.

4.1.29. Jest

Jest es un marco de pruebas de JavaScript enfocado en la simplicidad y la eficiencia. Permite trabajar con proyectos que usan Babel, TypeScript, Node, React, Angular, Vue, entre otros. Características destacadas incluyen configuración mínima, ejecución de pruebas en paralelo para maximizar el rendimiento, manejo fácil de mock objects, y generación de informes de cobertura de código. Además, facilita el seguimiento de objetos grandes a través de snapshots y ofrece una API completa y documentada para pruebas (Jest Core Team, 2024).

4.2. Trabajos relacionados

4.2.1. WeChat

Conocido en chino como Weixin, es una aplicación de chat gratuita para dispositivos móviles que ofrece video llamadas y llamadas de voz gratuitas, videos, imágenes, juegos, stickers y mensajes de texto para ayudarte a mantenerte conectado con las personas que más te importan. WeChat fue lanzado en 2011 por Tencent, un importante proveedor de servicios de internet en China. WeChat se ha desarrollado en una aplicación integral que las personas en China pueden usar para llamar a un taxi,

pedir boletos de tren y avión, ordenar comidas para llevar, hacer compras en línea, comprar boletos de cine, reservar un hotel, alquilar bicicletas públicas, pagar facturas del hogar e incluso comerciar bienes de segunda mano.

WeChat ha creado un nuevo estilo de vida que se ajusta a la modernidad en una era digital y mejora la experiencia de vida en línea de cientos de millones de usuarios con sus características innovadoras. La plataforma integra mensajería instantánea y entretenimiento social, permitiendo a los usuarios participar en comunicaciones en tiempo real a través de mensajes de texto y multimedia gratuitos, hacer videollamadas o compartir fotos o videos en sus "Momentos".

Aparte de las similares características de mensajería y llamadas gratuitas que WhatsApp, Facebook y otras redes sociales tienen, otras características recreativas de estilo de vida incluyen "Juegos", "Galería de stickers", "Compras", "Mini Programa", y servicios convenientes para agregar amigos como "Escanear", "Agitar" y "Personas cercanas" que hacen que WeChat sea único y entretenido.

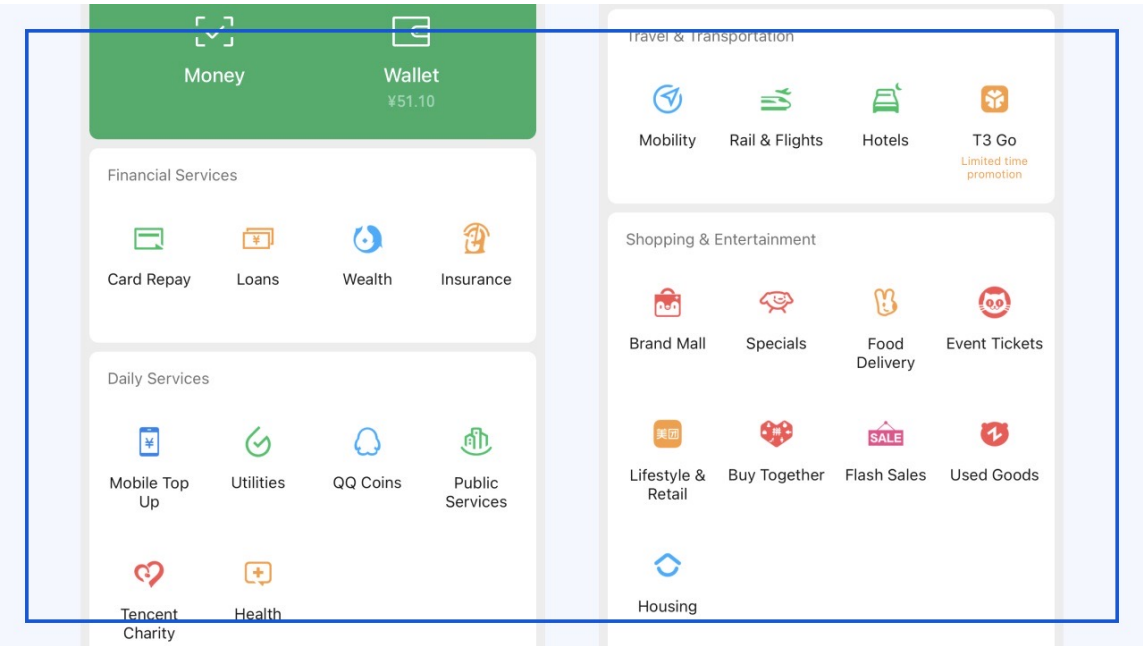


Figura 1: Servicios de WeChat QPSoftware (2022)

WeChat también ofrece a las empresas "Cuentas Oficiales" para crear experiencias

de consumo originales a través de su plataforma abierta y servicios extendidos. Estas empresas utilizan WeChat Pay, una característica y servicio que contribuye en gran medida a las economías de consumo sin efectivo en China, lo que es un estilo de vida digital verdaderamente móvil.

Lo que hace a WeChat más interesante es que sus características y funciones evolucionan continuamente siguiendo las necesidades de los usuarios. Se ha convertido en un conector y plataforma abierta en diversas industrias, conectando a los usuarios entre sí, a dispositivos inteligentes y servicios empresariales. Un video en Youtube llamado "WeChat Top Features Guide" podría dar una idea general de sus capacidades sobresalientes en la vida diaria.

La flexibilidad técnica y accesibilidad de WeChat a varias versiones de sistemas constituyen sus principales características únicas. WeChat es accesible en dispositivos móviles a través de redes de datos y Wi-Fi en todo el mundo, esto significa que, independientemente de la ubicación en el mundo, es posible enviar mensajes o realizar videollamadas siempre que se disponga de un dispositivo móvil con acceso a internet.

Para diferentes dispositivos y sistemas operativos, WeChat ha diseñado versiones personalizadas para cada uno: WeChat Windows, WeChat Mac, WeChat Web, WeChat para Android y iOS, teléfonos Blackberry y Nokia. Los usuarios de más de 100 países pueden registrar cuentas de WeChat con sus números de teléfono. WeChat también admite 21 diferentes idiomas en su interfaz (Jiang, 2018).

4.2.2. Rappi

Es considerada una Super App debido a la amplia gama de servicios que ofrece a sus usuarios. Inicialmente comenzó como una aplicación de entrega de alimentos y bebidas en Colombia en 2015, pero desde entonces ha expandido su cartera para cubrir una variedad de servicios en nueve países de América Latina. Además de las entregas de comida y otros productos, Rappi ofrece servicios como manicuras, actividades domésticas y masajes. En 2020, la compañía lanzó dos nuevos verticales: Rappi Entertainment, que cuenta con funciones dedicadas al streaming de música, eventos en vivo y juegos, y Rappi Travel, para reservar boletos de avión, autobuses y

hoteles, con la intención de agregar alquiler de autos, tours y seguro de viaje (Lima, 2023).

En cuanto a sus servicios financieros, Rappi ha estado trabajando para consolidar su estatus como una Super App en América Latina expandiendo sus servicios financieros en la región. Esto ha incluido asociaciones con bancos locales y la creación de productos financieros, como RappiPay, que permite a los usuarios enviar dinero sin costo, pagar facturas, comprar en la plataforma, en tiendas físicas y en línea. Rappi también lanzó RappiCash, un servicio de entrega a domicilio de efectivo, y ofrece tarjetas de débito y crédito con beneficios atractivos como la falta de una tarifa anual o membresía, cashback en compras realizadas y un programa de referencias exclusivo. Además, Rappi ha estado trabajando para establecer bancos propios en asociación con bancos locales. Por ejemplo, en marzo de 2023, Rappi y Davivienda anunciaron que solicitarían autorización para establecer y operar un nuevo banco, que ofrecería productos de ahorro y crédito para individuos y empresas, así como para rappitenderos (Love, 2023).

Por último, Rappi también ha lanzado su banco digital, Rappi Bank, en varios mercados, ofreciendo una línea de crédito de capital de trabajo destinada a los socios de la Super App (Lima, 2023).

4.2.3. Line

Según Sehl (2021), Line es considerada una "Super App" debido a la amplia gama de funciones y herramientas que ofrece. Originalmente lanzada como una aplicación de mensajería móvil, Line ha evolucionado hasta convertirse en un centro de entretenimiento, actividades sociales y cotidianas, similar a las super aplicaciones como WeChat de China y KakaoTalk de Corea del Sur. Las funcionalidades de Line van más allá de la mensajería móvil, las llamadas de voz y las video llamadas. La aplicación integra herramientas para banca, compras y más. Además, centraliza el acceso a servicios de salud, agentes de viajes, restaurantes y tiendas, y tableros de empleo. Line también funciona como un agregador de noticias y un servicio de transmisión de medios, proporcionando a los usuarios acceso a artículos diarios, más de 70 millones

de canciones, manga, videos en vivo, juegos y mucho más. Aquí hay una lista no exhaustiva de lo que se puede hacer con Line:

- Llamadas de voz y video.
- Mensajeria de texto.
- Compartir y crear stickers.
- Foros e hilos OpenChat.
- Realizar pagos o enviar dinero con Line Pay.
- Pedir comida a domicilio con Line Man.
- Recibir predicciones de adivinos de Line.
- Descubrir y leer manga.
- Consultar con profesionales de la salud.
- Enviar regalos a amigos con Line Gift.
- Invertir en acciones o criptomonedas.
- Transmitir música, conciertos en vivo, deportes y más.
- Ver y hablar con streamers en Line Live.

Line es una filial consolidada del gigante de Internet de Corea del Sur, Naver, y SoftBank Corp., cada compañía tiene una participación del 50 % en Line. En marzo de 2021, Line se fusionó con el afiliado de SoftBank, Yahoo! Japón, que fue renombrado Z Holdings. Por su parte, Line fue renombrada A Holdings Corporation, aunque sigue siendo marca y reconocida como Line1.

En cuanto a su demografía, con 86 millones de usuarios en todo el país, Line es la aplicación social más grande de Japón, pero también es popular en otros lugares. Line es la red social más popular en Tailandia, donde se publican 21 millones de

historias todos los días, recibiendo 170 millones de visitas diarias. Según el Informe Anual 2020 de Naver corp (2020), Line cuenta con 165 millones de usuarios activos mensuales (MAU) en Japón y el sudeste asiático. A nivel mundial, esa cifra se eleva a 182 millones de MAU.

4.2.4. InstaCare: Super Health App

En el sitio web de InstaCare (2024), se definen como una plataforma de salud ubicada en Pakistán que facilita el acceso a servicios médicos, permitiendo consultas en línea con doctores, reservaciones de pruebas de laboratorio, y pedidos de medicamentos. Se enfoca en hacer la atención médica accesible y asequible mediante soluciones digitales, mejorando la calidad del cuidado de la salud. Además, según Google, Play Store (2024) esta ha sido galardonada como la aplicación de salud digital más innovadora de Pakistán, por medio de esta se puede:

- Encontrar los mejores médicos.
- Reservar citas (en persona o en línea).
- Reservar pruebas de laboratorio.
- Recibir medicamentos directamente en su puerta, todo desde la comodidad de su hogar.

4.2.5. Medsi

Según Healthcare IT Today Contributors (2023), una empresa emergente en el sector de la salud en México, se enfoca en abordar la gran brecha en el acceso a atención médica asequible en América Latina, comenzando específicamente en el mercado mexicano. Esta "fintech meets healthtec" ha lanzado su primera oferta de financiación de la salud, que incluye una línea de crédito rotativa disponible para toda la familia que puede ser utilizada en servicios y procedimientos de salud esenciales o electivos. Al ser aprobados mediante un proceso de solicitud intuitivo de tres minutos, los usuarios pueden programar los desembolsos y Medsi paga directamente al médico

o clínica privada utilizando un código QR a través de la aplicación Medsi Credit, que está disponible en Google Play y App Store. Además, recientemente, Medsi ha recaudado 10 millones de dólares en una ronda de financiamiento de deuda de CAPEM México, destinados a impulsar la incorporación de decenas de miles de clientes a largo plazo en su mercado.

En enero, Medsi expandió su base de clientes potenciales lanzando una nueva solución de "ahorro a crédito" que permite a los usuarios realizar un pago inicial para un procedimiento médico y luego recibir una línea de crédito de Medsi por un valor diez veces mayor al del pago inicial después de haber realizado cuatro pagos quincenales con éxito. Esta oferta está dirigida especialmente a más del 55 por ciento de los mexicanos que pertenecen al sector informal o son nuevos en el crédito, avanzando hacia una verdadera inclusión financiera incremental. Además, Medsi busca convertirse en la primera Super App que actúe como un mercado de salud y un centro comunitario para conectar y mejorar la transparencia entre pacientes y médicos, automatizando procesos y eliminando ineficiencias combinando tecnología con un toque humano. La misma plataforma Medsi (2024), se definen como una empresa que combina tecnología financiera y de salud para mejorar el acceso a la atención médica en México. La compañía ofrece líneas de crédito revolvente para servicios médicos, accesibles a través de su aplicación móvil. El crédito revolvente es una línea de crédito que permanece disponible incluso a medida que el saldo se paga. Los prestatarios pueden acceder al crédito hasta una cantidad específica y tener acceso continuo a ese crédito. Pueden optar por pagar el saldo en su totalidad o realizar pagos regulares. Cada pago, menos los intereses y las tarifas aplicables, restablece el crédito disponible para el titular de la cuenta. Ejemplos de crédito revolvente incluyen tarjetas de crédito, líneas de crédito y líneas de crédito con garantía hipotecaria. Estos funcionan de manera diferente a los préstamos a plazos (Investopedia, 2024). Esto permite a los usuarios, especialmente aquellos en el sector informal o sin historial crediticio, financiar su atención médica de manera eficiente y asequible. Permitiendo hacer la mayoría de su funcionamiento por medio de la Super app que tienen.

5. Requerimientos Super App

En este capítulo, se realizó una evaluación de los requerimientos mínimos esenciales para el desarrollo de una Super App, tomando como referencia el marco teórico previamente investigado. Se identifica y analiza un primer requerimiento de una App / Micro App para integrar, seleccionado según las prioridades de la institución de salud de alta complejidad. Además, se elaboraron documentos de especificación detallados que sirvieron como herramientas cruciales en los capítulos siguientes.

5.1. Requerimientos mínimos de Super App

Para este punto se aclara que no se encontró una documentación específica académica que listara o describiera de forma explícita cuáles son los requerimientos mínimos de un desarrollo de software para ser considerado Super App. Sin embargo, durante la investigación del marco teórico sobre el tema se encontraron diferentes características que permitieron extraer los siguientes requerimientos:

- Autenticación única y centralizada.
- Soporte múltiples idiomas.
- Compatibilidad con sistema operativo móvil iOS y Android.
- Integración de Apps / Micro Apps.

Estos se detallan a continuación.

5.1.1. Autenticación única y centralizada

Según como lo indica Baquero (2021, Pag. 17), en donde se está refiriendo a las ventajas de las Super Apps, se afirma que la Super App debe proporcionar un inicio de sesión único, lo que significa que los usuarios no necesitan tener un nombre de usuario y contraseña para su servicio de entrega de comida y uno diferente para reservar un taxi. Hay un inicio de sesión único para acceder a todas las funcionalidades y miniprogramas (Apps / Micro Apps), lo cual va a permitir a la Super App

ofrecer diversos servicios con una experiencia de usuario uniforme e individual; Esto es posible gracias a la estandarización y personalización de la interfaz de usuario.

Con base en lo anterior, se identifica un criterio esencial para el desarrollo de la Super App: la implementación de un sistema de autenticación única. Este sistema no solo simplificará el acceso de los usuarios, sino que también garantizará una transición fluida entre las diversas aplicaciones y microaplicaciones integradas (Apps / Micro Apps). La autenticación única será el fundamento que permitirá a los pacientes disfrutar de una variedad de servicios sin la necesidad de iniciar sesión múltiples veces. Además, es imperativo que la Super App tenga la capacidad de extender esta característica de autenticación a todas las aplicaciones y servicios asociados, asegurando así una experiencia de usuario coherente y eficiente.

5.1.2. Soporte múltiples idiomas

Dentro del contexto del marco teórico y, específicamente, los trabajos relacionados previamente existentes (Estado del Arte), es notable que todas estas Super Apps ofrecen la opción de personalizar el idioma para brindar una experiencia de usuario a medida. En la institución de salud de alta complejidad, se atiende a pacientes que hablan español e inglés. Por lo tanto, es un requisito por parte de la institución que la aplicación soporte como mínimo estos dos idiomas.

Además, es esencial que la Super App esté diseñada para integrar nuevos idiomas de manera fluida en el futuro. Esto significa que no es necesario realizar reprocesos ni desarrollos críticos para añadir idiomas adicionales. La flexibilidad y la escalabilidad en el soporte de idiomas son cruciales para adaptarse a las necesidades cambiantes de los pacientes y garantizar una atención inclusiva y personalizada.

5.1.3. Compatibilidad con sistema operativo iOS y Android

Al desarrollar la Super App para pacientes de la institución de salud de alta complejidad se debe garantizar su accesibilidad a la mayoría de los usuarios. Según los datos proporcionados por StatCounter (2023b), una plataforma que analiza más de 5 mil millones de páginas vistas mensualmente en una red de más de 1.5 millo-

nes de sitios web, se ha identificado una predominancia significativa de los sistemas operativos Android e iOS en el mercado global y colombiano.

A nivel mundial (Septiembre 2022 – Septiembre 2023), la distribución del uso de sistemas operativos móviles se puede observar en la Figura 2 y Tabla 1.

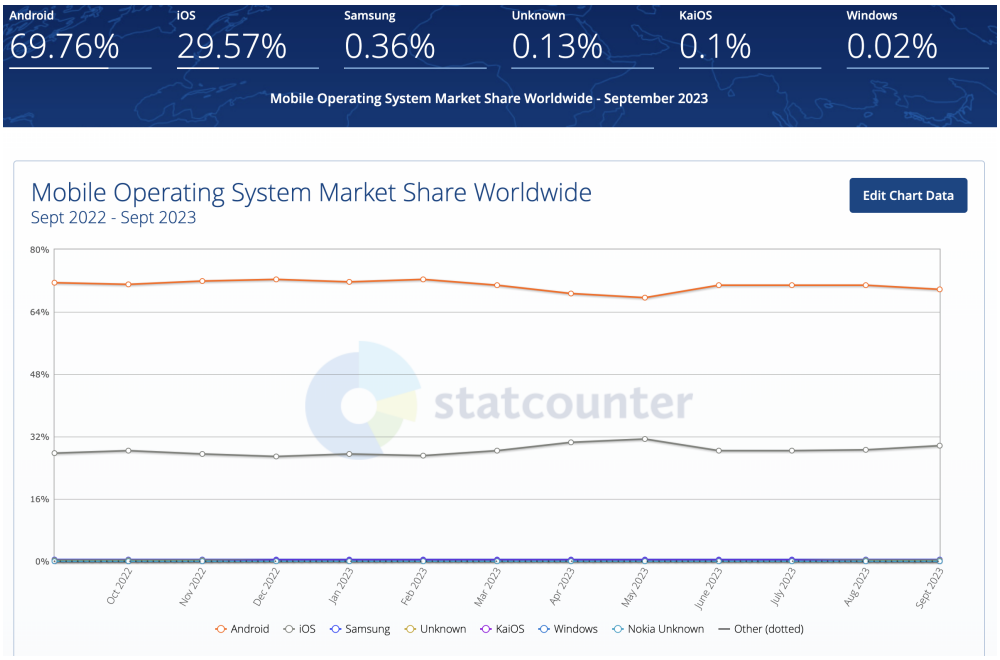


Figura 2: Distribución del uso de sistemas operativos móviles en el mundo StatCounter (2023b)

Sistema Operativo	**Porcentaje de Uso**
Android	69.76 %
iOS	29.57 %
Samsung	0.36 %
Desconocido	0.13 %
KaiOS	0.1 %
Windows	0.02 %

Tabla 1: Distribución del uso de sistemas operativos móviles en el mundo (Septiembre 2022 – Septiembre 2023)

En Colombia, durante el mismo período, la distribución se puede observar en la Figura 3 y Tabla 2

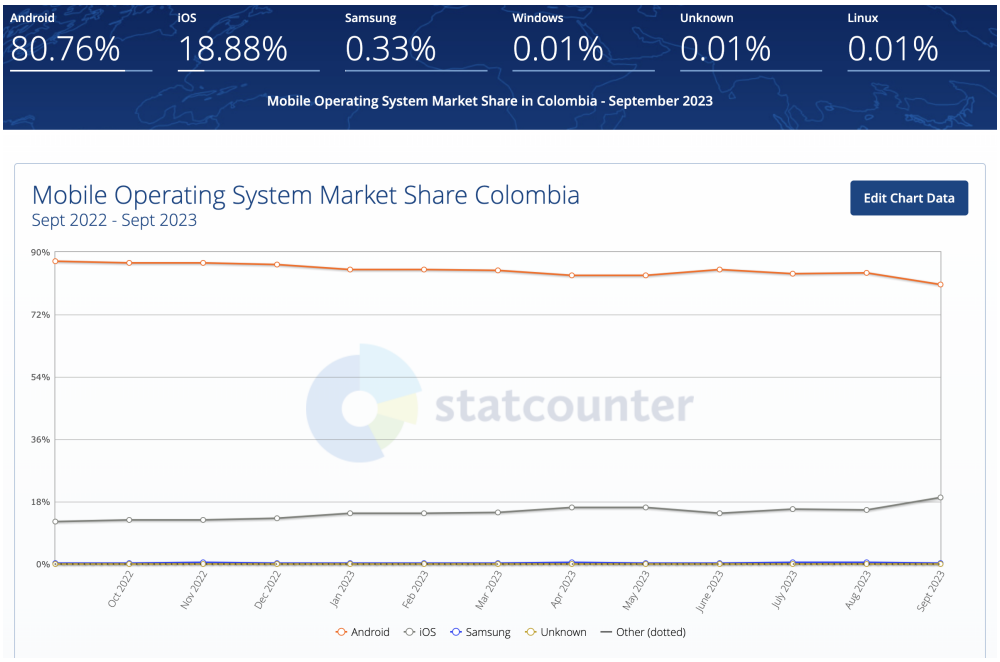


Figura 3: Distribución del uso de sistemas operativos móviles en Colombia StatCounter (2023a)

Sistema Operativo	**Porcentaje de Uso**
Android	80.76 %
iOS	18.88 %
Samsung	0.33 %
Windows	0.01 %
Desconocido	0.01 %
Linux	0.01 %

Tabla 2: Distribución del uso de sistemas operativos móviles en Colombia (Septiembre 2022 – Septiembre 2023)

Estas estadísticas, que se actualizan diariamente y están sujetas a un período de

revisión de calidad de 45 días, resaltan la importancia crítica de desarrollar la Super App compatible con iOS y Android. Al adaptarse a estos dos sistemas operativos predominantes, la aplicación será accesible para el 99.64 % de los usuarios de dispositivos móviles en Colombia donde se encuentran la mayoría de pacientes atendidos por la institución de salud de alta complejidad (Android: 80.76 %, iOS: 18.88 %). Esta estrategia de compatibilidad multiplataforma no solo asegura una penetración de mercado óptima, sino que también facilita el acceso a los servicios de salud ofrecidos por la Super App.

5.1.4. Integración de Apps / Micro Apps

En el contexto actual de la digitalización acelerada, las Super Apps emergen como un fenómeno crucial en los mercados en desarrollo, consolidando múltiples servicios y funcionalidades en una sola plataforma Baquero (2021, Pag. 3). Estas aplicaciones multifuncionales no solo están transformando la experiencia del usuario, sino que también están redefiniendo los modelos de negocio y las estrategias de mercado en la economía digital.

La integración de Apps / Micro Apps dentro de las Super Apps es un componente esencial para su éxito y expansión. Según un estudio detallado sobre el auge de las Super Apps en países emergentes, estas aplicaciones se caracterizan por ofrecer una variedad de servicios que abarcan múltiples industrias, todo dentro de una única interfaz Ye (2022, Pag. 2). Esta característica es posible gracias a la capacidad de integrar y gestionar eficientemente diversas aplicaciones y servicios, lo que permite a los usuarios acceder a una gama amplia de funcionalidades sin la necesidad de descargar aplicaciones individuales.

De nuevo Baquero (2021, Pag. 3), enfatiza la importancia de la integración de múltiples funcionalidades dentro de una sola aplicación. La autora destaca cómo las Super Apps, como WeChat y Rappi, han logrado consolidar mercados al ofrecer un ecosistema de servicios integrados, desde pagos móviles hasta servicios de entrega y más allá. Esta integración no solo mejora la experiencia del usuario, sino que también crea oportunidades para la innovación y el crecimiento empresarial.

La capacidad de integrar Micro Apps se evidencia en la proliferación de mini-programas dentro de plataformas como WeChat Baquero (2021, Pag. 56). Estos mini-programas son aplicaciones subyacentes que operan dentro de la Super App principal, ofreciendo funcionalidades específicas sin requerir una instalación separada. Este nivel de integración optimiza la experiencia del usuario, reduce los costos de adquisición de clientes y fomenta un ecosistema digital más cohesionado y eficiente.

Con todo lo anterior, es claro que para maximizar su impacto y eficiencia, la Super App destinada a pacientes de una institución de salud de alta complejidad debe priorizar de manera obligatoria la integración de Apps / Micro Apps. Esta característica no solo enriquece la experiencia del usuario, facilitando un acceso más rápido y eficiente a los servicios de salud que se tengan, sino que también permite a la institución explorar nuevos modelos de negocio, acceder a datos valiosos y adaptarse rápidamente a las cambiantes demandas del mercado y las necesidades específicas de los pacientes y/o servicios.

5.1.5. Generación de documentos de especificación de requerimientos

En la actualidad la institución de salud de alta complejidad para sus proyectos manejan metodologías tradicionales, para ser más específicos se basan en PMBOK (Project Management Body of Knowledge), el cual lo define Jamali and Oveisi (2016) en su artículo como un conjunto de procesos y campos de conocimiento que son ampliamente aceptados como las mejores prácticas dentro de la disciplina de gestión de proyectos. El PMBOK también es un estándar reconocido internacionalmente que proporciona los fundamentos de la gestión de proyectos aplicables a una amplia gama de proyectos. Esta guía es desarrollada y mantenida por el Project Management Institute (PMI) y es esencial para profesionales que buscan estructurar y estandarizar los procesos de gestión de sus proyectos. El PMBOK está estructurado alrededor de cinco grupos de procesos y diez áreas de conocimiento. Los cinco grupos de procesos son: Iniciación, Planificación, Ejecución, Monitoreo y Control, y Cierre. Estos grupos de procesos representan la secuencia en la que se deben llevar a cabo los proyectos desde su comienzo hasta su conclusión. Las diez áreas de conocimiento incluyen:

gestión de la integración del proyecto, gestión del alcance del proyecto, gestión del cronograma del proyecto, gestión de los costos del proyecto, gestión de la calidad del proyecto, gestión de los recursos del proyecto, gestión de las comunicaciones del proyecto, gestión de los riesgos del proyecto, gestión de las adquisiciones del proyecto y gestión de los interesados del proyecto. Cada área aborda un aspecto específico de la gestión de proyectos y juntas proporcionan un marco comprensivo para la administración eficaz de proyectos de cualquier tamaño y complejidad.

Sin embargo, se ha observado que el uso de la metodología tradicional en la institución tiende a extender la duración de los proyectos, acumulando trámites excesivos, numerosas reuniones y extensa documentación. Por lo tanto, para el desarrollo de la Super App, se propone un piloto inicial en el cual, para el documento de especificación de requerimientos funcionales, se incorporen elementos de metodologías ágiles, creando así un enfoque híbrido. Las metodologías ágiles han probado ser más efectivas que las tradicionales en varios aspectos cruciales, particularmente en proyectos de software y tecnologías de la información. A continuación, se expondrán algunos hallazgos de investigaciones recientes que subrayan las ventajas de las metodologías ágiles:

1. **Eficiencia y Satisfacción de los Interesados:** Un estudio en entornos de desarrollo de software financiero encontró que las metodologías ágiles aumentan significativamente la eficiencia y la transparencia del proceso de desarrollo, además de mejorar la satisfacción de los interesados. Esto se debe a la capacidad de las metodologías ágiles para adaptarse mejor al contexto específico de la organización, permitiendo ajustes rápidos y una mejor comunicación con los interesados, Munteanu and Dragoș (2021)

2. Según el artículo de Etim et al. (2021), habla sobre el manejo de proyectos de TI utilizando metodologías ágiles, destacando un caso específico en África. Resalta cómo el enfoque ágil facilita una entrega de producto más rápida con retroalimentación constante y cómo se distingue de prácticas tradicionales como el método cascada.

3. Se encontró otro artículo que tiene como título Hybrid Project Management

between Traditional Software Development Lifecycle and Agile Based Product Development for Future Sustainability, escrito por Leong et al. (2023), desde mi punto de vista muy interesante el cual revisa la sostenibilidad futura de la gestión de proyectos combinando metodologías ágiles y gestión de productos para mantenerse al día con las tendencias actuales en desarrollo de software, Se considera que, dado el rápido avance de la transformación digital, surge la interrogante de si los métodos tradicionales de gestión de proyectos pueden sostenerse a largo plazo.

Estos estudios muestran cómo las metodologías ágiles no solo mejoran la gestión de proyectos de software mediante una mayor adaptabilidad y respuesta a las necesidades del proyecto, sino también cómo pueden ser implementadas efectivamente en diversos contextos y combinadas con enfoques tradicionales para una gestión óptima.

Con el contexto y las bases establecidas, se ha desarrollado una nueva plantilla para el documento de especificación de requerimientos funcionales. Esta plantilla incorpora elementos fundamentados en el Software Engineering Body of Knowledge (SWEBOK), que según Fairley et al. (2014), esta es una guía que proporciona un conocimiento generalmente aceptado para la profesión de ingeniería de software, basado en fuentes académicas y las mejores prácticas de la profesión. Este documento es utilizado como referencia para curriculums académicos e industriales, programas de certificación, criterios de acreditación y licencias profesionales. La guía está diseñada para servir como un recurso documental esencial que ofrece un acceso tópico al cuerpo de conocimiento que soporta la disciplina de la ingeniería de software en general, pero se considera que los elementos más importantes y a resaltar son:

1. **Historias de usuario:** Atlassian (2024) define las historias de usuario como la unidad de trabajo más pequeña en un marco ágil. Es un objetivo final, no una función, expresado desde la perspectiva del usuario del software. Una historia de usuario es una explicación general e informal de una función de software escrita desde la perspectiva del usuario final o cliente. El propósito de una historia de usuario es articular cómo un elemento de trabajo entregará un valor particular al cliente. Ten en cuenta que los "clientes" no tienen por qué ser usuarios

finales externos en el sentido tradicional, también pueden ser clientes internos o colegas dentro de tu organización que dependen de tu equipo. Las historias de usuario son unas pocas frases en lenguaje sencillo que describen el resultado deseado. No entran en detalles, ya que los requisitos se añaden más tarde, una vez acordados por el equipo, para el formato y las columnas establecidas se tuvo en cuenta también un artículo de Felip (2020), el cual describe de manera bastante practica como escribir buenas historias de usuario.

2. **Prototipo de baja fidelidad:** En la institución en cuanto a desarrollos de software se encontró una problemática, pues en muchas ocasiones la interfaz del software desarrollado en la mayoría de los casos no era como el usuario se lo había imaginado, por lo cual por medio de esta nueva iniciativa se propone usar prototipos de baja fidelidad los cuales según Cox et al. (2022), define el prototipado de baja fidelidad como el uso de representaciones simplificadas de un producto para avanzar en la comprensión y el diseño del mismo. Este enfoque permite a los diseñadores y desarrolladores experimentar y ajustar conceptos rápidamente sin invertir tiempo y recursos en prototipos detallados desde las primeras etapas. El artículo también propone una taxonomía de las dimensiones de la fidelidad del producto, lo que ayuda a gestionar mejor estos prototipos para maximizar su valor en el proceso de desarrollo.

Como resultado de lo anterior, se ha generado el documento inicial de especificación de requerimientos para la Super App, designado como Anexo A.1. Este documento ha sido revisado y aprobado por el director del proyecto de grado, quien posee expertise en la materia. Además, servirá como el estándar para la especificación de todos los componentes de software relacionados con el desarrollo del proyecto actual.

5.2. Seleccionando App / Micro App

Considerando que la institución de salud de alta complejidad recibe múltiples solicitudes de los servicios hacia el área de informática, donde se piden diferentes desarrollos de aplicaciones móviles, para este desarrollo de Super App, se solicitó al

Subdirector de Informática y al Coordinador de Desarrollo de Software que listaran y seleccionaran los requerimientos según la prioridad.

En respuesta, se realizó una reunión presencial en las instalaciones de la institución, donde se listaron los siguientes requerimientos ya depurados para ser evaluados:

1. **Autorizaciones:** Este aplicativo ha sido solicitado por el área de mercadeo. Consiste en el desarrollo de una APP que a través de un smartphone, permita a los pacientes solicitar la tramitación de sus órdenes para procedimientos, exámenes y consultas con especialistas. El objetivo principal es agilizar la gestión y asignación de citas, haciendo el proceso más eficiente y amigable para el paciente.

2. **Resultados / Informes:** El servicio de laboratorios ha solicitado este aplicativo móvil. Su función principal es permitir que los pacientes puedan visualizar los resultados de sus exámenes de manera rápida y sencilla. Además, debe incluid un pequeño informe que explica los resultados, facilitando la comprensión y el seguimiento de los mismos por parte de los pacientes.

3. **Cargue de soportes:** Esta APP móvil estará pensada para que los pacientes puedan adjuntar los resultados de exámenes diagnósticos realizados externamente pero que son indispensables para el seguimiento exhaustivo de sus tratamientos y patologías. Aunque se originó desde el servicio de tele-consulta, está pensada para ser utilizada de manera transversal en todos los servicios de consulta externa, asegurando que los médicos tengan un contexto más amplio antes y durante las citas.

4. **Firma electrónica de documentos:** Originado desde el servicio de Gestión Documental, este requerimiento busca que los pacientes puedan recibir y firmar electrónicamente los diferentes documentos que actualmente se firman de manera física. Afirmando que esto no solo agilizaría el proceso sino que también contribuye a la seguridad y la eficiencia en la gestión documental.

5. **Tarjeta pacientes premium:** Solicitada por el servicio de mercadeo, esta aplicación móvil permitirá a los pacientes registrarse como premium y obtener

una tarjeta digital. Con ella, tendrán acceso a descuentos en diversos servicios, como cardiología no invasiva, imágenes diagnósticas, laboratorios clínicos, entre otros, mejorando la experiencia y los beneficios para los pacientes afiliados.

6. **Lista de citas:** El servicio de central de citas ha solicitado una aplicación móvil donde los pacientes puedan revisar sus citas pendientes. Además, podrán recibir alertas directamente en la aplicación sobre cambios en el estado de sus citas, mejorando la comunicación y la organización tanto para los pacientes como para el personal administrativo.

Todos los anteriores son una lista de 6 aplicaciones que se tienen como prioritarias o pensadas realizar en la institución a mediana o largo plazo, existen otras aplicaciones pero no se tienen en cuenta debido a que su impacto no sería relevante en la institución. Con esta lista de 6 aplicaciones se discute durante la reunión cual sería la aplicación ideal para la Super App de pacientes y es seleccionada la de Firma electrónica de documentos por las siguientes razones:

1. **Impacto alto en la institución:** La aplicación de Firma electrónica de documentos se identifica como una herramienta transversal, esencial en todos los servicios donde los pacientes interactúan y se requiere su consentimiento puede ser desde la autorización de uso de datos personales hasta los consentimientos informados previos a procedimientos médicos complejos, esta aplicación se convertirá en un pilar fundamental para optimizar y digitalizar estos procesos, garantizando una mayor eficiencia y seguridad.

2. **Eficiencia:** La implementación de la firma electrónica eliminará la necesidad de imprimir, firmar y escanear documentos físicos. Esto no solo acelera los procesos administrativos, sino que también contribuye a la reducción del uso de papel, generando un impacto positivo en el medio ambiente y en la reducción de costos operativos. La eficiencia en la gestión documental se traducirá en una atención más rápida y efectiva para los pacientes.

3. **Confidencialidad y seguridad:** Los documentos firmados electrónicamente ofrecen un nivel superior de seguridad. Cada firma puede ser verificada de igual

forma si la integridad del documento se mantiene intacta, esto, garantizando que la información no ha sido alterada post-firma y esto como consecuencia fortalece la confidencialidad de la información del paciente y asegura la trazabilidad y autenticidad de cada documento.

4. **Cumplimiento normativo:** La digitalización de la firma de documentos facilita el cumplimiento de las regulaciones y normativas vigentes en términos de seguridad y privacidad de la información. Al automatizar y digitalizar estos procesos, se minimizan los errores humanos, se garantiza la integridad de los datos y se fortalece la posición legal y reputacional de la institución en caso de auditorías o revisiones legales.

5. **Mejora la experiencia del paciente:** La aplicación contribuirá significativamente a reducir los tiempos de espera y los procesos por los cuales pasan los pacientes, pues estos ya recibirán los documentos a firmar con la mayor cantidad de datos completados desde el sistema, necesitando solo agregar su firma y completar datos específicos. Con esto se espera no solo agilizar la atención, sino que también mejore la percepción de los pacientes respecto a la eficiencia y la calidad del servicio proporcionado por la institución.

6. **Integración Técnica Compleja:** La integración técnica de esta aplicación es un desafío que, una vez superado, pavimentará el camino para la incorporación de otras aplicaciones en la Super App. La interacción con el ERP (SAP) a través de PO (Process Orchestration) será una prueba determinante para evaluar la capacidad técnica y funcional de la Super App, estableciendo un precedente de éxito para futuras integraciones.

7. **Etapa de Desarrollo Ideal:** La aplicación de Firma Electrónica ya cuenta con la aprobación institucional y se encuentra en una fase de desarrollo que se presume pueda llegar a facilitar su integración en la Super App. El financiamiento y los recursos ya asignados son indicativos de la confianza y el compromiso institucional hacia esta iniciativa, asegurando un respaldo sólido para su implementación y desarrollo.

En resumen y teniendo en cuenta todo lo anterior se puede afirmar que la decisión de seleccionar la App / Micro App de Firma electrónica de documentos como la primera para integrarse en la Super App de pacientes para la institución de salud de alta complejidad se fundamenta en una evaluación estratégica en cuanto a la necesidad de negocio y técnica, pues, no solo se alinea con las necesidades inmediatas de digitalización y eficiencia de la institución, sino que también sienta un precedente robusto para la integración futura de otras aplicaciones, además, este desarrollo se destaca por su capacidad para transformar y optimizar los procesos de los diferentes servicios de manera transversal en donde se busca mejorar la seguridad, la confidencialidad y la eficiencia en la gestión de documentos. Su implementación será un paso significativo hacia la búsqueda de una atención más ágil, segura y centrada en el paciente, demostrando el potencial y la versatilidad de la Super App en su conjunto. Para el desarrollo de esta App / Micro App se genera también el documento de especificación de requerimientos, los detalles se incluyen en el Anexo A.2.

6. Desarrollo de Super App

6.1. Comparación de Frameworks: React Native vs. Flutter

Según lo definido en la Sección 4.1.5 y el requerimiento descrito en 5.1.3 se realiza una búsqueda de las posibles herramientas para realizar tipos de aplicaciones multiplataforma, en este caso iOS y Android, encontrando lo siguiente:

Como se explica en el artículo Işitan and Koklu (2020), existen muchos frameworks para desarrollar aplicaciones móviles multiplataforma. Algunos de ellos son React Native, Flutter, Xamarin, Nativescript, Ionic Framework, Unity 3D, Cocos 2D, Titanium, Phonegap, Sencha Touch, Appcelerator Titanium, Apache Cordova, Rhodes, Onsen UI, Framework 7, Kony, Jasonette, iFactr, FeedHenry, Qt Corona. El uso de muchos de ellos ha disminuido considerablemente en los últimos años y algunos han comenzado a ser preferidos. En este estudio, en primer lugar, se determinaron los marcos que son tendencia en el último período se realizarán pruebas de rendimiento entre las aplicaciones a desarrollar. Para determinar los más preferidos, se utilizaron 3 plataformas de los cuales los desarrolladores obtienen más ayuda que son GitHub, StackOverflow y Google Trends. Con base en los datos obtenidos, se decidió comparar React Native, Flutter, Nativescript y Xamarin, ya que son las herramientas más populares para el desarrollo de aplicaciones móviles multiplataforma en los últimos años. En este mismo artículo de Işitan and Koklu (2020) se explica que el proceso de comparación consistió en ejecutar la misma aplicación la cual se escribirá en cada plataforma de desarrollo de aplicaciones móviles multiplataforma en donde se realizaron las siguientes mediciones:

1. Tamaño de la aplicación
    - El tamaño del recurso de la aplicación en el disco.
    - El tamaño del archivo de instalación de la aplicación.
2. Tiempo de Renderizado
3. Uso de los Recursos del Dispositivo

- Uso de CPU
- Uso de Memoria
- Consumo de Energía
- Uso de la Red

Los detalles y resultados de estas mediciones se pueden encontrar en el artículo Işitan and Koklu (2020), ya mencionado con anterioridad. Resumiendo las recomendaciones y resultados de este mismo artículo, se destaca que según los datos recolectados, Unity3D es uno de los frameworks más preferidos, ideal para aplicaciones con un enfoque en juegos o gráficos. Para elegir una herramienta de desarrollo de aplicaciones móviles multiplataforma, es crucial considerar el rendimiento de la aplicación y el soporte de desarrolladores. React Native y Flutter son populares en este campo, pero Flutter ha ganado terreno por su rendimiento superior. Un criterio importante es el soporte de software de terceros. Es beneficioso dividir la aplicación en módulos y verificar el soporte existente. React Native tiene la ventaja de usar Javascript, un lenguaje ya conocido por muchos, mientras que Flutter requiere aprender Dart. Aunque se recomienda React Native o Flutter, la elección final depende del desarrollador. Teniendo en cuenta este estudio realizado y sus recomendaciones finales se tendrán en cuenta como punto de partida para seleccionar entre React Native y Flutter como frameworks para el desarrollo de la Super App.

6.1.1. Uso actual de React Native y Flutter

En el punto anterior se preseleccionaron React Native y Flutter como frameworks para el desarrollo de la Super App, apoyándose en los resultados del artículo Işitan and Koklu (2020) del año 2000. Aunque el artículo no es excesivamente antiguo, la institución de salud tiene una política de bajo riesgo en la adopción de tecnologías emergentes. Por ello, se realizó una revisión adicional para validar la relevancia actual de estos frameworks. Los datos más recientes de Vailshery (2023), mostrados en la Figura 4, reafirman que React Native y Flutter siguen siendo líderes en el campo hasta 2022. Están respaldados por una extensa base de usuarios y una comunidad de

desarrolladores activa. Esta consistencia en el mercado garantiza que la Super App se construirá con tecnologías que no solo son actuales sino que también continúan siendo predominantes.

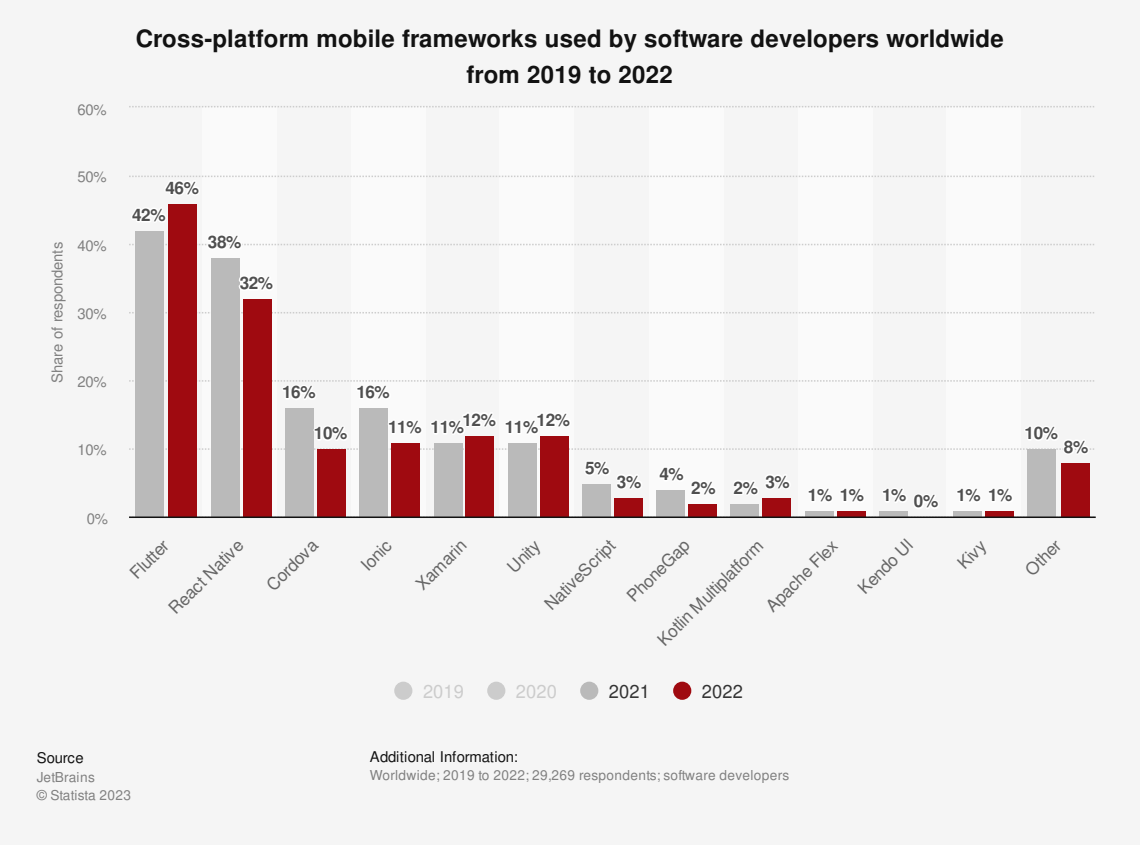


Figura 4: Distribución del uso de sistemas operativos móviles en el mundo Vailshery (2023)

6.1.2. ¿Super App con React Native o Flutter?

Para responder este interrogante se vio necesario considerar el stack tecnológico utilizado por la institución de salud, el cual se resumirá en la siguiente tabla:

Tipo de desarrollo	**Tecnología**	**Descripción**
Frontend	Nextjs	El Marco de React para la Web. Utilizado por algunas de las empresas más grandes del mundo, Next.js permite crear aplicaciones web de pila completa extendiendo las últimas características de React e integrando herramientas de JavaScript basadas en Rust para las compilaciones más rápidas (Vercel, 2023).
Backend	Nodejs + Expressjs	Nodejs es un entorno de ejecución de JavaScript de código abierto y multiplataforma (Foundation and contributors, 2023). Expressjs, infraestructura web rápida, minimalista y flexible para Node.js, API con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla (StrongLoop, 2023).
Aplicaciones móviles	React Native	React Native es un framework de JavaScript diseñado para desarrollar aplicaciones móviles reales con renderizado nativo tanto para iOS como para Android. Está basado en React, la biblioteca de JavaScript de Facebook para construir interfaces de usuario. Sin embargo, a diferencia de enfocarse en los navegadores, su objetivo son las plataformas móviles. Esto significa que los desarrolladores web pueden crear aplicaciones móviles que aparentan y funcionan como si fueran "nativa", utilizando una biblioteca de JavaScript ya conocida. Además, como la mayoría del código puede compartirse entre las plataformas, React Native facilita el desarrollo simultáneo para Android e iOS (Bonnie Eisenman, 2024).

Tabla 3: Stack tecnológico institución de salud.

Como se observa en la tabla anterior todo el stack tecnológico para el desarrollo de aplicaciones está enfocado en el lenguaje de programación Javascript, el cual es el requisito principal para pertenecer al área, esto con el fin de que los desarrolladores del equipo puedan entender y realizar tareas fullstack, es decir cualquiera podría entender, prestar soporte, desarrollar nuevas funcionalidades y crear proyectos de frontend, backend y aplicaciones móviles. Esta decisión ha sido tomada desde el año 2021 en la institución precisamente por la problemática que existía que se estaba desarrollando aplicaciones en diferentes lenguajes, lo cual se comenzaba a tener una dependencia de un desarrollador en caso de ser la única persona que conocía el lenguaje por lo cual se optó por tomar la decisión de unificar un único lenguaje con diferentes tecnologías y/o frameworks según un análisis realizado por el área. Siendo consecuentes con esto el lenguaje seleccionado para este desarrollo de Super App fue React native.

Tras analizar las herramientas disponibles para el desarrollo de aplicaciones móviles multiplataforma, React Native y Flutter se destacan como líderes en el sector, ambos con un amplio respaldo de usuarios y una comunidad de desarrolladores activa. La institución de salud ha establecido una política de coherencia en su stack tecnológico, y con un enfoque predominante en Javascript, React Native se presenta como la opción más coherente para el desarrollo de la Super App. Es importante mencionar que, si se optara por Flutter, implicaría la necesidad de aprender Dart, el lenguaje de programación desarrollado por Google, lo que podría representar un desafío adicional para el equipo y no se alinearía con el stack tecnológico de desarrollo de la institución.

Esta elección no solo se alinea con la estrategia tecnológica actual de la institución, sino que también garantiza una integración más fluida con las tecnologías ya implementadas. Además, facilita la colaboración entre los miembros del equipo de desarrollo, al mantener un lenguaje común y una base tecnológica unificada. En resumen, la decisión de adoptar React Native para la Super App refleja un enfoque bien fundamentado, orientado a maximizar la eficiencia y responder adecuadamente a las necesidades cambiantes del entorno de salud.

6.2. Estrategias de desarrollo de Super Apps con React native

Utilizando la versión paga de Chat GPT Plus, con acceso a búsquedas mejoradas mediante plugins de internet, se realizó una indagación en Bing y Scholar AI, complementando con consultas en Google y Google Scholar. El propósito fue hallar métodos técnicos para desarrollar Super Apps móviles usando React Native. Esta exploración arrojó dos enfoques principales que se evaluaron para este proyecto.

6.2.1. Enfoque con Re.Pack y Module Federation

Teniendo en cuenta la definición de Re.Pack en la Sección 4.1.11 y Module Federation en la Sección 4.1.12, y el articulo A Step-By-Step Guide to Super App Development, Callstack (2023b), el cual, proporciona una guía paso a paso sobre el desarrollo de super aplicaciones, con énfasis en el uso de React Native y herramientas como Re.Pack y Module Federation. En este se detalla cómo configurar estas herramientas haciendo uso de dos aplicaciones una nombrada como HostApp y la otra como MiniApp, dentro de un monorepo, para demostrar la integración y la reutilización de componentes entre ellas. Se aborda la configuración de webpack y el ModuleFederationPlugin para facilitar la compartición de dependencias y librerias, y por ultimo se explican las estrategias para manejar las librerias y los estados compartidos dentro del ecosistema de la Super App que se plantea.

La guía destaca la necesidad de alinear dependencias para evitar conflictos de versiones y discute las complejidades de manejar activos entre aplicaciones federadas, ofreciendo soluciones como la carga de activos remotos o su incrustación directa en el paquete. También se examina la gestión del estado compartido, sugiriendo la extracción del estado compartido en un módulo separado para facilitar su uso en aplicaciones múltiples.

Para clarificar la metodología descrita, se presenta la Figura 5. Esta visualización, en conjunto con lo escrito previamente, facilita la comprensión de varios puntos:

- La aplicación principal, o Super App, así como las Mini Apps que se integrarán, deben ser desarrolladas utilizando React Native.

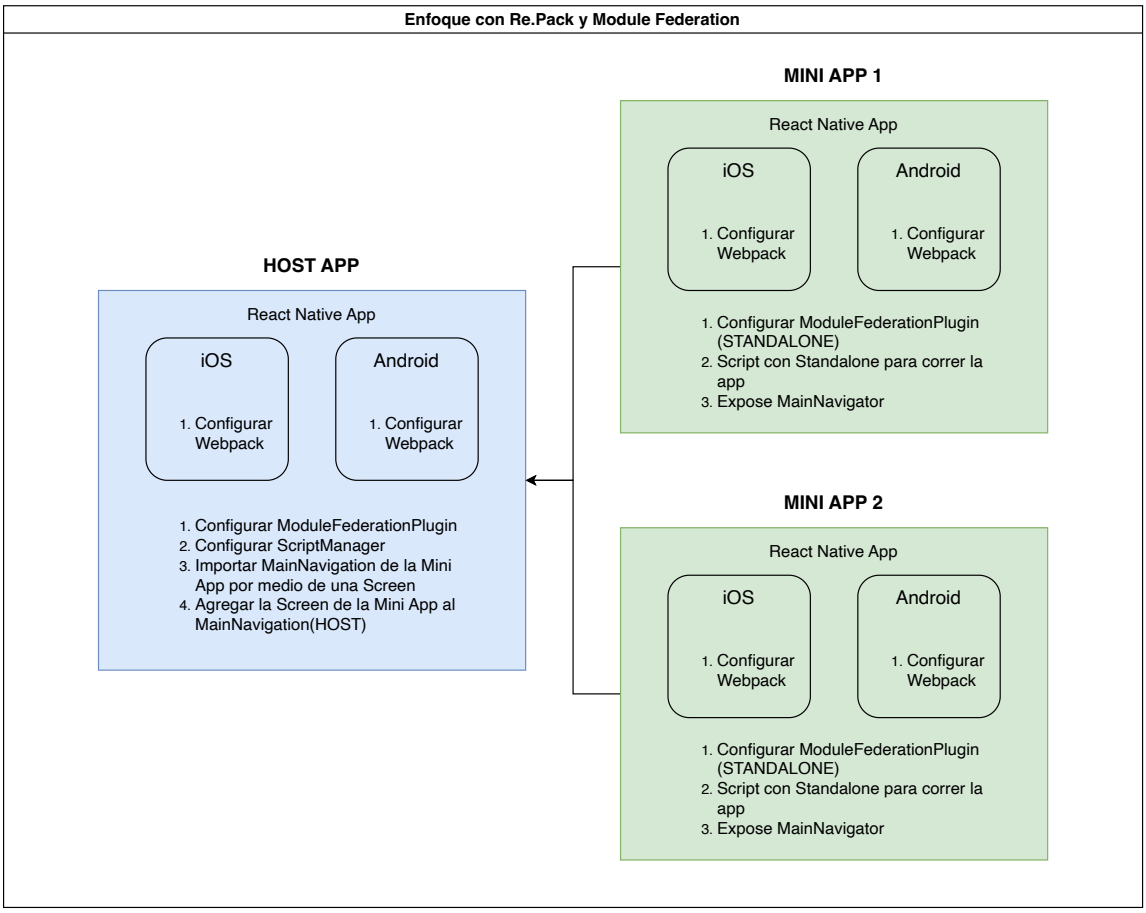


Figura 5: Enfoque con Re.Pack y Module Federation

- Las Mini Apps necesitan una configuración específica para que expongan su MainNavigator, permitiendo así definir cómo se navegará desde la Super App.
- Se hace un uso intensivo del ModuleFederationPlugin, una herramienta de código abierto creada por Callstack. A pesar de ser de código abierto, es importante notar que procede de una compañía privada.

6.2.2. Enfoque con Webview

Para aplicaciones nativas en Android según GoogleLLC (2023) en su página de desarrollo afirman que: Si se desea publicar una aplicación web (o solo una página web) como parte de una aplicación cliente, se puede usar WebView para hacerlo. La clase WebView es una extensión de la clase View de Android que permite mostrar páginas web como parte del diseño de una actividad. No incluye las funciones de un navegador web completamente desarrollado, como controles de navegación o una barra de direcciones. De forma predeterminada, todo lo que hace WebView es mostrar una página web. Una situación común en la que sirve usar WebView es cuando, en la app, se desea proporcionar información que tal vez deba actualizarse, como en un acuerdo de usuario final o una guía del usuario. Dentro de la app para Android, se puede crear una Activity que contenga una WebView y, luego, usarla para mostrar el documento que está alojado en línea. Otra situación en la que WebView puede resultar útil es si la app proporciona al usuario datos que siempre requieren una conexión a Internet para recuperar información, como el correo electrónico. En ese caso, en lugar de realizar una solicitud de red, analizar los datos y renderizarlos en un diseño de Android, posiblemente sea más fácil crear una WebView en la app para Android que muestre una página web con todos los datos del usuario. Se puede diseñar una página web específica para dispositivos Android y, luego, implementar una WebView en la app para Android que cargue esa página.

Para aplicaciones nativas en iOS, según AppleInc. (2023) en su página de desarrollo, se afirma que: Un objeto WKWebView es una vista nativa de la plataforma que se utiliza para incorporar contenido web de forma fluida en la interfaz de usuario de una aplicación. Una vista web soporta una experiencia completa de navegación

por Internet y presenta contenido HTML, CSS y JavaScript junto a las vistas nativas de la aplicación. Se recomienda usarlo cuando las tecnologías web satisfacen más fácilmente los requisitos de diseño y estilo de la aplicación que las vistas nativas. Por ejemplo, se podría usar cuando el contenido de una aplicación cambia con frecuencia. Una vista web ofrece control sobre la navegación y la experiencia del usuario a través de objetos delegados. Se utiliza el delegado de navegación para reaccionar cuando el usuario hace clic en enlaces en el contenido web, o interactúa con el contenido de una manera que afecta la navegación. Por ejemplo, se podría evitar que el usuario navegue a nuevo contenido a menos que se cumplan condiciones específicas. Se usa el delegado de UI para presentar elementos de UI nativos, como alertas o menús contextuales, en respuesta a interacciones con el contenido web.

En la discusión previa, se observa cómo tanto Android como iOS ofrecen soluciones para renderizar e interactuar con contenido web dentro de sus plataformas nativas. A partir de estas capacidades integradas, la comunidad de desarrolladores ha creado un módulo para React Native, conocido como react-native-webview, ReactNativeCommunity (2023a). Este módulo, disponible en GitHub, facilita la incorporación de componentes WebView en proyectos de React Native. Originalmente parte del Core de React Native, react-native-webview fue liberado en 2018 y entregado a la comunidad para su mejora y evolución continua.

Dentro del repositorio de GitHub, se ofrece una documentación detallada, conocida como React Native WebView API Reference, ReactNativeCommunity (2023b), que presenta una serie de métodos de la API. Estos permiten interacciones fluidas entre las aplicaciones móviles de React Native y el contenido web que se desea integrar. De igual manera siguiendo la búsqueda se encuentra un artículo artículo publicado por ArrowHiTech (2023), llamado React native webview: The comprehensive guide you need to know. En este se explica que React Native WebView permite a las aplicaciones cargar páginas web, y aunque antes era parte del núcleo de React Native, ahora debe instalarse como una biblioteca separada. La guía detalla el proceso de instalación y configuración de la biblioteca, tanto para proyectos de iOS como de Android, incluyendo pasos como la habilitación de AndroidX y la ejecución de pod install para iOS.

El artículo prosigue mostrando cómo se utilizan las propiedades básicas de WebView, como cargar una URL o HTML directamente en el componente. Un aspecto importante que cubre es el manejo de los cambios en el estado de la navegación mediante la función onNavigationStateChange, que permite realizar acciones adicionales en respuesta a la navegación del usuario, como detener la carga o inyectar JavaScript.

También, se abordan temas como el soporte para la carga y descarga de archivos en WebView, detallando la configuración de los permisos necesarios en ambas plataformas y cómo controlar las cargas de archivos múltiples o individuales. También se explora cómo inyectar código JavaScript dentro de WebView, utilizando métodos como injectedJavaScript, injectJavaScript y postMessage. De igual manera que en el enfoque anterior se busca clarificar la metodología descrita por medio de la Figura 6 lo cual también permitió concluir lo siguiente:

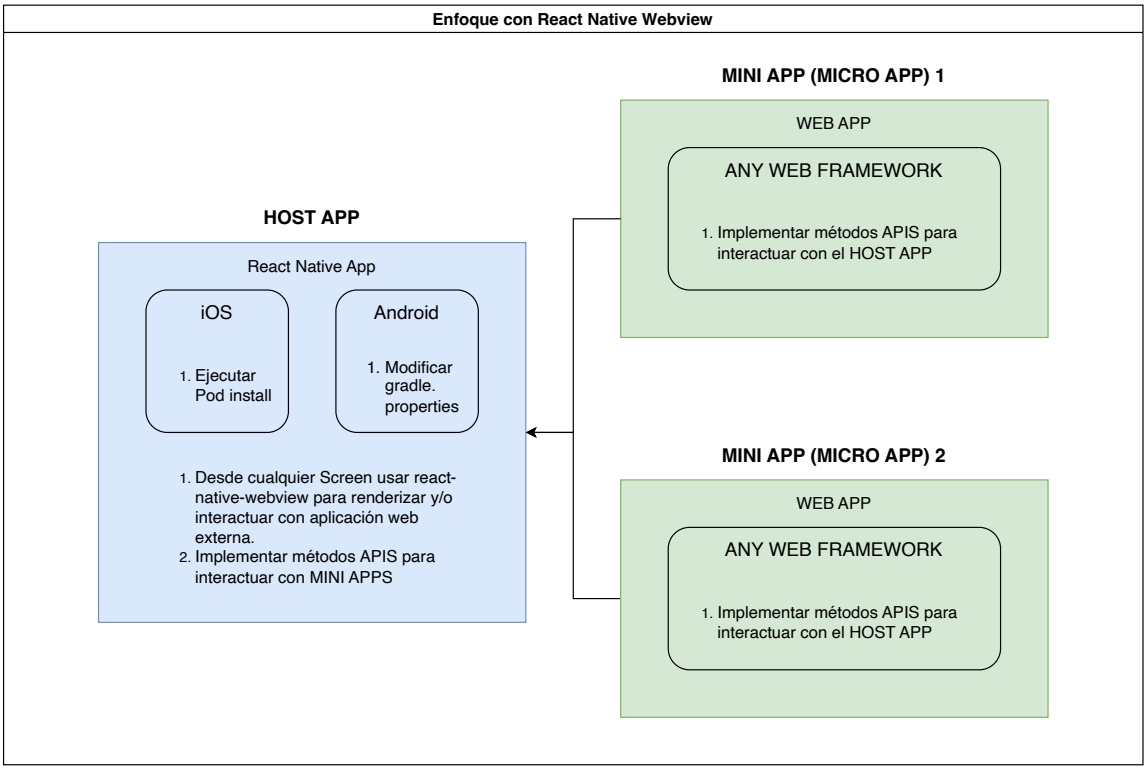


Figura 6: Enfoque con Webview

- El módulo react-native-webview, permite una interacción más directa entre las aplicaciones nativas y el contenido web, soportando tanto la carga de páginas web como el manejo de contenido dinámico y la inyección de JavaScript.
- No importa el Framework, herramienta, arquitectura o lenguaje con el cual se ha desarrollado la Micro App, después de que sea una aplicación WEB podría integrarse.
- Es totalmente Open Source, mantenido por comunidad de desarrolladores activos.

6.2.3. Enfoque seleccionado

El enfoque seleccionado para el desarrollo de la Super App en este proyecto de grado fue el uso de WebView, por las siguientes razones y ventajas:

- **Integración versátil con Micro Apps:** WebView permite la integración con Micro Apps desarrolladas en cualquier lenguaje de programación. Esto es crucial, ya que la institución de salud de alta calidad trabaja con diversos proveedores de software, y las Micro Apps futuras no estarán limitadas a ser desarrolladas exclusivamente en React Native.
- **Independencia de proveedores de software:** Al ser React-native-webview un proyecto de código abierto, su uso y mantenimiento no dependen de una empresa privada. Esto asegura una mayor independencia y evita costos asociados con licencias o dependencias de proveedores específicos.
- **Flexibilidad en el desarrollo y mantenimiento:** La naturaleza de código abierto de React-native-webview permite una adaptación y mejora continua por parte de la comunidad de desarrolladores. Esto facilita la personalización y la implementación de mejoras de seguridad específicas para el proyecto.
- **Rapidez y facilidad en la Integración de contenidos web:** Utilizar WebView facilita la incorporación de contenido web existente dentro de la Super

App, lo que puede ser más rápido y facil que desarrollar funcionalidades equivalentes desde cero en React Native.

- **Rapidez en el desarrollo y lanzamiento al mercado:** Utilizar WebView puede acelerar significativamente el proceso de desarrollo y lanzamiento de la Super App. Al integrar Micro Apps existentes, se reduce el tiempo y el esfuerzo necesarios para desarrollar nuevas funcionalidades desde cero.

- **Facilidad de actualización y mantenimiento:** Con WebView, las actualizaciones de las Micro Apps pueden realizarse de manera independiente sin necesidad de actualizar toda la Super App. Esto significa que los cambios o mejoras en las Micro Apps se pueden implementar rápidamente y con menor impacto en el sistema general. Además de esto al realizar una actualización en la Micro App, no será necesario el despliegue de la Super App de nuevo en las tiendas de Play Store (Android) y App Store (iOS) y el usuario podrá ver los cambios de manera casi inmediata.

- **Flexibilidad en la integración de contenido:** WebView no solo permite integrar aplicaciones web, sino también contenido interactivo como videos, mapas y gráficos, lo cual puede enriquecer la funcionalidad y el atractivo visual de la Super App.

Como se puede apreciar en los puntos anteriores, este enfoque ofrece numerosas ventajas. Sin embargo, es importante también resaltar algunos riesgos o consideraciones que requieren atención:

- **Rendimiento y experiencia de usuario:** Es crucial evaluar el impacto en el rendimiento y la experiencia de usuario al integrar Micro Apps mediante WebView. Aunque es una solución flexible, puede haber diferencias en la velocidad y la fluidez en comparación con las aplicaciones nativas.

- **Seguridad y aislamiento de aplicaciones:** Es esencial prestar atención a las estrategias de seguridad, especialmente en lo que respecta al aislamiento entre la Super App y las Micro Apps integradas. Es crucial garantizar que las

Micro Apps no comprometan la seguridad de la aplicación principal. Al tener las Micro Apps corriendo en servidores externos, se debe procurar el uso de certificados SSL, además, de restringir de manera correcta de la información que se comparten entre ellas.

- **Compatibilidad y pruebas:** Es importante asegurar que las Micro Apps cuenten con sus respectivas pruebas de manera independiente para mantener la calidad del software.
- **Actualizaciones y mantenimiento a largo plazo:** Se debe planificar cómo se manejarán las actualizaciones tanto de la Super App como de las Micro Apps integradas. Esto incluye la gestión de cambios en las APIs de WebView y React Native, así como en las Micro Apps individuales.

6.3. Arquitectura de la solución

Es importante resaltar que la Super App es una parte integral del ecosistema de software que estamos desarrollando, pero no es la totalidad de la solución. Es crucial en la arquitectura de la solución considerar la interacción y la integración eficiente de todos los componentes, sistemas, servicios e infraestructura necesarios. Es importante resaltar lo definido en la Sección 4.1.7, donde la arquitectura de software y de integración se basa en prácticas comprobadas que aseguran la viabilidad técnica y operativa a largo plazo de la solución de software. Con esto en mente, se propone una metodología iterativa para el desarrollo de la arquitectura de software la cual permitirá ir mejorando atributos de calidad que sean necesarios durante el desarrollo, implementación y vida útil de la solución que se planteó. Para comprender la metodología planteada se recomienda leer las siguientes secciones del marco teórico:

- Importancia de la arquitectura de software. (Sección 4.1.13)
- Especificación de requisitos de atributos de calidad y decisiones de diseño arquitectónico. (Sección 4.1.14)
- Requisitos Arquitectónicamente Significativos (ASRs). (Sección 4.1.15)

- ADD (Attribute-Driven Design). (Sección 4.1.16)

6.3.1. Metodología práctica para el desarrollo de la arquitectura de software de la solución

Basado en la información detallada en las secciones anteriores, se propuso una metodología estructurada que permitirá esbozar la arquitectura inicial del software y proporcionará una guía para su refinamiento continuo ante nuevos desafíos o requerimientos de negocio. La metodología se articula en una serie de pasos definidos en una plantilla, que incluye:

1. **ASR:** Comenzar por el ASR que se ha seleccionado para trabajar según prioridad del negocio.

   1.1 **Nombre del ASR:** Debe ser un atributo de calidad.

   1.2 **Método de obtención:** Teniendo en cuenta que lo investigado en los puntos anteriores indica que las posibles fuentes pueden variar, en este campo es necesario relatar de manera simple como se llega a la priorización del actual ASR.

   1.3 **Unidad:** Especificar la unidad de medida (por ejemplo, porcentaje de tiempo, minutos, milisegundos para latencia).

   1.4 **Respuesta esperada:** Definir cuantitativamente los resultados esperados tras la iteración de la metodología propuesta.

   Los pasos subsiguientes se enfocan en la caracterización del ASR para comprender el contexto de aplicación:

   1.5 **Caracterización del ASR:**

      1.5.1 **Actor:** Cualquier entidad que interactúe con el sistema.

      1.5.2 **Estimulo:** Acción o evento que requiere una respuesta del sistema.

      1.5.3 **Ambiente:** Condiciones operativas del sistema al recibir el estímulo.

1.5.4 **Artefacto:** Componentes del sistema involucrados en responder al estímulo.

1.5.5 **Respuesta esperada:** La reacción o resultado que el sistema debe generar ante el estímulo.

Es crucial reconocer que al priorizar un atributo de calidad se pueden afectar otros, lo cual se debe documentar:

1.6 **Atributo impactado:** Identificar el atributo de calidad afectado.

1.7 **Impacto:** Describir cómo se afecta este atributo.

2. **ADD (Attribute-Driven Design):** En esta fase se incluyen variaciones a los pasos del ADD tradicional.

2.1 **Entradas:** Las entradas estándar para este proceso incluyen propósitos comerciales, requisitos funcionales, restricciones y preocupaciones. Se establece que deben contener de forma obligatoria los siguientes elementos.

2.1.1 **Requerimientos funcionales:** Detalles de las funcionalidades que el sistema debe proveer, incluyendo procesos y comportamientos esperados.

2.1.2 **Requerimientos no funcionales:** Los requerimientos no funcionales especifican cómo un sistema debe comportarse y establecen estándares como eficiencia y seguridad. Por ejemplo, un sistema puede requerir que "el tiempo de respuesta para las transacciones financieras no debe exceder dos segundos, incluso bajo carga máxima de usuarios".

2.1.3 **Restricciones técnicas:** Limitaciones impuestas por la infraestructura actual o por el uso de tecnologías específicas.

2.1.4 **Restricciones de negocio:** Limitaciones dictadas por factores económicos, legales o de mercado que afectan la implementación del sistema.

2.2 **Motivadores de iteración:** A través de estos, definimos los objetivos del diseño arquitectónico.

2.2.1 **Motivador:** Nombre corto del motivador.

2.2.2 **Descripción:** Explicación de por qué este elemento es un motivador clave, con un enfoque en los objetivos de negocio.

2.3 **Elementos del sistema a refinar:** Selección de componentes de la arquitectura que deben mejorarse para cumplir con requisitos específicos.

2.3.1 **Componente identificado:** Nombre del componente seleccionado.

2.3.2 **Descripción del componente:** Descripción detallada del componente y su función.

2.4 **Conceptos para cumplir con motivadores:** Definición de alternativas de diseño, selección de estilos, patrones y tácticas arquitectónicas.

2.4.1 **Estilo de arquitectura:** Nombres de un estilo de arquitectura de software.

2.4.2 **Justificación estilo:** Explicación del motivo por el cual se eligió este estilo en particular.

2.5 **Componentes, responsabilidades y relaciones:** En este se busca materializar todo lo diseñado en el punto 2.4 en elementos concretos de la arquitectura, se dejan claras las responsabilidades y se definen relaciones.

2.5.1 **Nombre:** Designación del componente arquitectónico.

2.5.2 **Responsabilidad:** Detalle de las funciones o procesos a cargo del componente.

2.5.3 **Relaciones:** Enumeración de las interacciones del componente con otros elementos del sistema.

2.6 **Diagramas de puntos de vistas y de conocimiento:** Representación gráfica de las vistas arquitectónicas para evaluar la coherencia y la completitud, además de documentar visualmente los elementos implicados.

2.6.1 **Punto de vista de contexto:** Describe las relaciones, dependencias e interacciones entre el sistema a diseñar y su ambiente (personas, sistemas, entidades externas, etc.).

2.6.2 **Punto de vista funcional N1:** Detalle de los elementos funcionales del sistema, sus responsabilidades principales, interfaces e interacciones.

2.6.3 **Punto de vista funcional N2 (Opcional):** Similar al anterior, pero con mayor nivel de detalle o enfoque.

2.6.4 **Punto de vista de información:** Ilustra la forma en que el sistema almacena, manipula, maneja y distribuye información.

2.6.5 **Punto de vista de desarrollo:** Describe cómo la arquitectura soporta el proceso de desarrollo del sistema.

2.6.6 **Punto de vista despliegue:** Describe el ambiente en el cual el sistema será desplegado, incluyendo sus dependencias tecnológicas.

2.6.7 **Punto de vista de operación (Opcional):** Describe cómo el sistema operará, será administrado y soportado cuando esté en producción.

2.7 **Validación del cumplimiento de motivadores:** Este paso tiene como finalidad la comprobación, junto con los stakeholders, colegas y demás partes interesadas, de la consecución de los objetivos primordiales al finalizar una iteración o fase de diseño.

2.7.1 **Descripción de la validación:** Se elabora una descripción del método de validación a implementar, la cual deberá ser coherente con los lineamientos especificados en los literales 1.3 y 1.4 de la metodología.

2.7.2 **Resultado de la validación:** Se debe presentar de forma explícita y cuantificable si los resultados obtenidos se alinean con las expectativas previamente definidas en los literales 1.3 y 1.4 de la metodología.

6.3.2. Selección de ASRs para la implementación inicial de la arquitectura de la Super App en la institución de salud de alta complejidad.

Las decisiones arquitectónicas deben alinearse estrictamente con los intereses comerciales y los objetivos estratégicos. Refiriéndonos a la Sección 3.1, donde se establece el objetivo general del proyecto, se busca asegurar que la solución propuesta aborde

tres aspectos fundamentales: modularidad, integración y seguridad. Estos constituyen los pilares más críticos para la institución de salud de alta complejidad y, por tanto, han servido como ejes centrales para la toma de decisiones arquitectónicas.

1. **ASR de Integrabilidad:** En Bass et al. (2022, Capítulo 7), La integrabilidad, en el contexto del desarrollo de software, se refiere a la capacidad de integrar eficientemente componentes de software, considerando los costos y riesgos técnicos asociados. Esto implica no solo hacer que componentes desarrollados de manera independiente cooperen entre sí, sino también gestionar los riesgos relacionados con el cronograma, el rendimiento y la tecnología. La integración puede involucrar la incorporación de una unidad de software, C, o un conjunto de unidades, C1, C2, ..., Cn, en un sistema, S. Este sistema puede ser una plataforma o un sistema existente que ya contiene ciertos componentes y se espera integrar más. Una consideración clave es el nivel de control sobre el sistema S y el grado de comprensión de los componentes Ci, especialmente cuando estos provienen de proveedores externos. Este atributo de calidad es desafiante porque implica planificar para el futuro con información incompleta. Algunas integraciones son más sencillas que otras, dependiendo de si han sido anticipadas y acomodadas en la arquitectura del sistema. Por otro lado, otras pueden ser más complejas si no se han previsto. Ver anexo A.3 donde se aplica la metodología práctica planteada en la Sección 6.3.1.

2. **ASR Modularidad:** Según Ciceri et al. (2022, Capítulo 4) la modularidad en el desarrollo de software es un principio introducido por David Parnas en la década de 1970. Según Parnas, un módulo debe contener solo una decisión de diseño (encapsulación) y la estructura de datos para esta decisión de diseño debe estar encapsulada dentro de la localidad del módulo. En lenguajes de programación modernos, los módulos son unidades dentro de un sistema de software, como clases, componentes o capas. Nuestro cerebro prefiere razonar sobre sistemas en varios niveles de unidades para lograr un aumento de capacidad en nuestra memoria. Lo crucial es que nuestro cerebro se beneficia de estas unidades solo si los detalles pueden representarse como una unidad coherente

que forma algo significativo. Las unidades de programa que combinan elementos arbitrarios y no relacionados no son significativas y no serán aceptadas por nuestro cerebro. Por lo tanto, un sistema modular con unidades de programa coherentes y significativas tendrá una baja deuda técnica y una baja complejidad innecesaria. Ver anexo A.4 donde se aplica la metodología practica creada en la Sección 6.3.1.

3. **ASR de Seguridad:** Bass et al. (2022, Capítulo 11), define la seguridad como la medida de la capacidad del sistema para proteger datos e información del acceso no autorizado, al mismo tiempo que permite el acceso a personas y sistemas autorizados. Un ataque, es decir, una acción contra un sistema informático con la intención de causar daño, puede tomar varias formas. Puede ser un intento no autorizado de acceder a datos o servicios, modificar datos o intentar negar servicios a usuarios legítimos. El enfoque más simple para caracterizar la seguridad se centra en tres características: confidencialidad, integridad y disponibilidad (CIA):

   - **Confidencialidad:** Es la propiedad de que los datos o servicios estén protegidos contra el acceso no autorizado. Por ejemplo, un hacker no puede acceder a tus declaraciones de impuestos en una computadora gubernamental.
   - **Integridad:** Se refiere a que los datos o servicios no están sujetos a manipulación no autorizada. Por ejemplo, tu calificación no ha sido cambiada desde que tu instructor la asignó.
   - **Disponibilidad:** Es la propiedad de que el sistema estará disponible para uso legítimo. Por ejemplo, un ataque de denegación de servicio no te impedirá pedir un libro en una librería en línea.

   Ver anexo A.5 donde se aplica la metodología practica creada en la Sección 6.3.1.

Al tener claro una metodología de arquitectura de software y unos atributos de calidad seleccionados se realizan los siguientes desarrollos basados por cada documento

generado.

6.4. Desarrollos basados en el ASR de integrabilidad

Considerando la Figura 7, extraída del Anexo A.3, el diagrama funcional Nivel 2 destaca por ser el primer documento de arquitectura para la solución de la Super App. Contiene la mayoría de los nuevos desarrollos y modificaciones a desarrollos existentes, realizados de la siguiente manera:

6.4.1. Isolated patient authentication microservice (Microservicio aislado de autenticación de pacientes), Módulo integración autenticación pacientes y Módulo funcionalidades autenticación pacientes en el ERP (SAP).

La Figura 7 ayudó a tomar la decisión de priorizar el desarrollo del microservicio aislado de autenticación de pacientes, sus módulos de integración y funcionalidades en el erp, pues se puede observar como este es un pre-requisito crítico para el backend de la Super App. El desarrollo se llevó a cabo por un equipo multidisciplinario de la institución de salud de alta complejidad, con integración en el ERP SAP de la siguiente forma:

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	Especificación, arquitectura y desarrollo del microservicio.
Analista especialista ABAP	Desarrollo de funcionalidades de autenticación y creación de tablas para datos de pacientes.
Analista especialista INTEGRADOR	Desarrollo de integración de autenticación de pacientes (SAP PI/PO).

Tabla 4: Equipo responsable del desarrollo de microservicio aislado de autenticación de pacientes.

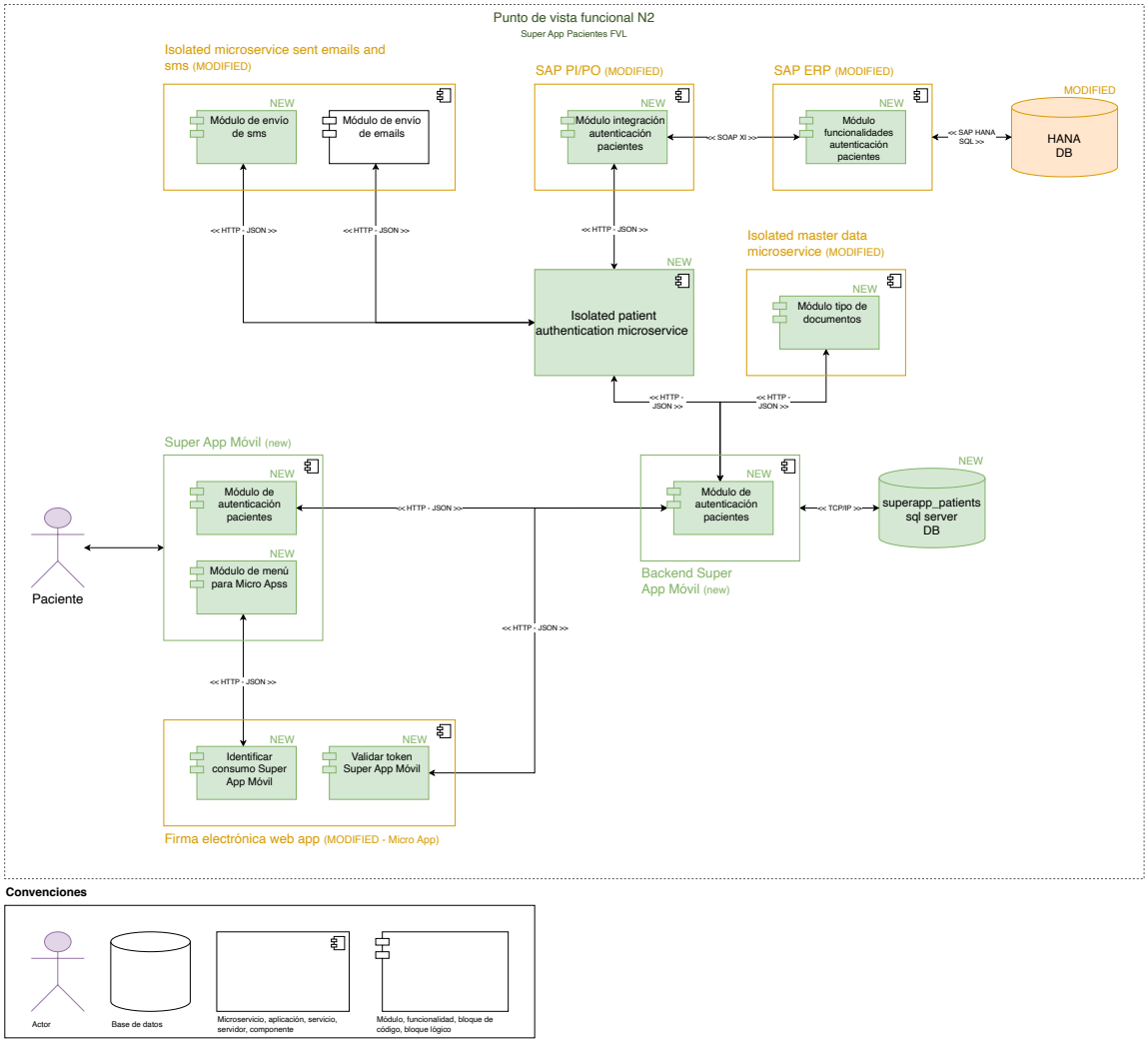


Figura 7: Vista funcional nivel 2, ASR Integrabilidad. Anexo A.1

Después de 4 meses, se obtuvieron los siguientes resultados para la institución:

1. Documento de especificación del microservicio.
2. Documentación de arquitectura de software.
3. Se ha desarrollado módulo de integración de autenticación de pacientes en SAP PI/PO.
4. Se ha desarrollado módulo con funcionalidades para la autenticación de paciente en el ARP de SAP de la institución.
5. Funcionalidades en ERP (SAP) con contraseña y roles centralizados para pacientes.
   Nota: Los roles centralizados asignan al paciente un rol según el servicio prestado o a prestar en la institución, visible al autenticarse y determinando el acceso a Micro Apps en la Super App.
6. Se ha desarrollado un microservicio independiente para autenticación de pacientes, integrable con el backend de la Super App y otros sistemas de la institución, incluyendo proveedores externos. Este microservicio brinda funcionalidades a través de una REST API, que incluye servicios como:
   - **GET /ping-micros,** permite realizar un ping al microservicio y conocer si está online.
   - **POST /patient/verify-exist,** permite verificar por medio del tipo de documento y número de documento si es o ha sido paciente en la institución de salud de alta complejidad.
   - **POST /patient/security-questions,** permite obtener unas preguntas de seguridad para validar la identidad del paciente por medio del tipo de documento y número de documento.
   - **POST /patient/check-answers,** permite verificar las respuestas a las preguntas generadas en el servicio anterior.

- **POST /patient/create-password,** permite la creación de un password, por medio de un código el cual ha sido enviado al email o número celular del paciente.
- **POST /patient/login,** permite realizar la autenticación e inicio de sesión de un paciente, creando un JWT.
- **POST /patient/send-code-verify-email,** genera y envía un código a un email para poder realizar el proceso de verificación del email.
- **POST /patient/verify-email,** Permite el envío de datos para la verificación del email.
- **POST /patient/token-validation-refresh,** permite validar el json web token (JWT), además de esto si es válido genera un JWT nuevo como refresh.
- **POST /patient/send-code-verify-phone,** genera y envía un código a un número de celular para poder realizar el proceso de verificación del número de celular.
- **POST /patient/verify-phone,** permite el envío de datos para la verificación del número de celular.

6.4.2. Actividades de modificación del microservicio aislado de envío de emails y SMS

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	Especificación y desarrollo del módulo de envío de SMS.
Analista de calidad de software	Coordinación con el proveedor de SMS (MASIV).
Supervisor operativo centro de contacto	Coordinación con el proveedor de SMS (MASIV).

Tabla 5: Equipo responsable del desarrollo de modificación del microservicio aislado de envío de emails y SMS

Después de 1 mes, se lograron los siguientes avances:

1. Actualización de la documentación y cambio de nombre del servicio a microservicio de envío de emails y SMS.

2. La institución de salud de alta complejidad inicialmente tenía un servicio de Masiv para envío de envío de mensajes de texto (SMS) centrados en mercadeo, lo que implicaba una baja prioridad y retrasos de hasta 8 horas. Tras negociar con Masiv, ahora se dispone de una segunda cuenta optimizada que permite el envío prioritario de SMS para códigos de seguridad y verificaciones, con una promesa de envió inmediato.

3. Nuevas funcionalidades del módulo accesibles vía REST API, incluyendo:

    - **POST /sms/send-sms-simple,** permite él envió de un mensaje de texto a un único receptor, puede agregar caracteres especiales y URLs.

El desarrollo y las modificaciones mencionadas hasta el momento fueron cruciales para la implementación integral de la Super App, pero un poco más enfocadas al microservicio aislado de autenticación de pacientes. Los siguientes desarrollos se enfocarán en las innovaciones y características únicas de la Super App.

6.4.3. Actividades de modificación de microservicio aislado de datos maestros

Cargo	Actividades
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	Especificación y desarrollo del módulo de tipos de documentos.
Analista especialista ABAP	Desarrollo de funcionalidad para obtener tipos de documento en el ERP SAP.
Analista especialista INTEGRADOR	Desarrollo de integración hacer uso de la funcionalidad de obtener datos de tipos de documentos por medio de PI/PO SAP.

Tabla 6: Equipo responsable del desarrollo de modificación de microservicio aislado de datos maestros

En 2 semanas, se consiguieron los siguientes resultados:

1. Una nueva funcionalidad en el microservicio aislado de datos maestros para obtener los tipos de documentos desde una fuente centralizada la cual es el ERP SAP.

2. Actualización de librerías y versiones de servidores.

3. Nueva funcionalidad de tipos de documento disponible para la Super App y otras aplicaciones por medio de REST API:

    - **POST /documents-types-all,** permite la obtención de todos los tipos de documentos posibles al momento de realizar un primer ingreso de los pacientes en el ERP SAP.

Para los desarrollos que se nombraran a continuación del Backend y Frontend de la Super App se hace uso del documento de especificación creado en la Sección 5.

6.4.4. Actividades de desarrollo del Backend para la Super App: Módulo de Autenticación de Pacientes

Cargo	Actividades
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	Especificación y desarrollo del módulo de autenticación de pacientes en el backend de la Super app.

Tabla 7: Equipo responsable del desarrollo del Backend para la Super App: Módulo de Autenticación de Pacientes

En un tiempo de 2 semanas, se logró lo siguiente:

1. Documento de especificación del backend de la Super App, inicialmente del módulo con todas las funcionalidades referentes a la autenticación.

2. Se completó la creación y configuración de la base de datos para el backend de la Super App (superapp_patients) en SQL Server, así como la generacion de las tablas necesarias.

3. Se ha desarrollado un microservicio independiente para autenticación de pacientes, integrable con el backend de la Super App y otros sistemas de la institución, incluyendo proveedores externos. Este microservicio brinda funcionalidades a través de una REST API, que incluye servicios como:

   - **GET /ping-server,** permite realizar un ping al servidor y conocer si está online.
   - **POST /patient/documents-types-all,** permite obtener los tipos de documentos disponibles y su descripción en idioma español o ingles según parámetro en la petición.
   - **POST /patient/login,** permite realizar la autenticación y almacena la información en la base de datos.

- **POST /patient/verify-exist,** permite verificar por medio del tipo de documento y número de documento si es o ha sido paciente en la institución de salud de alta complejidad.
- **POST /patient/send-code-verify-email,** genera y envía un código a un email para poder realizar el proceso de verificación del email.
- **POST /patient/send-code-verify-phone,** genera y envía un código por medio de SMS a un número de celular para poder realizar el proceso de verificación del número de celular.
- **POST /patient/create-password,** permite la creación de un password, por medio de un código el cual ha sido enviado al email o número celular del paciente por medio de SMS.
- **POST /patient/security-questions,** permite obtener unas preguntas de seguridad para validar la identidad del paciente por medio del tipo de documento y número de documento.
- **POST /patient/check-answers,** permite verificar las respuestas a las preguntas generadas en el servicio anterior.
- **POST /patient/verify-email,** permite el envío de datos para la verificación del email.
- **POST /patient/verify-phone,** permite el envío de datos para la verificación del número de celular.

6.4.5. Actividades de desarrollo Super App: Módulo de Autenticación de Pacientes

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Especificación y desarrollo del módulo de autenticación de pacientes de la Super App.

Tabla 8: Equipo responsable del desarrollo Super App: Módulo de Autenticación de Pacientes.

En un tiempo de 6 semanas, se logró lo siguiente:

1. Se desarrollan las interfaces y funcionalidades del aplicativo móvil de la Super App, teniendo en cuenta lo especificado en la Sección 5.

2. Para este desarrollo se tiene en cuenta el manual de identidad de la institución de salud de alta complejidad, teniendo en cuenta características como tipografía, gama cromática sugerida y correcto uso de marca.

3. En el manejo de la autenticación centralizada a través de la Super App, inicialmente se empleó la librería AXIOS (Sección 4.1.17) para consumir el backend mencionado en la sección anterior. Aunque esta librería funcionó adecuadamente durante la fase de desarrollo, se presentaron inconvenientes al crear una versión de prueba "build release" para dispositivos móviles Android, donde los llamados al backend no operaban correctamente. La investigación de este problema reveló que es un error común en proyectos que utilizan React Native y AXIOS, tal como se documenta en varias fuentes y se verifica en /facebook/react-native/issues/24039. La solución sugerida en estos informes implica modificar archivos de configuración específicos de Android. Ante esto, se decidió cambiar de estrategia y prescindir del uso de librerías externas para las peticiones, optando en su lugar por el uso del API FETCH nativo de JavaScript (Sección 4.1.18), ver Figura 8. Para el manejo de estados y persistencia de la información después de haber realizado la autenticación y en general de la Super App se hace uso de la librería Zustand (ver Sección 4.1.19), ver Figura 9.

```
// Se arma el objeto con data para la petición de Login al backend
const dataBackendRequest: TLoginBackendRequest = {
  documentType: documentType.TypeDocument || '',
  documentNumber: data.documentNumber,
  password: data.password,
  deviceInformation,
  ipAddress,
  userAgent,
};
const url = `${env.this_app.url_main_endpoint}/patient/login`;
const response = await fetch(url, {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
    Referer: bundleId,
  },
  body: JSON.stringify(dataBackendRequest),
});

const jsonData = await response.json();
```

Figura 8: Petición de servicio de autenticación usando API FETCH de Javascript.

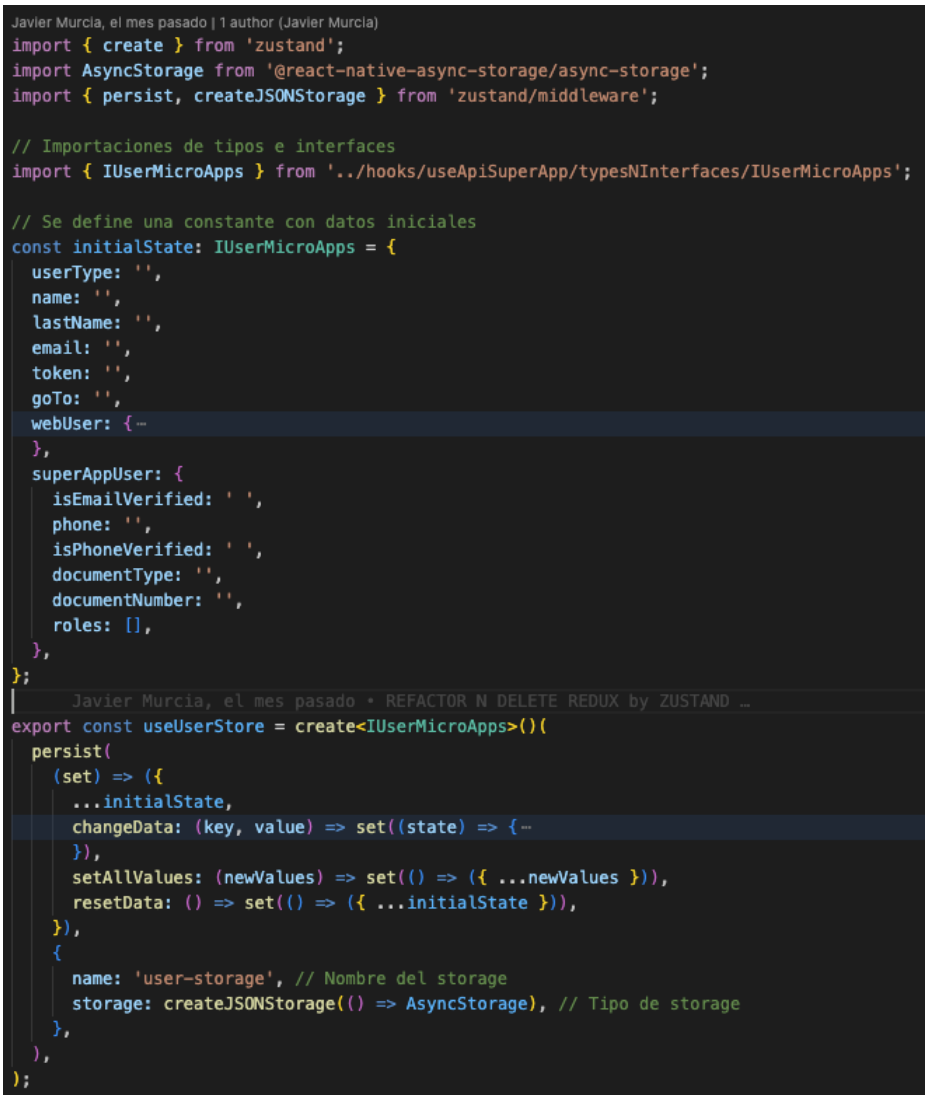


Figura 9: Manejo de estados y persistencia de datos de autenticación usando Zustand.

4. Para el soporte de múltiples idiomas, se implementó la librería i18next (ver la Sección 4.1.20). Como se muestra en la Figura 10, se detalla el proceso de inicialización para habilitar el soporte tanto en español como en inglés, tal como se describe en la Sección 5.1.2. La Figura 11 ilustra cómo, mediante la gestión de archivos .JSON, se facilita la localización de textos en varios idiomas. Por lo tanto, para incluir un nuevo idioma en el futuro, bastará con crear el correspondiente archivo "NAME_LANGUAGE.JSON" y añadirlo a la configuración para que esté disponible en toda la Super App.

app > localization > TS i18n.ts > [@] default

You, hace 9 minutos | 2 authors (Javier Murcia and others)

```
1  import i18n from 'i18next';
2  import { initReactI18next } from 'react-i18next';
3
4  import EN from './locales/EN.json';
5  import ES from './locales/ES.json';
6
7  i18n.use(initReactI18next).init({
8    compatibilityJSON: 'v3',
9    resources: {
10     EN: {
11       translation: EN,
12     },
13     ES: {
14       translation: ES,
15     },
16   },
17   fallbackLng: 'ES', // use 'es' if detected language is not available
18   interpolation: {
19     escapeValue: false, // react already safes from xss
20   },
21 });
22
23 export default i18n;      Javier Murcia, hace 5 meses • First commit fr
```

Figura 10: Configuración de i18next en la Super App.

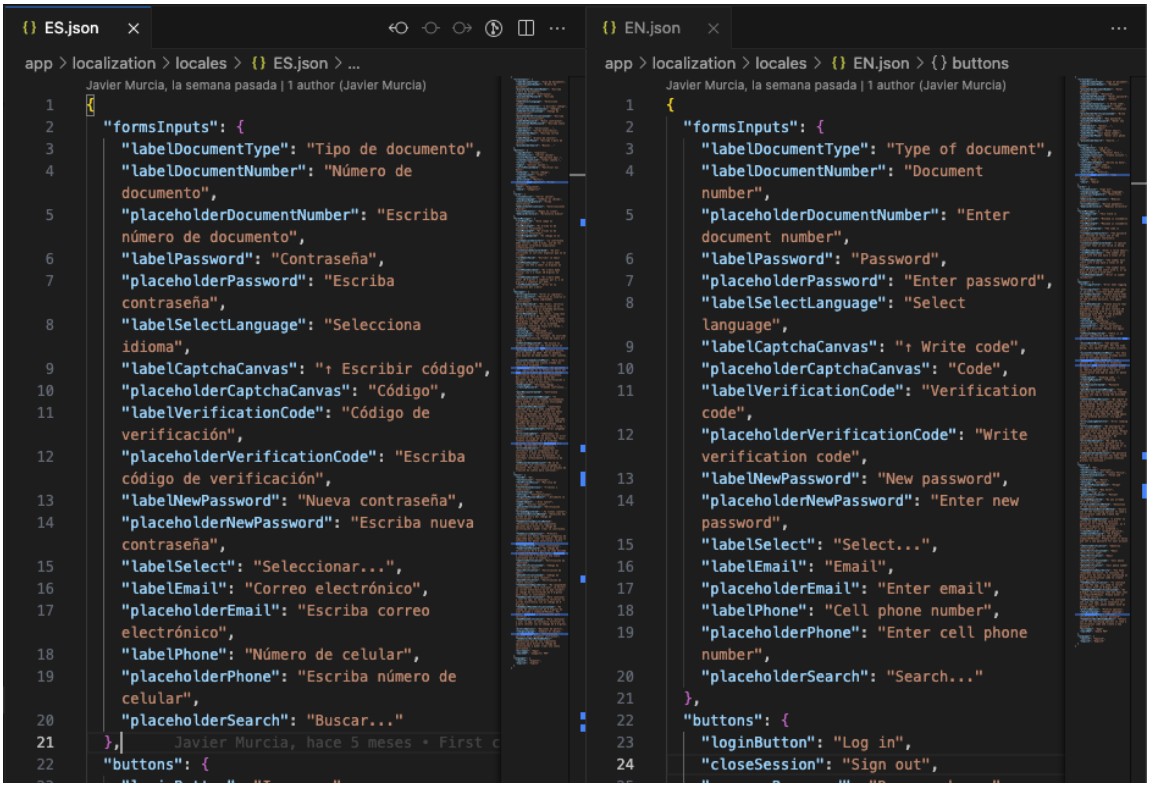


Figura 11: Archivos JSON contenido del idioma inglés y español de la Super App.

5. Conforme a la selección y decisión fundamentadas en la Sección 6.1, se construyó la aplicación móvil de la Super App utilizando React Native (ver definición en 6.1.2). Dentro de este desarrollo, se destacó la utilización de React Native Paper (ver la Sección 4.1.21), una librería clave por su oferta de componentes de interfaz gráfica que facilitaron un desarrollo más eficiente. Como resultado, se proporcionó a los usuarios de iOS y Android una experiencia de usuario coherente con las aplicaciones nativas de sus respectivos sistemas operativos. Las Figuras 12 y 13 ilustran la apariencia inicial de los componentes de selección (email o teléfono) antes y después de elegir una opción, mostrando las adaptaciones específicas de cada sistema operativo, todo ello logrado con una base de código unificada como se expone en la Figura 14.

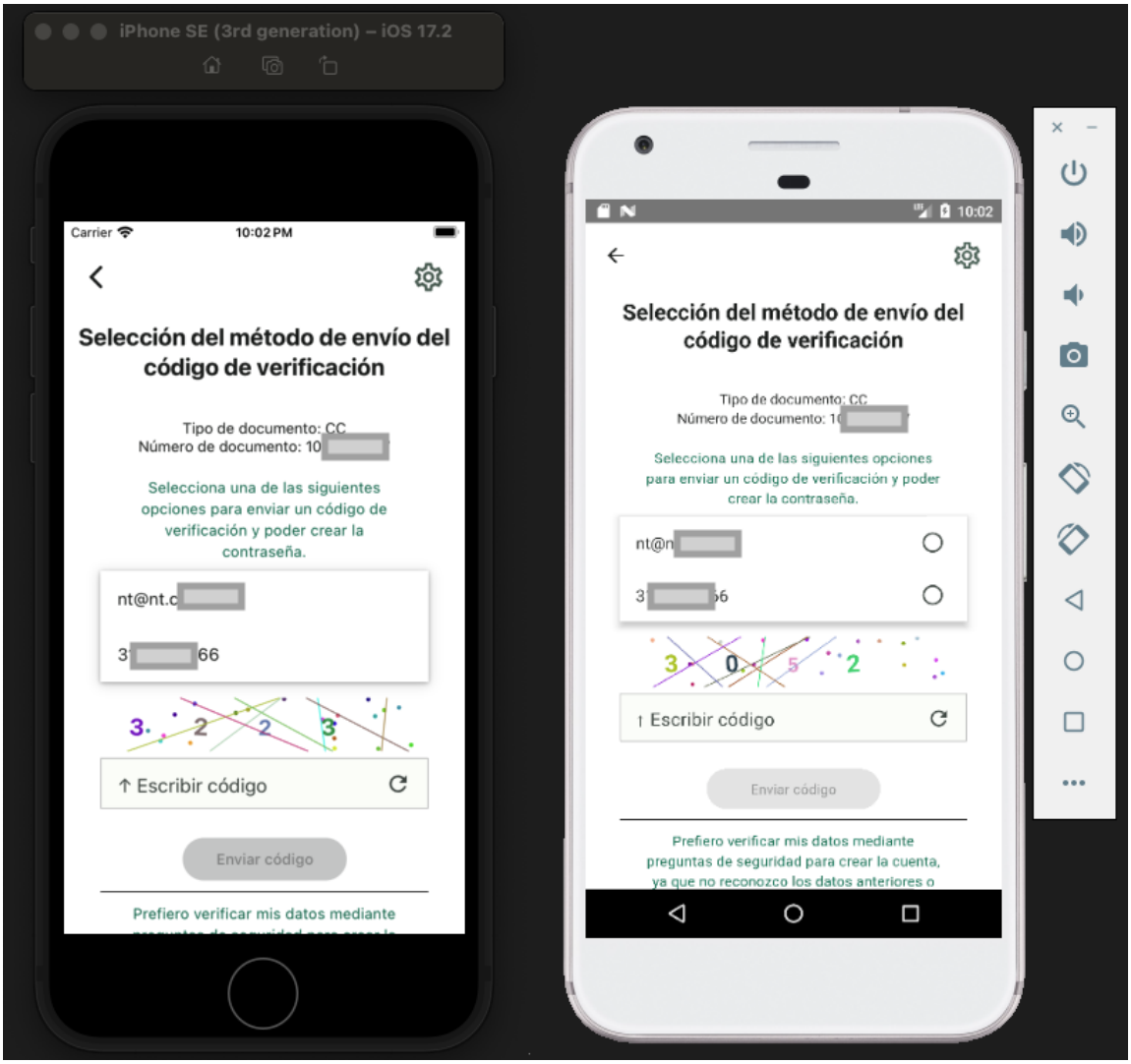


Figura 12: Opciones para seleccionar entre email o número celular, iOS y Android respectivamente

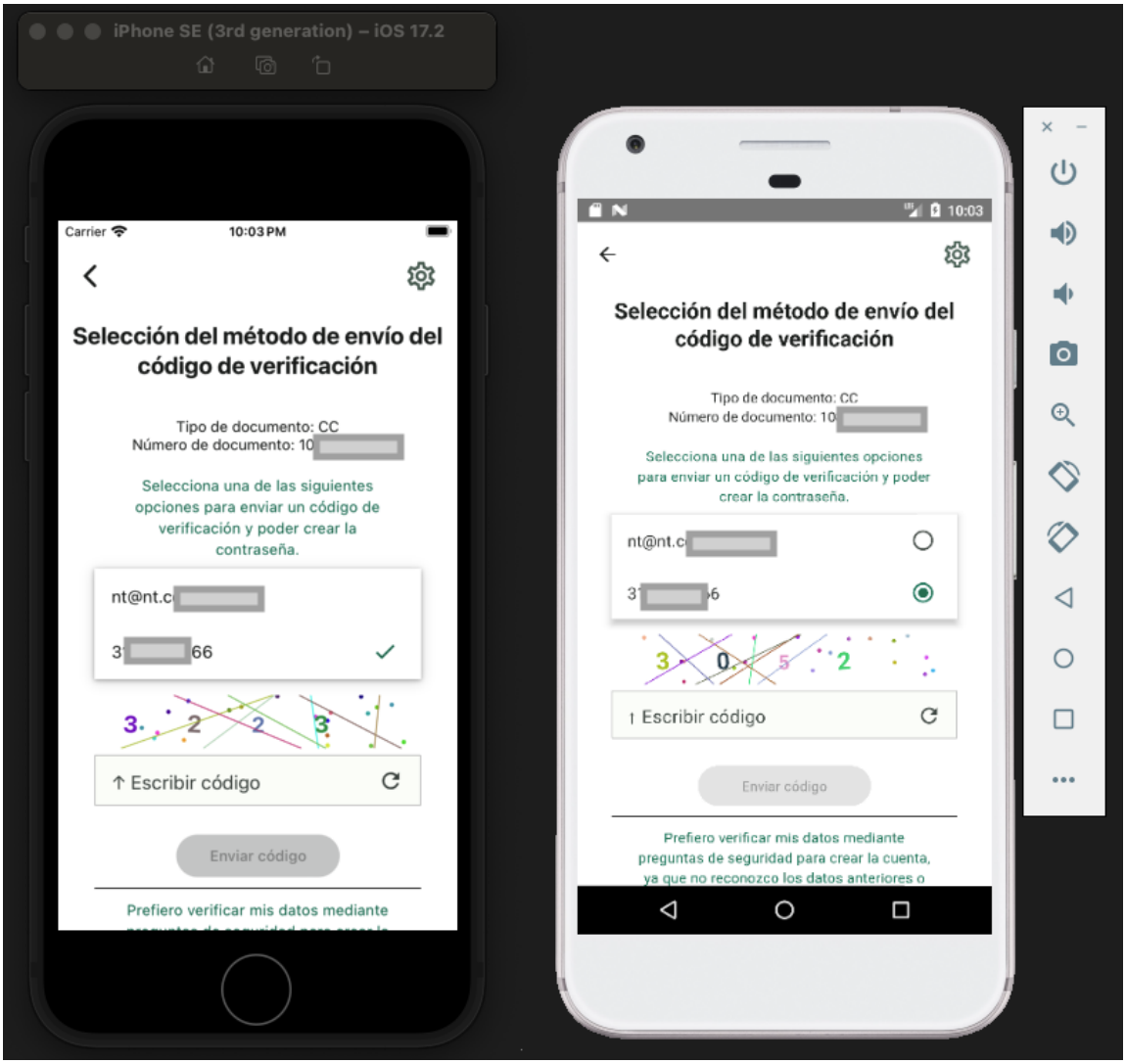


Figura 13: Numero celular seleccionado, iOS y Android respectivamente

```
<View style={{ ...globalStyles.viewContainerInput }}>
  <Surface
    elevation={2}
    style={{ backgroundColor: 'white', marginTop: 8 }}
  >      Javier Murcia, hace 3 meses • MODIFY AND REFACTOR …
    <RadioButton.Group
      onValueChange={(valueSelected) => onChange(valueSelected)}
      value={value}
    >
      {email && email !== '' && (
      <View testID="test-emailItem">
        <RadioButton.Item
          label={email}
          value="email"
        />
      </View>
      )}
      {phone && phone !== '' && (
        <View testID="test-phoneItem">
          <RadioButton.Item
            label={phone}
            value="phone"
          />
        </View>
      )}
    </RadioButton.Group>
    {error && error.message && (
    <HelperText type="error" visible={!!error}>
      {t(error.message)}
    </HelperText>
    )}
  </Surface>
</View>
```

Figura 14: Código único de componente de RadioButton, para selección de opción.

6.4.6. Actividades de desarrollo Super App: Módulo de menú para Apps / Micro Apps

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Especificación y desarrollo del menú para Apps / Micro Apps.

Tabla 9: Equipo responsable del desarrollo Super App: Módulo de menú para Apps / Micro Apps.

En un tiempo de 1 semana, se logró lo siguiente:

1. Siguiendo la estrategia establecida en la Sección 6.2, se seleccionó WebView para el desarrollo de la Super App, con el fin de permitir la integración dinámica y el desarrollo independiente de Apps / Micro Apps. Para ello, se utilizó la librería React Native WebView (ver Sección 4.1.22). Se procedió a crear y codificar un módulo reutilizable llamado ViewWebView en la Super App, diseñado para ser empleado a través de parámetros en todas las Apps / Micro Apps que se quieran integrar en la Super App. La Figura 15 ilustra cómo se implementó este módulo con la librería React Native WebView. Por otro lado, la Secciones 6.4.7 y 6.4.8, que se expondrán más adelante, expandirán la información sobre la utilización de este módulo en la Super App.

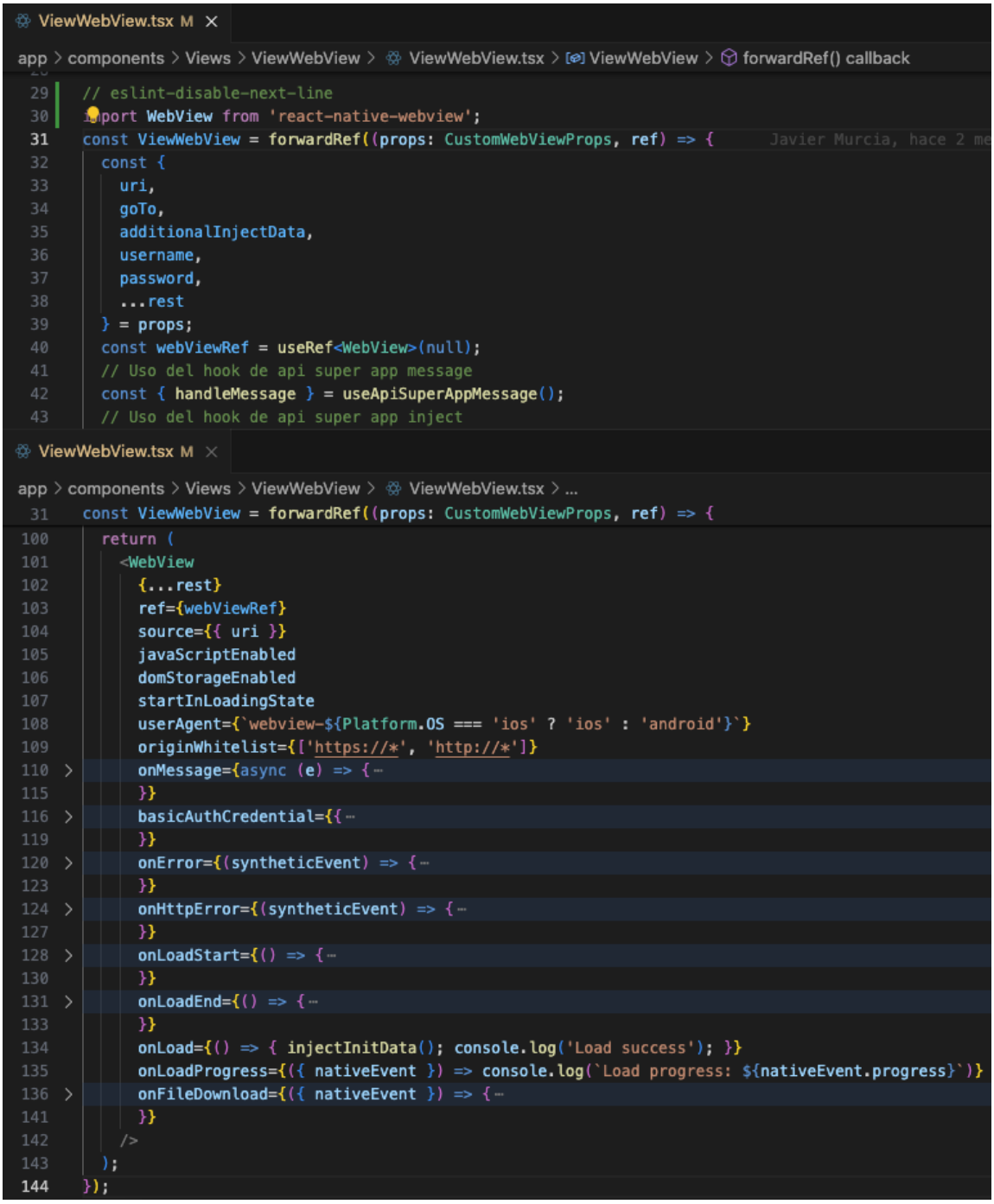


Figura 15: Implementación de librería React Native WebView en la Super App.

6.4.7. Actividades de desarrollo Micro App Firma electrónica: Identificación consumo y funcionalidad con Super App

Cargo	Actividades
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Especificación y desarrollo de Micro App Firma Electrónica.
Analista de desarrollo	- Desarrollo de Micro App Firma Electrónica.

Tabla 10: Equipo responsable del desarrollo de Micro App Firma Electrónica: Identificación consumo y funcionalidad con Super App Móvil

En un tiempo de 6 semanas, se logró lo siguiente:

1. Se desarrollan las interfaces y funcionalidades del demo Micro App de Firma Electrónica , teniendo en cuenta lo especificado en la Sección 5.2.

2. Un factor determinante para la elección de esta primera Micro App fue que se encontraba en una etapa avanzada de desarrollo; es decir, ya contaba con código preexistente y funcionalidades operativas. Sin embargo, al intentar integrarla como la primera Micro App de la Super App y llevar a cabo pruebas iniciales, se detectaron problemas de carga en su interfaz gráfica y funcionalidades al utilizar el módulo ViewWebView. El diagnóstico, realizado mediante depuración, reveló que la incompatibilidad se presentaba al ejecutar la aplicación en un emulador con Android 7.0 (API LEVEL 24 - Nougat). En concordancia con el desarrollo, y tras una reunión con el coordinador, se estableció que la Super App debe operar a partir de esta versión de Android, con el objetivo de cubrir como mínimo el 97 % de los dispositivos actuales, cifra respaldada por la Figura 32. Dada la diversidad socioeconómica y de edades de los pacientes de la institución de salud, se adopta esta cobertura como estándar mínimo.
   Ya teniendo claro la mínima versión a soportar se encontró que en la documentación de Android 7 (Android Open Source Project (2023)) maneja por defecto como navegador predeterminado Chromium, y al entrar a revisar la

Version	SDK / API level	Version code	Codename	Cumulative usage [1]	Year [4]
Android 15 DEV	Level 35	VANILLA_ICE_CREAM	Vanilla Ice Cream [2]	—	*TBD*
Android 14	Level 34	UPSIDE_DOWN_CAKE	Upside Down Cake [2]	12.6%	2023
	▪ `targetSdk` will need to be 34+ for new apps and app updates by August 31, 2024.				
Android 13	Level 33	TIRAMISU	Tiramisu [2]	41.4%	2022
	▪ `targetSdk` must be 33+ for new apps and app updates since August 31, 2023.				
Android 12	Level 32 Android 12L	S_V2	Snow Cone [2]	58.5%	
	Level 31 Android 12	S			2021
Android 11	Level 30	R	Red Velvet Cake [2]	75.0%	2020
Android 10	Level 29	Q	Quince Tart [2]	84.0%	2019
Android 9	Level 28	P	Pie	90.0%	2018
Android 8	Level 27 Android 8.1	O_MR1	Oreo	92.1%	2017
	Level 26 Android 8.0	O		95.1%	
Android 7	Level 25 Android 7.1	N_MR1	Nougat	95.6%	2016
	Level 24 Android 7.0	N		97.0%	
Android 6	Level 23	M	Marshmallow	98.4%	2015
Android 5	Level 22 Android 5.1	LOLLIPOP_MR1	Lollipop	99.2%	
	Level 21 Android 5.0	LOLLIPOP, L		99.5%	2014
	▪ Jetpack Compose requires a `minSdk` of 21 or higher. ▪ Google Play services v23.30.99+ (August 2023) drops support for API levels below 21.				

Figura 16: Versiones de android, niveles API y porcentaje acumulativo de usuarios. API Levels Contributors (2024)

librería principal gráfica de componentes que se estaba manejando en la Micro App de Firma Electrónica era Material UI, la cual en su página establece que solo soporta los navegadores descritos en la Figura 17 y para un correcto funcionamientos a partir de las versiones ahí descritas, en este punto ya se encuentra una incompatibilidad, teniendo en cuenta esto se evalúan dos opciones una seria realizar un framework propio con componentes gráficos realizados por el área de desarrollo, donde se use estilos y funciones de javascript de versiones anteriores que sea soportadas por todos los navegadores o investigando se encuentra que Bootstrap (ver Sección 4.1.23), en su actual versión soporta navegadores de dispositivos móviles desde la versión de Android 6, como se puede ver en la Figura 18, por lo cual se toma la decisión de refactorizar el código y hacer cambio a Bootstrap con el fin de cumplir el soporte a dispositivos con versiones más antiguas y no tener que tomar como área del desarrollo, soporte y evolución de un framework propio.

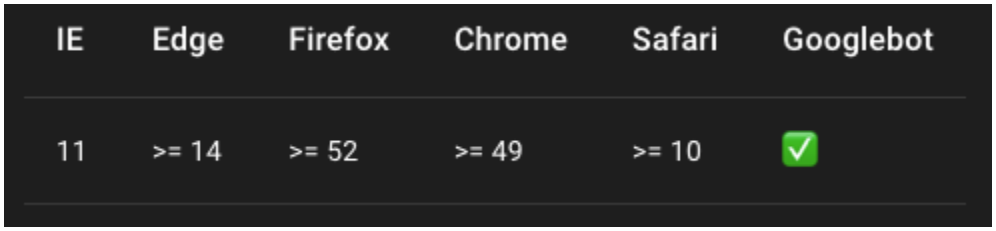


Figura 17: Navegadores soportados por Material UI. Material UI Team (2024)

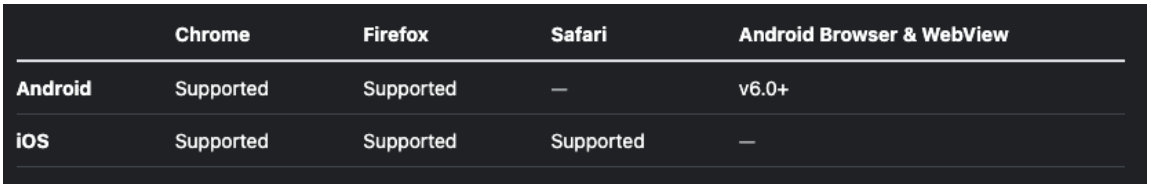


Figura 18: Navegadores soportados por Bootstrap. Bootstrap Team (2024b)

Ya resuelto esto, al final se obtiene una Micro App de Firma Electrónica funcional, tanto para los casos WEB cuando se envía URL por Correo electrónico o SMS para realizar la firma como se muestra en la Figura 19 y el caso de firma del documento desde la Super App como se muestra en la Figura 20.

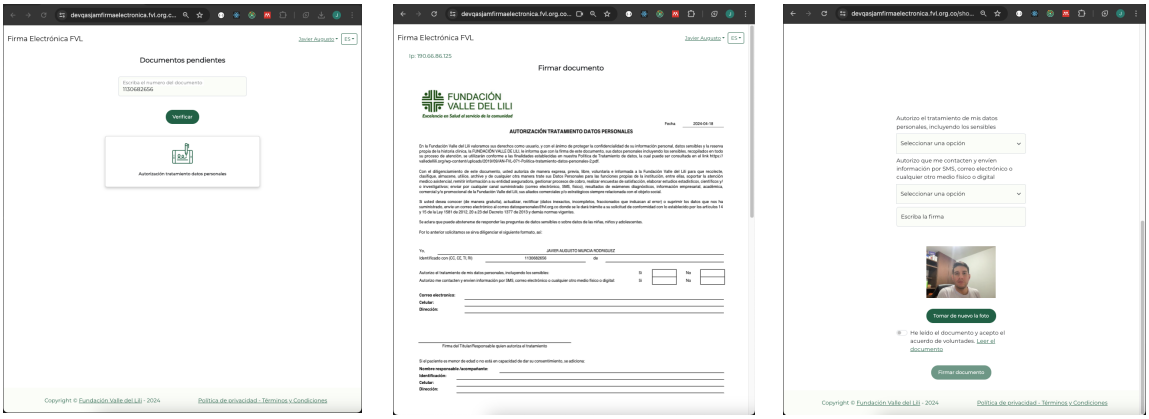


Figura 19: Micro App Firma Electrónica funcionando en WEB.

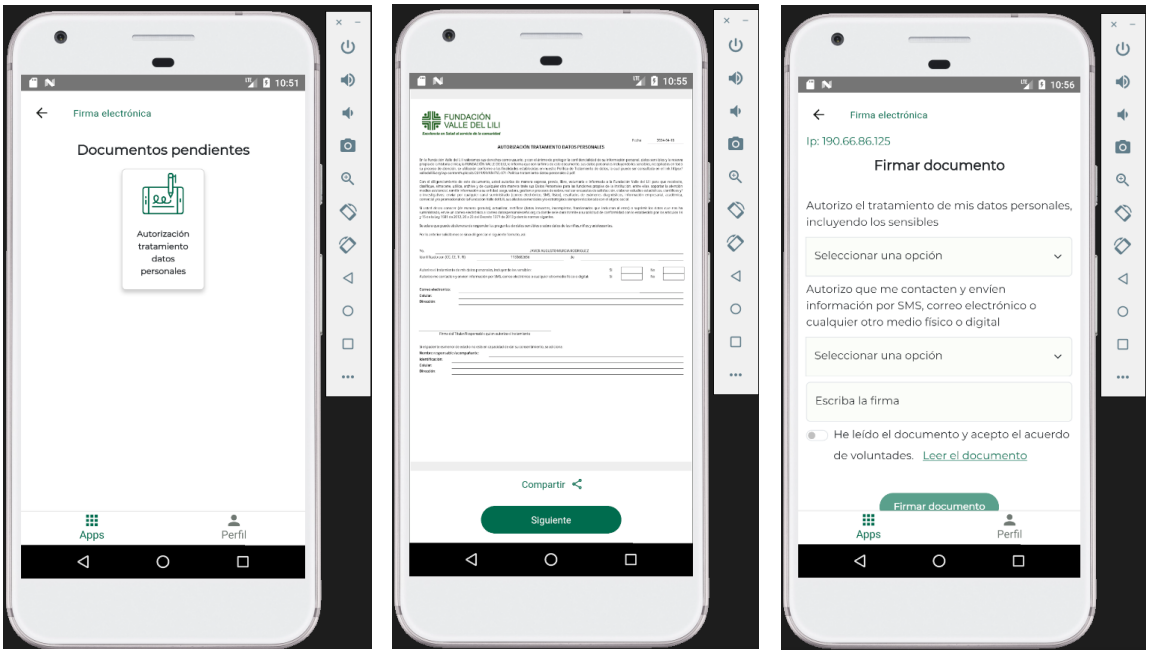


Figura 20: Micro App Firma Electrónica funcionando en Super App.

3. La Micro App de Firma Electrónica detecta su consumo por la Super App a través de una interfaz de ventana (ver Sección4.1.24). De acuerdo con la guía de React Native WebView (Contributors (2024b)), al cargar la Micro App, la librería inyecta un objeto window.ReactNativeWebView que provee funcionalidades para la comunicación entre la Micro App y la Super App. Este método no solo facilita la comunicación e integración, como se detalla en la Sección 6.4.8, sino que también permite ejecutar validaciones y ajustar la interfaz gráfica. La Figura 21 línea 19, se muestra cómo se realiza la validación si es desde la Super App el consumo y se realiza adaptación de la interfaz gráfica, específicamente en el ancho del componente de MicroAppCard para su correcta visualización tanto en la Web como en la Super App, el resultado final se representado en la Figura 22.

src > components > Cards > MicroAppCard > MicroAppCard.tsx > MicroAppCard

```tsx
const MicroAppCard = ({ title, imageSource, onPress }: Props) => {
  const [cardWidth, setCardWidth] = useState('33%');
  const [cardMarginTop, setCardMarginTop] = useState(5);

  useEffect(() => {
    // Se detecta si consumo es desde Super App
    if (window.ReactNativeWebView) {
      const updateWidth = () => {
        const windowWidth = window.innerWidth;
        const MAX_CARD_WIDTH = 120;
        const calculatedWidth = Math.min(windowWidth / 3 - 10, MAX_CARD_WIDTH);
        setCardWidth(`${calculatedWidth}px`);
      };
      window.addEventListener('resize', updateWidth);
      updateWidth();
      return () => window.removeEventListener('resize', updateWidth);
    }

    setCardMarginTop(20);
    return setCardWidth('40vh');
  }, []);
      j@vier Murcia, hace 2 meses • ADD src/pages/ShowAllDocuments/ …
  return (
    <div style={{ width: cardWidth, margin: '3px', marginBottom: '20px' }}>
      <Card onClick={() => onPress()} className="card-style">
        <Card.Img
          variant="top"
          src={imageSource}
          style={{ marginTop: cardMarginTop, height: 60, objectFit: 'contain' }}
        />
        <Card.Body>
          <Card.Text
            style={{
              textAlign: 'center',
              marginTop: '5px',
              fontSize: 12,
              fontWeight: 600,
            }}
          >
            {title}
          </Card.Text>
        </Card.Body>
      </Card>
    </div>
```

Figura 21: Cálculo de ancho de componente Card de la Micro App.

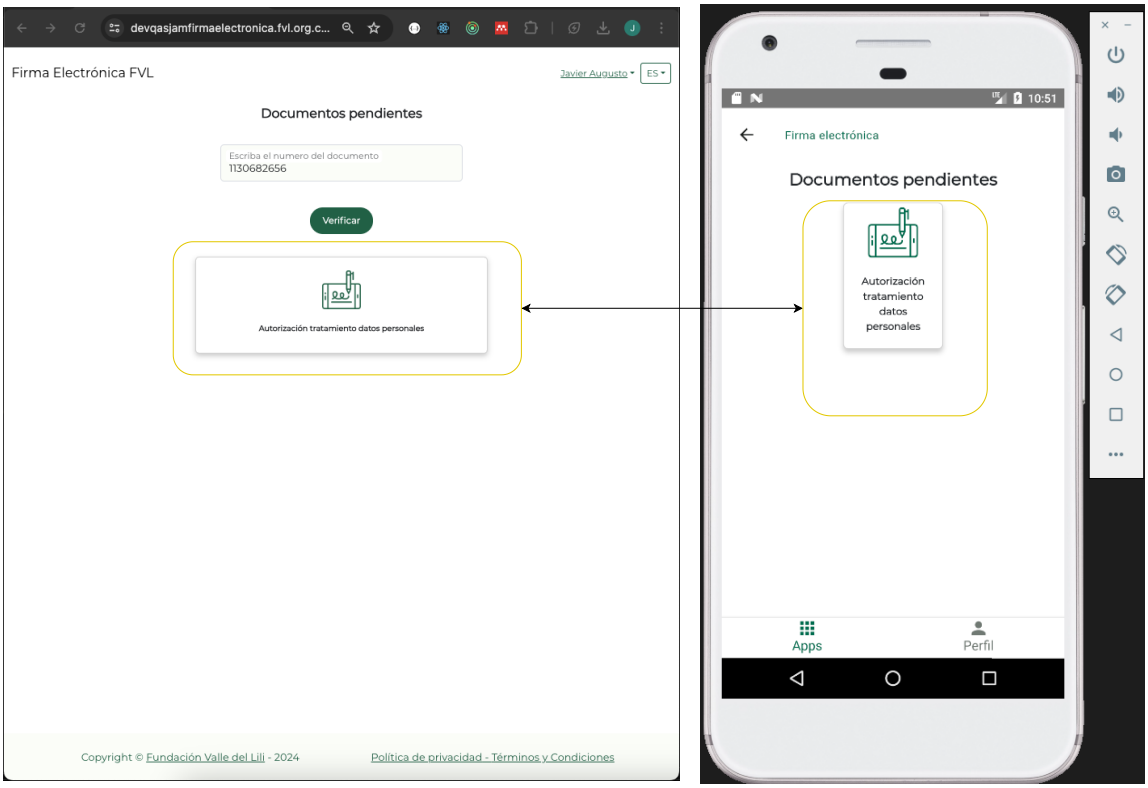


Figura 22: Cálculo de ancho de componente Card de la Micro App, resultado.

6.4.8. Desarrollo e integración de Micro App de Firma Electrónica y Super App

Cargo	Actividades
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Desarrollar componentes y módulos para la integración entre Micro App de Firma Electrónica y Super App

Tabla 11: Equipo responsable del desarrollo e integración de Micro App de Firma Electrónica y Super App

En un tiempo de 3 semanas, se logró lo siguiente:

1. Se realiza una integración de manera correcta entra la Micro App de Firma Electrónica y la Super App.

2. Se desarrolló una API (ver Sección 4.1.25) que incluye cuatro Hooks (ver Sección 4.1.26), dos diseñados para su uso en la Super App y los dos restantes para su implementación en las Apps / Micro Apps a integrar. Estos Hooks se basan en los métodos proporcionadas por React Native WebView (Contributors (2024b)) para facilitar la comunicación entre la interfaz nativa y la web integrada. La documentación y análisis realizados sobre React Native WebView indican que existen dos métodos principales para esta comunicación:

   - **Método por inyección de código (injectJavaScript):** Este enfoque permite la inserción de código JavaScript directamente desde la Super App hacia las Apps/Micro Apps. Se han creado dos Hooks específicos para este propósito llamado useApiWebInjectByEvent. El primero está diseñado para la inicialización y el uso en la Super App, mientras que el segundo está pensado para las Apps / Micro Apps. En el lado de la Super App, el proceso implica la inyección de código que corresponde a diversos eventos dentro de las Apps / Micro Apps. El segundo Hook establece oyentes para eventos denominados superAppEvent, que, basados en los datos y el

tipo de evento del mensaje, desencadenan acciones correspondientes. Esta mecánica se ilustra en la Figura 23. Actualmente, se han definido solo dos

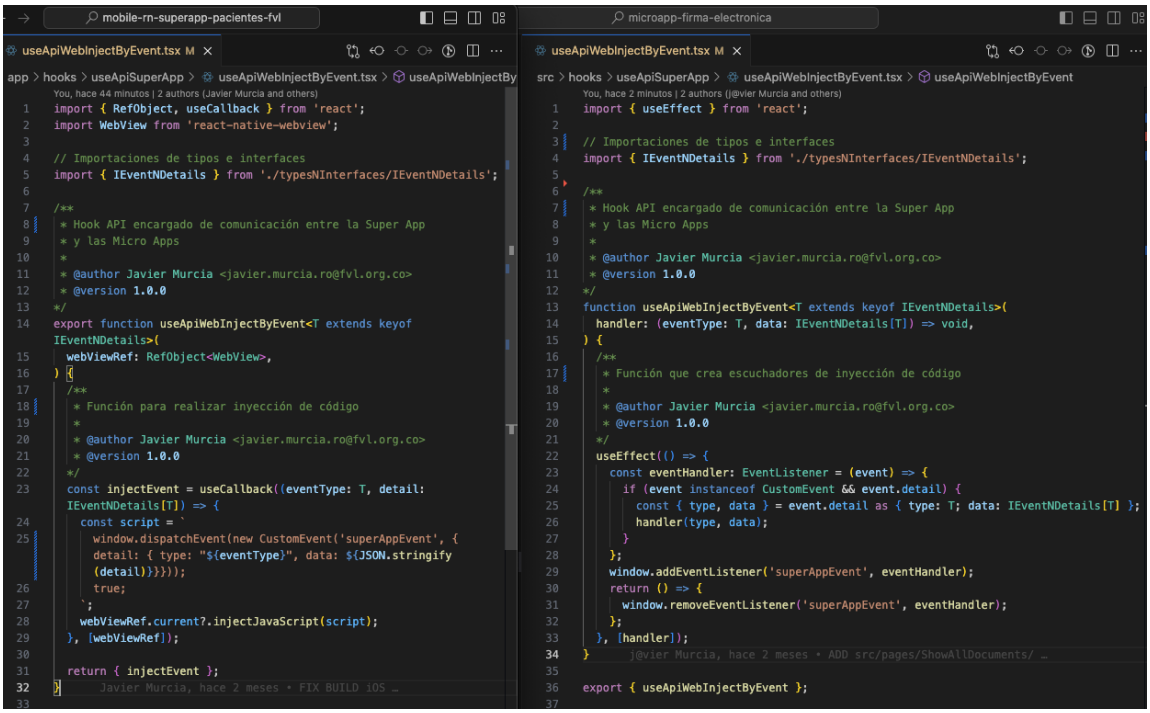


Figura 23: Inicialización y uso de hooks useApiWebInjectByEvent.

eventos, visibles en la Figura 24, que son cruciales para la inserción de datos del usuario y la configuración del idioma, asegurando la coherencia de la información. Es esencial que las interfaces declaradas para estos eventos sean idénticas en ambos entornos. La Figura 25 presenta un ejemplo práctico de cómo la Super App inyecta la configuración de idioma del usuario en la Micro App de Firma Electrónica. A través del Hook y sus oyentes, la Micro App está atenta a un evento setLanguage, procesándolo para aplicar el valor recibido y mantener así la consistencia idiomática en la interfaz, manteniendo la experiencia del usuario.

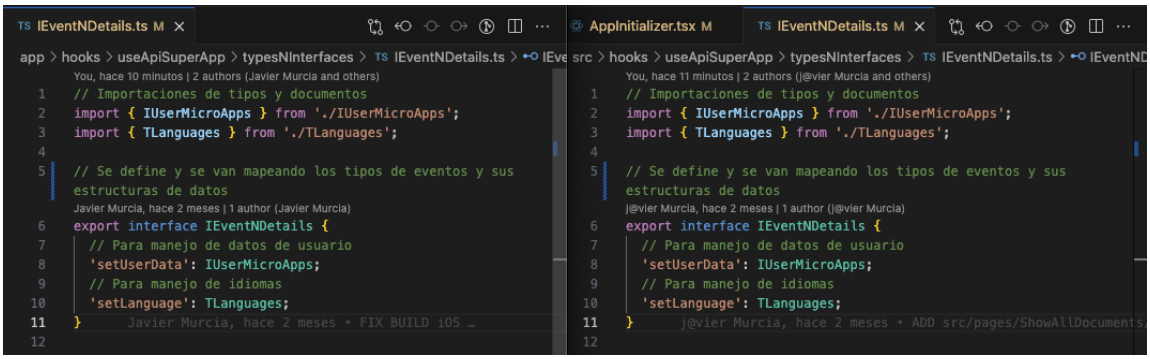


Figura 24: Interfaces para el uso de hooks useApiWebInjectByEvent.

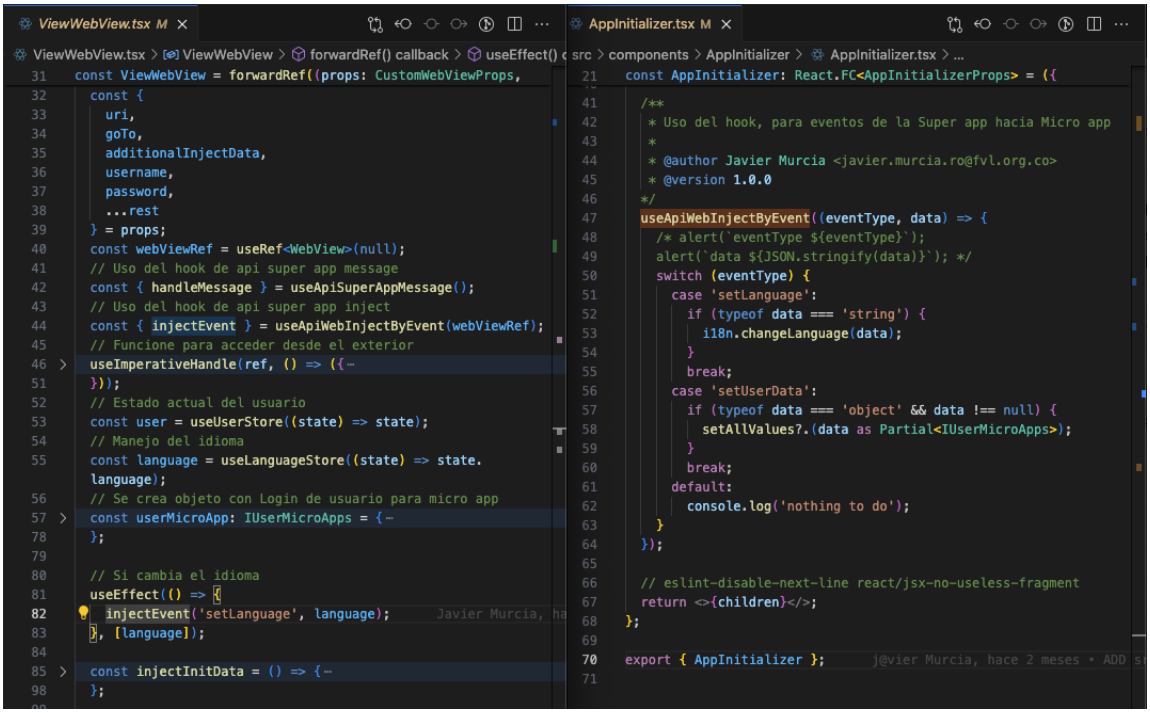


Figura 25: Implementación de hooks useApiWebInjectByEvent.

- **Método por mensajería (postMessage)** Este segundo método facilita la comunicación entre las Apps / Micro Apps y la Super App. Siguiendo un proceso similar al anterior, se desarrollaron dos Hooks llamados useApiSuperAppMessage. En este caso, la comunicación se realiza en sentido inverso: las Apps / Micro Apps primero verifican si están siendo consumidas por una Super App y luego envían información a través del método postMessage. Por su parte, la Super App recibe estos mensajes utilizando un gestor de mensajes, como se ilustra en la Figura 26.

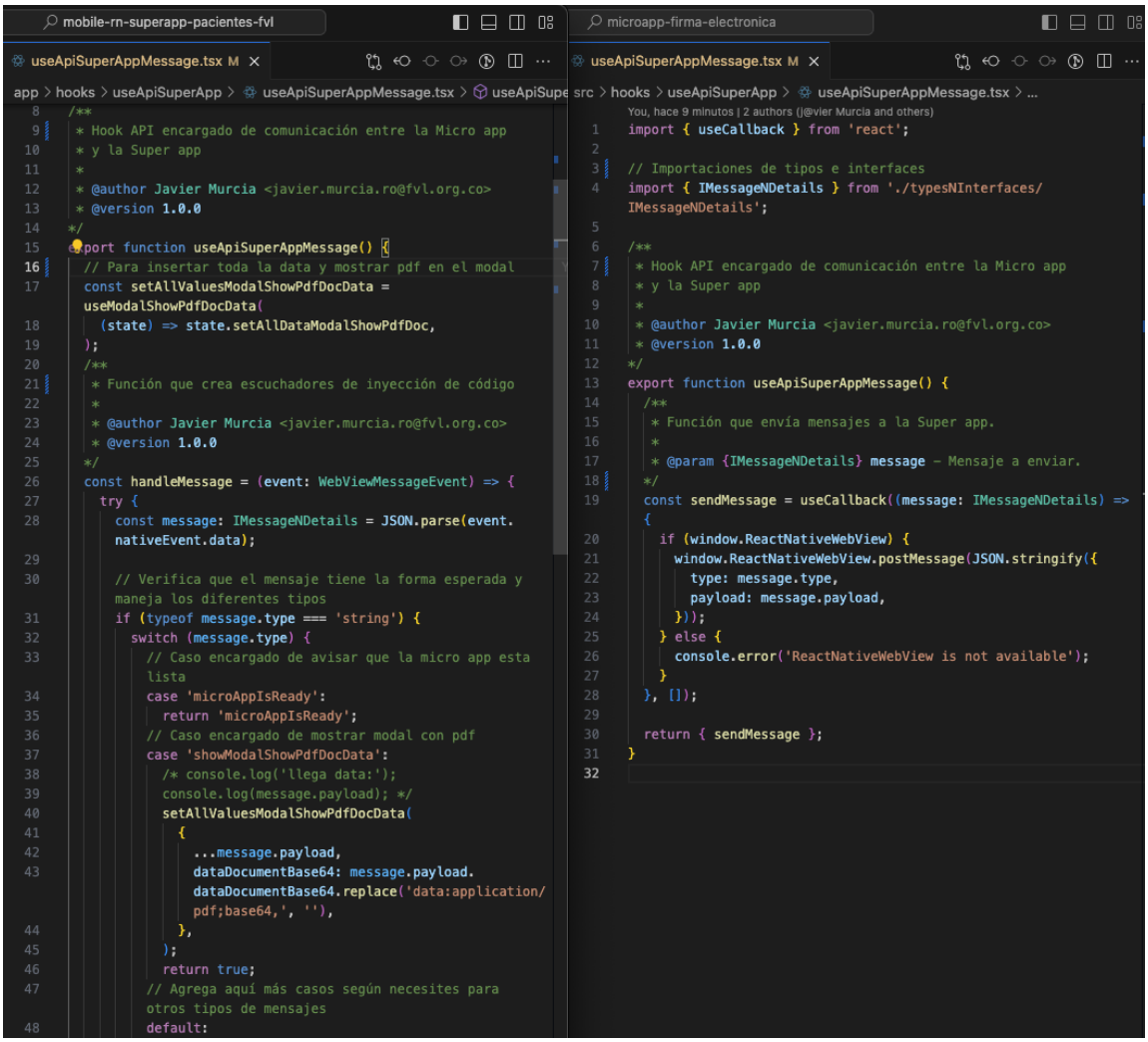


Figura 26: Inicialización y uso de hooks useApiSuperAppMessage.

Para mantener la coherencia en la comunicación, los tipos de mensajes se definen a través de interfaces que deben ser consistentes en ambos extremos, detalle que se puede apreciar en la Figura 27 con mensajes de tipo microAppIsReady y showModalShowPdfDocData. Finalmente, la

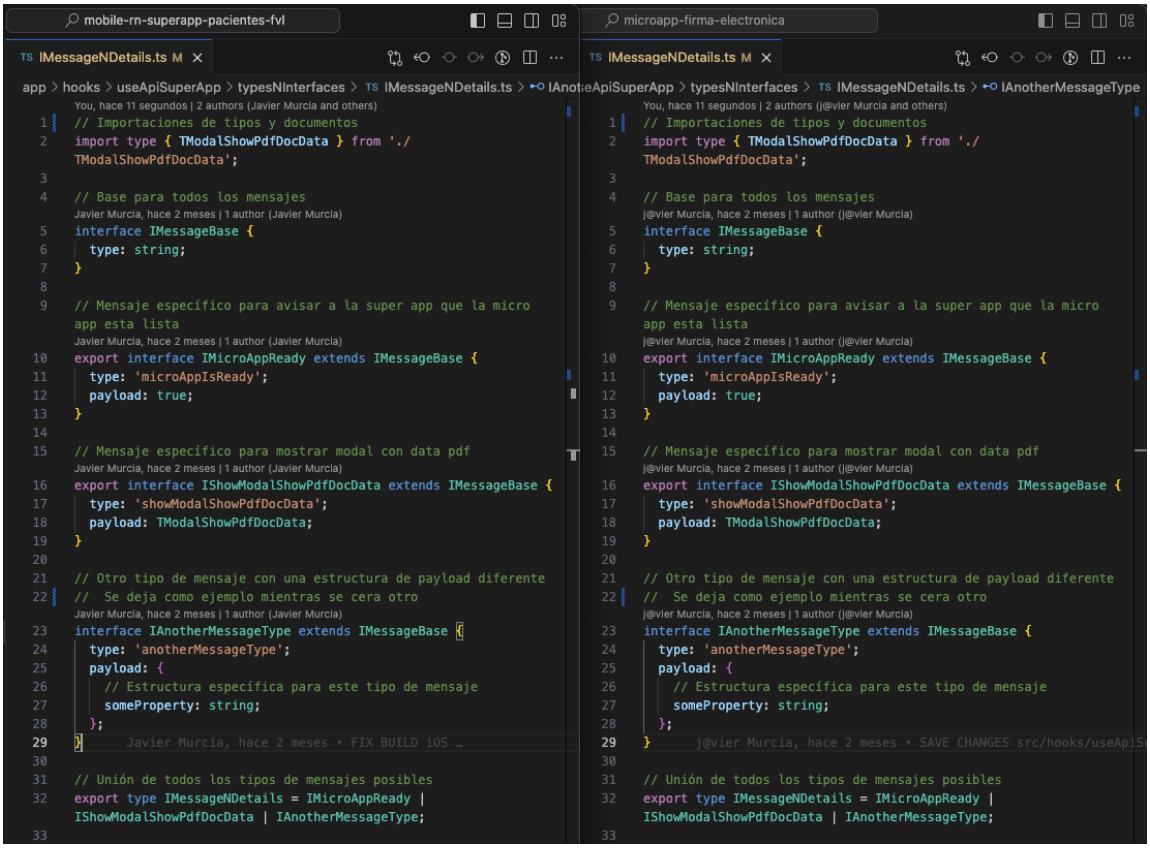


Figura 27: Interfaces para el uso de hooks useApiSuperAppMessage.

Figura 28 presenta un ejemplo de cómo se utilizan estos Hooks en la Micro App de Firma Electrónica para verificar si la interacción proviene de la Super App y, de ser así, proceder al envío de datos necesarios para visualizar un documento en formato PDF de manera nativa dentro de la Super App.

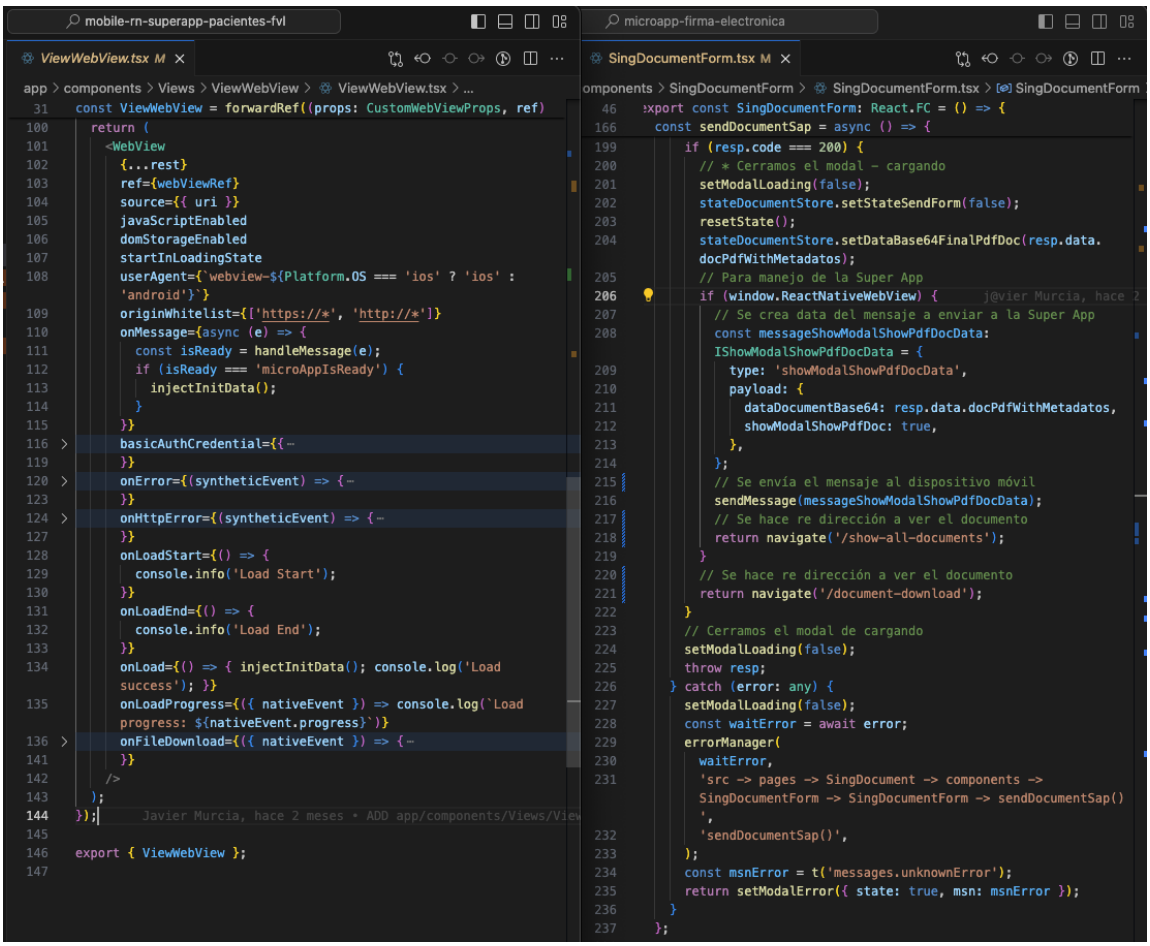


Figura 28: Implementación de hooks useApiSuperAppMessage.

3. Con el propósito de estandarizar y asegurar el uso adecuado de la API para la correcta integración entre la Super App y las Micro Apps, se ha elaborado un documento guía titulado “Integración con Super App Pacientes FVL”. Este documento también establece reglas claras con buenas prácticas y requisitos mínimos para la integración. Se encuentra disponible para consulta en el Anexo A.6.

6.5. Desarrollos basados en el ASR de modularidad

Continuando con el enfoque de desarrollo previamente establecido, se describe una modificación aplicada al Módulo de menú para micro apps. Esta actualización implementa estilos y tácticas arquitectónicas diseñados para potenciar la modularidad en la incorporación y manejo eficiente de nuevas Apps / Micro Apps en la Super App. Para obtener mayor detalle, se puede consultar el Anexo A.4.

6.5.1. Modificación de desarrollo Super App: Módulo de menú para Apps / Micro Apps

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Implementación del desarrollo del menú para Apps / Micro Apps, enfocado al ASR de modularidad.

Tabla 12: Equipo responsable para la modificación de desarrollo Super App: Módulo de menú para Apps / Micro Apps

En un tiempo de 1 semana, se logró lo siguiente:

1. Se realiza modificación al módulo agregando un nuevo componente funcional llamado MicroAppsNavigator usando librerías de react native, con el fin de modularizar la integración y manejo de Apps / Micro Apps en la Super App.

2. Al realizar modificación se realizaron las respectivas validaciones para ver su correcto funcionamiento.

3. Se toma la decisión de implementarlo en la rama master (principal) de la Super App.

6.6. Desarrollos basados en el ASR de seguridad

El último atributo de calidad considerado para este proyecto es la seguridad. Por ello, se realizan pruebas de concepto antes de proceder con la implementación completa en la solución de la Super App. Dada la importancia que la institución otorga a la protección de los datos de los pacientes, es esencial obtener la revisión y aprobación del área de seguridad de la institución. Una vez aprobadas, se procederá con la implementación de las modificaciones detalladas en el Anexo A.5.

6.6.1. Codificación: Prueba de concepto de Encriptación RSA-OAEP

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Implementación del desarrollo de prueba de concepto de Encriptación RSA-OAEP.

Tabla 13: Equipo responsable para la prueba de concepto de Encriptación RSA-OAEP

En un tiempo de 1 semana, se logró lo siguiente:

1. Se realiza investigación y documentación de propuesta para encriptación de datos.
2. Se desarrolló una función junto con un código de ejemplo para la generación de llaves públicas y privadas RSA.
3. Se realiza función a utilizar y código de ejemplo para encriptación de datos.
4. Se realiza función a utilizar y código de ejemplo para desencriptación de datos.

5. Se realiza una prueba verificando el funcionamiento de todo lo realizado, además de una prueba modificando llave privada para verificar el correcto funcionamiento.

7. Pruebas

Para el desarrollo de este proyecto, las pruebas de software son fundamentales dado que la institución opera bajo estrictas políticas de calidad que han establecido diversos procesos y modelos orientados hacia el mejoramiento continuo y la excelencia. Esto ha permitido a la institución obtener acreditaciones de estándares nacionales e internacionales, Fundación Valle del Lili (2024a), y ser reconocida como el cuarto mejor hospital de Latinoamérica y el primero en Colombia, según Fundación Valle del Lili (2024b).

Desde el departamento de informática se asume una gran responsabilidad ya que el software será utilizado directamente por los pacientes y podría afectar directamente la reputación de la institución. En este contexto, las pruebas de software, según IBM (2024a), que son el proceso de evaluación y verificación de un producto o aplicación de software para saber si hace lo que se supone que debe hacer, además, si se va a algo menos técnico y más práctico se puede afirmar que las pruebas son necesarias porque con ellas se puede ayudar a reducir los riesgos en las aplicaciones, y lograr de esta manera que se identifiquen los defectos antes de que se ejecuten, con esto se previenen los fallos, Softesting (2019) y justamente eso es lo que se busca desde el área de informática por lo cual se realizaron las siguientes acciones con ayuda del equipo de analistas de calidad de software con los que cuenta el área.

A continuación, se detallan los tipos de pruebas que se han realizado hasta el momento a la aplicación

- Pruebas de unidad o unitarias.
- Pruebas de integración.

7.1. Pruebas unitarias

Teniendo en cuenta la definición de pruebas unitarias establecida en el marco teórico de la Sección 4.1.27, la institución aún no dispone de un formato o proceso establecido para realizar este tipo de pruebas, por ello, se aprovecha este proyecto para proponer un enfoque sistemático para las pruebas unitarias, enfatizando su

importancia basándose en las ventajas señaladas por Amazon Web Services (2024). Algunas de estas ventajas incluyen:

- **Detección eficaz de errores:** Si hay errores de entrada, salida o basados en la lógica dentro de un bloque de código, las pruebas unitarias ayudan a detectarlos antes de que los errores lleguen a producción. Cuando cambia el código, se ejecuta el mismo conjunto de pruebas unitarias, junto con otras pruebas como las de integración, y se esperan los mismos resultados. Si las pruebas fallan (o como se le suelen llamar pruebas rotas), indica errores basados en la regresión. Las pruebas unitarias también ayudan a encontrar errores más rápido en el código. Los desarrolladores no dedican mucho tiempo a las actividades de depuración. Pueden identificar con rapidez la parte exacta del código que tiene un error.

- **Documentación:** Es importante documentar el código para saber exactamente lo que se supone que debe hacer ese código. Dicho esto, las pruebas unitarias también actúan como una forma de documentación. Otros desarrolladores leen las pruebas para ver qué comportamientos se espera que muestre el código cuando se ejecute. Utilizan la información para modificar o refactorizar el código. Refactorizar el código hace que sea más eficaz y esté mejor compuesto. Puede volver a ejecutar las pruebas unitarias para verificar que el código funciona según lo esperado después de los cambios.

En el Anexo A.7 se encuentra el formato inicialmente propuesto para el desarrollo de las pruebas unitarias. Este formato fue revisado y recibido con muy buen agrado por los analistas de calidad de software del área, lo que indica un paso positivo hacia la estandarización de estas prácticas dentro del área de desarrollo de software de la institución.

7.1.1. Desarrollo de pruebas unitarias

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Creación del documento para documentar las pruebas unitarias. - Implementación del desarrollo de pruebas unitarias.
Analista de calidad de software	- Definición y priorización de pruebas teniendo en cuenta metodologías del área.

Tabla 14: Equipo responsable para desarrollo de pruebas unitarias

En un tiempo de 3 semanas, se logró lo siguiente:

1. Se creó un documento para documentar las pruebas unitarias, que servirá como guía estándar para futuras pruebas unitarias del software.

2. Se entrega el documento con las pruebas definidas y priorizadas para la Super App.

3. Se ha seleccionado Jest (Sección 4.1.29) como la herramienta para realizar pruebas unitarias. Este marco de pruebas es recomendado por el equipo de React Native y es ampliamente utilizado por el equipo de Facebook (META), como se menciona en su página oficial y se puede ver en la Figura 29. Además, según NPM, Jest Core Team (2024), Jest cuenta con más de 16 millones de descargas semanales, lo que refleja una amplia adopción y respaldo de una gran comunidad activa.

4. Se realizo la instalación y configuración de Jest en la Super App.

5. Se completó la codificación de todas las pruebas unitarias que fueron definidas y priorizadas. En la Sección 7.1.2, se proporciona un ejemplo detallado de cómo se realiza la codificación de una de estas pruebas utilizando Jest.

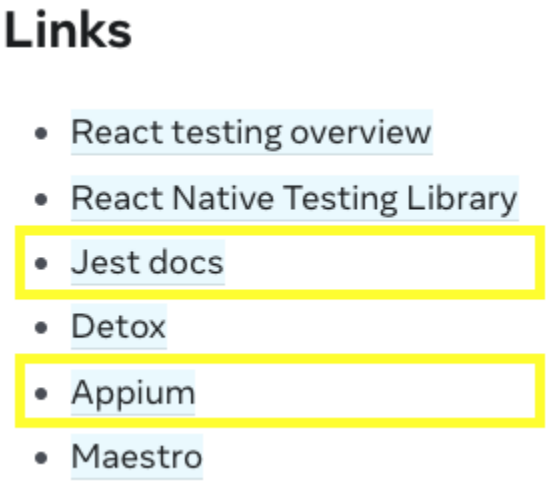


Figura 29: Recomendaciones para pruebas según React Native Documentation Contributors (2024).

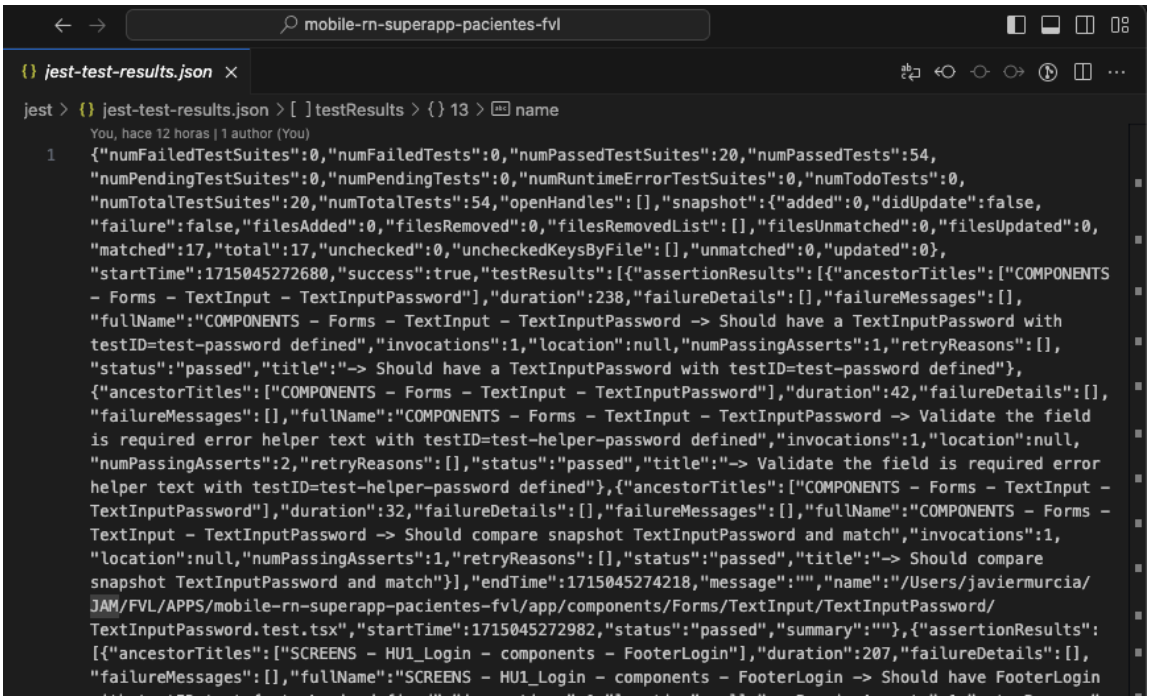


Figura 30: Ejemplo resultado de pruebas Jest con comando –json.

6. Con la automatización de las pruebas unitarias usando Jest, los analistas de calidad de software solicitaron la posibilidad de generar reportes para mantener una trazabilidad de las pruebas realizadas y los resultados obtenidos. Al revisar la documentación de Jest, específicamente en Jest Documentation Contributors (2024), se identificó que el comando –json permite exportar los resultados de las pruebas en formato JSON, dirigiendo toda la salida y los mensajes a stderr. Esto facilita la integración con otras herramientas o el análisis de datos en sistemas automatizados.

   Se realizó una prueba rápida para evaluar la salida en formato JSON de las pruebas, resultando en un documento JSON que resultó ser difícil de leer y poco funcional, como se muestra en la Figura 30. Dado que este archivo no ofrece una visualización amigable de la información, se decidió desarrollar una utilidad que permitiera procesar los resultados del JSON y generar un archivo en formato .xlsx (Excel). El código de esta utilidad, escrita en JavaScript y denominada convertTestResults.js, se muestra en la Figura 31. Además, en el Anexo A.8 se encuentra el resultado en Excel de las pruebas unitarias realizadas, facilitando así una mejor gestión y visualización de la información.

Figura 31: convertTestResults.js.

7.1.2. Ejemplo de prueba unitaria, TextInputPassword

A continuación, se presenta un ejemplo de un componente desarrollado a partir de un TextInput, denominado TextInputPassword. Este componente está específicamente diseñado para la entrada de contraseñas. En la Figura 32 se muestra el archivo que renderiza el componente de la contraseña, incluyendo las definiciones de testID (Líneas 81 y 85). Este identificador es esencial, ya que permite acceder al componente para realizar las validaciones y acciones necesarias durante y desde el archivo de ejecución de las pruebas.

Continuando con el ejemplo, se procedió a definir el archivo de pruebas, denominado TextInputPassword.test.tsx. La extensión .test es crucial, ya que permite a Jest reconocer este archivo como parte de las pruebas y proceder a su ejecución. Dentro de este archivo, se define un componente funcional llamado TestJestWrapper que utiliza el hook useForm para manejar el estado y la validación del formulario, específicamente para un campo llamado password. Este formulario inicializa el campo password con un valor vacío y configura que la revalidación del mismo se realice al perder el foco (onBlur). Se establece intencionalmente un error en el campo password para probar cómo el componente TextInputPassword maneja y muestra errores, utilizando un mensaje de error predeterminado desde los recursos de localización (idioma). Para el renderizado, TestJestWrapper envuelve el componente TextInputPassword, al cual le pasa el objeto control de useForm, dentro de un PaperProvider. Este PaperProvider aplica el tema MD3LightTheme de react-native-paper, asegurando que los estilos y diseños del tema se implementen correctamente en el componente. Esto es crucial para probar que el componente de entrada de contraseñas funciona adecuadamente en contexto con el manejo de errores y la aplicación de temas visuales, tal como se muestra en la Figura 33.

Por último, en las siguientes líneas del archivo de pruebas TextInputPassword.test.tsx se puede ver en el código la configuración de las pruebas para el componente TextInputPassword, inicia preparando el entorno con la inicialización del sistema de internacionalización y una limpieza post-prueba para evitar interferencias. Las pruebas verifican que el componente se renderiza correctamente y es accesible mediante

Figura 32: convertTestResults.js.

Figura 33: Código de prueba con Jest al componente TextInputPassword parte 1.

un testID específico, utilizando temporizadores falsos para gestionar la asincronía. También se asegura que el texto de error "Este campo es obligatorio" se muestre correctamente, validando la visibilidad y precisión del mensaje, esencial para la experiencia de usuario en formularios. Adicionalmente, se realiza una comparación de snapshots para detectar cambios no intencionados en la interfaz visual, esto se puede verificar paso a paso en la Figura 34.

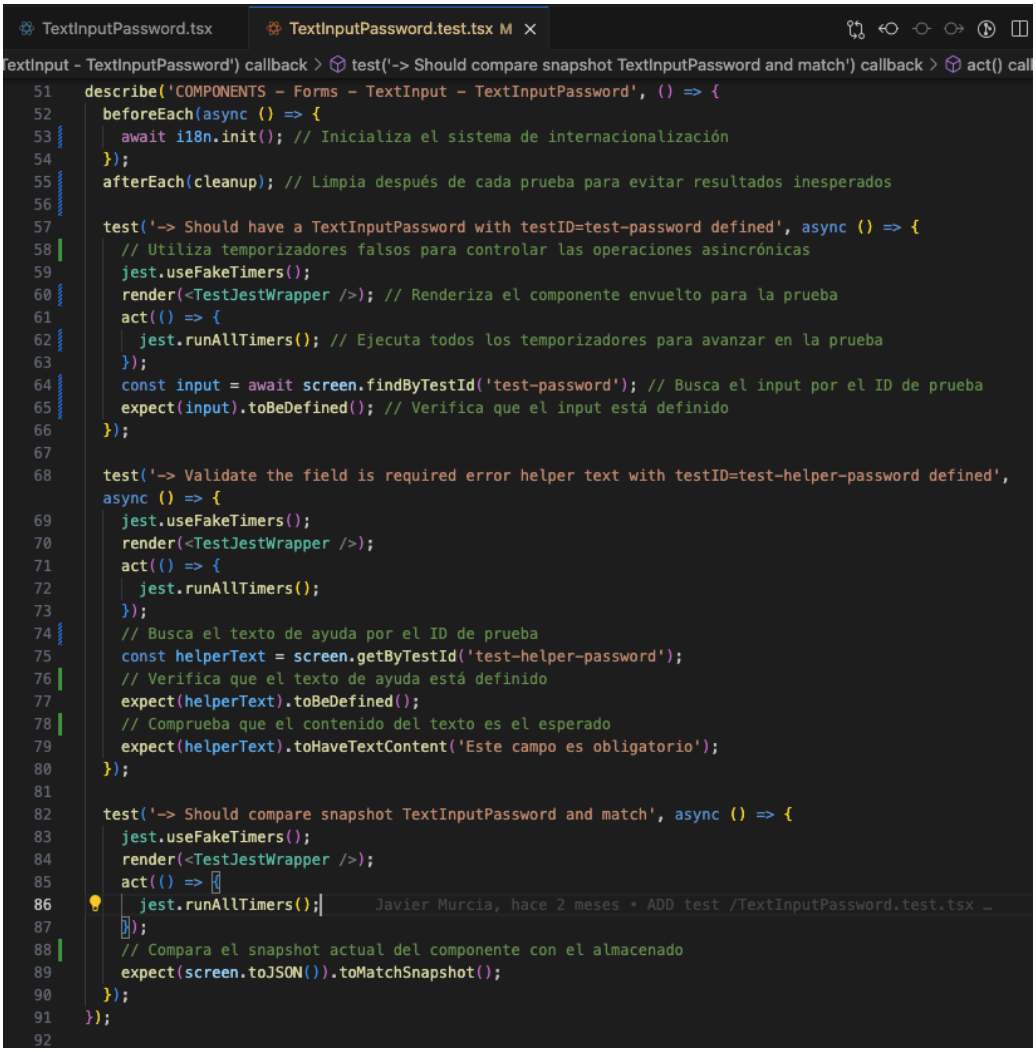


Figura 34: Código de prueba con Jest al componente TextInputPassword parte 2.

7.2. Pruebas integrales

En la Sección 4.1.28, se han definido las pruebas integrales. La institución actualmente utiliza Postman, una plataforma robusta que facilita el desarrollo de APIs al permitir a los usuarios diseñar, probar y documentar APIs de manera eficiente. Postman ofrece funcionalidades como la creación de solicitudes, pruebas automatizadas, simulación de entornos y colaboración en equipo, lo que la convierte en una solución integral para el ciclo de vida del desarrollo de API, según Postman (2024). Sin embargo, el área de calidad de software ha señalado en varias ocasiones que, aunque Postman es eficaz para ciertas tareas, no permite la realización de pruebas integrales de manera automatizada desde los frontends de los aplicativos. Esto ha llevado a la necesidad de realizar pruebas funcionales que requieren intervención humana, consumiendo tiempo y esfuerzo del personal. Debido a estas limitaciones, muchas veces estas pruebas resultan ser muy básicas y dejan fuera numerosos casos de prueba por el alto costo en tiempo y esfuerzo de las personas encargadas.

Teniendo en cuenta este contexto y continuando con la dinámica establecida en la sección anterior, se exploraron las opciones recomendadas por el equipo de React Native mostradas en la Figura 29. Entre estas opciones, se investigó Appium, que se define en su página oficial como una plataforma abierta para la automatización de pruebas de interfaces de usuario en diversas plataformas como móvil, web y escritorio. Appium admite múltiples lenguajes de programación y se compone de cuatro partes principales: Appium Core, Drivers, Clients, y Plugins, cada una especializada para integrar la funcionalidad de Appium en diferentes entornos y sistemas operativos, Appium Contributors (2023a).

Tras revisar la documentación y validar que Appium puede integrarse con el stack tecnológico que maneja la institución, se consideró viable su utilización para la automatización de pruebas por parte del equipo de desarrollo de la Super App. En este sentido, se propone utilizar el mismo formato que las pruebas unitarias, aunque estas pruebas, al ser integrales, se definirán a partir de las historias de usuario creadas en el documento de especificación en la Sección 5.1.5.

7.2.1. Desarrollo de pruebas integrales

Cargo	**Actividades**
Analista especialista WEB (ING. Javier Augusto Murcia Rodriguez - Autor)	- Modificación del documento para documentar las pruebas integrales. - Implementación del desarrollo de pruebas integrales.
Analista de calidad de software	- Revisión y aprobación del documento y las pruebas integrales propuestas para la automatización usando Appium.

Tabla 15: Equipo responsable para desarrollo de pruebas integrales

En un tiempo de 3 semanas, se logró lo siguiente:

1. Se realiza modificación del formato de documento de pruebas unitarias adecuándolo a las pruebas de integración, es revisado y aprobado por parte del área de calidad de software.

2. Se ha entregado el documento con las pruebas integrales definidas, basándose en el documento de requerimientos de la Super App. Esto implica que las pruebas se realizarán teniendo en cuenta las historias de usuario especificadas en dicho documento.

3. Se realizaron pruebas integrales utilizando la plataforma Appium, enfocándose inicialmente en dispositivos móviles Android. Las tareas ejecutadas incluyeron:

    - Instalación de Appium de manera global en un equipo Mac, haciendo uso del paquete de instalación de librerías npm de nodejs, Appium Contributors (2023b).
    - Appium opera con dos drivers distintos para la ejecución de pruebas en diferentes plataformas: UiAutomator2 para Android y XCUITest para iOS. Inicialmente, se realizo solo la instalación del driver UiAutomator2 para

Android. Además de la instalación, hay ciertos requisitos que deben cumplirse, como tener instalado el SDK de Android Studio y el Java JDK, entre otros. Detalles adicionales sobre estos requisitos se pueden consultar en Appium Contributors (2023c).

- Se realizó la instalación de Appium Inspector, es una herramienta de GUI para inspeccionar y interactuar con aplicaciones móviles durante las pruebas, permitiendo visualizar y manipular la interfaz de usuario de la app. Proporciona funcionalidades para especificar detalles del servidor Appium, interactuar con la captura de pantalla de la app, buscar elementos y ejecutar comandos del driver de Appium. Es accesible como una aplicación de escritorio independiente o como una aplicación web, Appium Inspector Contributors (2024). En este caso, se instala y se utiliza desde la aplicación de escritorio para validar los testID declarados y poder crearlos, modificarlos o eliminarlos en el código según las pruebas establecidas y los componentes a interactuar en las mismas.
- Se creó un proyecto en Node.js denominado "tests-nodejs-super-app-pacientes-fvl" con su respectivo repositorio. Este proyecto quedo funcionado como aplicación de consola destinada a realizar las pruebas automatizadas de la Super App. Se proporcionarán más detalles sobre este en la Sección 7.2.2.
- En el proyecto creado en el punto anterior, se procedió a la codificación de todas las pruebas integrales automatizadas que fueron especificadas en el documento.

7.2.2. Creación, configuración y codificación del proyecto para automatización de pruebas integrales usando Node.js y Appium

Para la creación de este proyecto, se consultó la documentación de Appium, encontrando una guía (Appium Contributors (2024)) que ilustra cómo escribir pruebas en JavaScript. Esto proporcionó un punto de partida sólido para desarrollar el proyecto utilizando Node.js, permitiendo así mantenerse dentro del stack de tecnologías que la institución soporta.

Se comenzó codificando un script principal (promptForPlatform, Figura 35) cuya tarea inicial, al lanzar la aplicación, es preguntar a través de la consola a qué plataforma se desean realizar las pruebas, donde el usuario puede digitar "A" para Android o "I" para iOS.

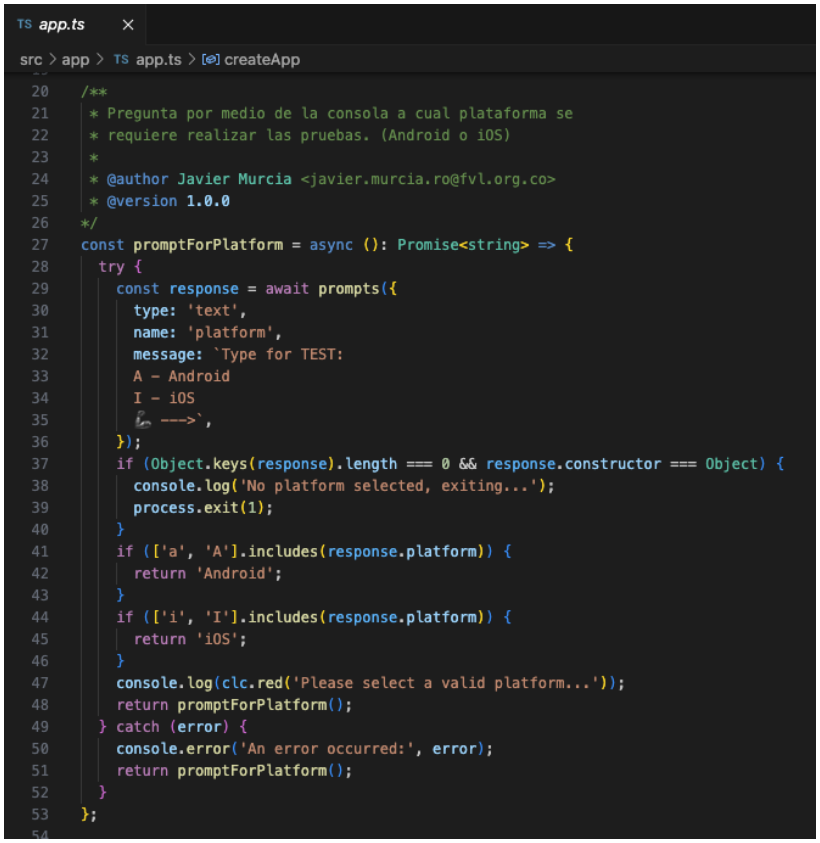


Figura 35: Selección de plataforma para pruebas integrales.

Al seleccionar Android, que es la funcionalidad implementada hasta ahora, se realiza el llamado a una función denominada runTestAndroid. Esta función contiene todas las pruebas, que están comentadas inicialmente, permitiendo activarlas o desactivarlas según sea necesario, como se muestra en la Figura 36, de igual manera en esta Figura se puede ver en la parte superior la configuración para poder crear el driver usando el servidor Appium y lograr la conexión ya sea con un dispositivo emulado o uno físico.

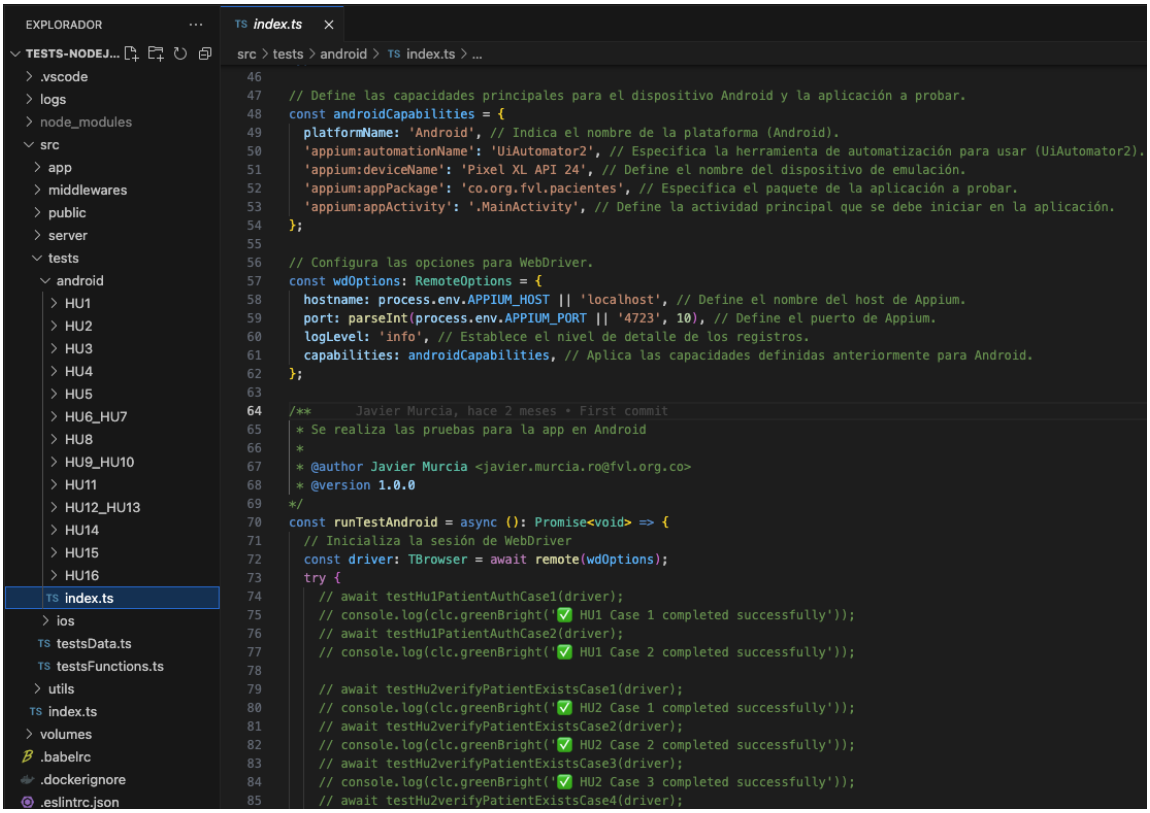


Figura 36: Función encargada de correr pruebas unitarias.

Durante la codificación de las pruebas, se observó que muchas acciones o procesos eran repetitivos, lo que resultaba en scripts con muchas líneas de código. Para optimizar este aspecto, se creó un archivo llamado testsFunctions. Este archivo centraliza las funciones comunes permitiendo su reutilización a lo largo del proyecto. Un

ejemplo de esto se puede ver en la Figura 37, donde la función “scrollFindElementAndClick” se encarga de realizar desplazamientos en la pantalla actual, buscando un componente gráfico por medio del testID y haciendo clic en él si es encontrado y todo por medio de los parámetros recibidos.

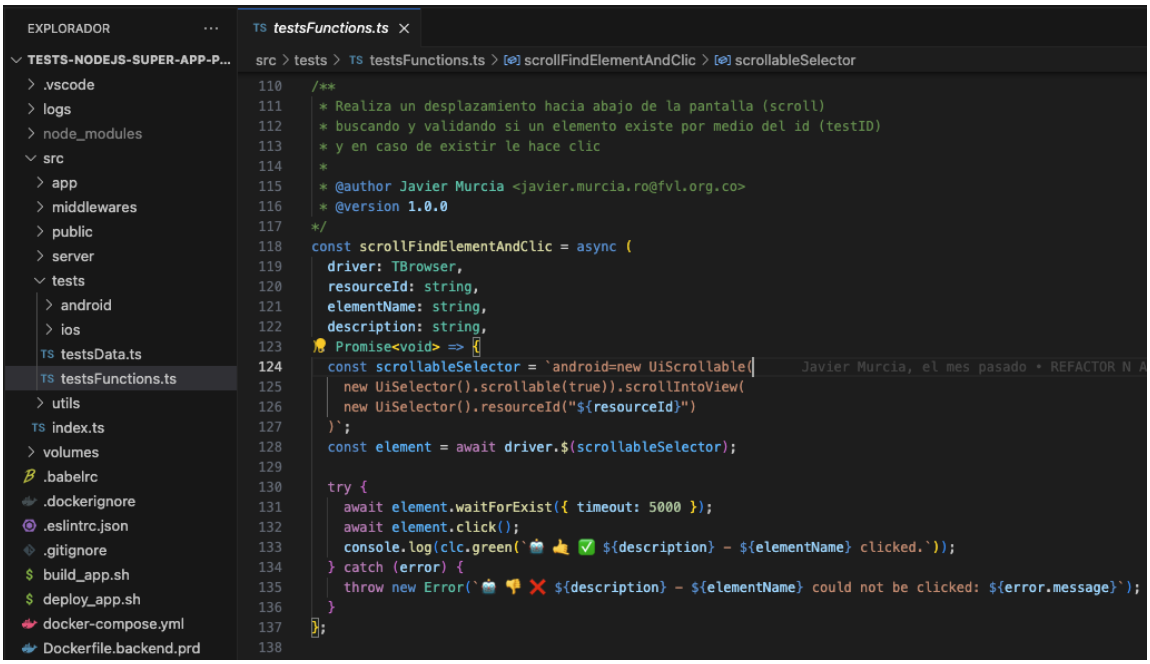
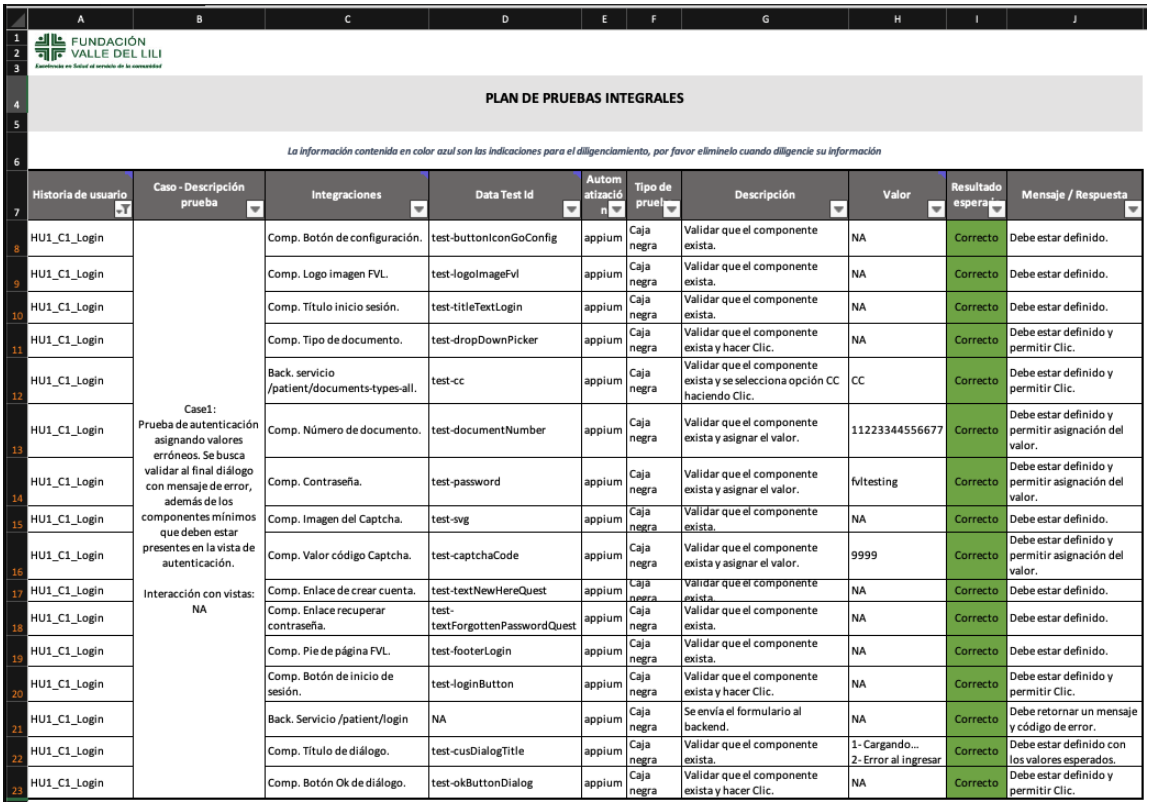


Figura 37: Implementación de función estándar para pruebas integrales.

7.2.3. Ejemplo de prueba integral, TextInputPassword

Como ejemplo para las pruebas integrales en este proyecto, se ha seleccionado la Historia de Usuario 1 (HU1 - Autenticación de paciente). Esta prueba involucra dos casos distintos:

- **Caso 1, Validación de error al introducir datos incorrectos:** Se prueba la inserción de datos de un usuario que no tiene cuenta creada. El objetivo es validar que la aplicación retorne un error apropiado, lo que verifica que la Super App gestiona correctamente los errores de autenticación desde el backend. La especificación de este caso se ilustra en la Figura 38 .



FUNDACIÓN VALLE DEL LILI

Experiencia en Salud al servicio de la comunidad

PLAN DE PRUEBAS INTEGRALES

La información contenida en color azul son las indicaciones para el diligenciamiento, por favor elimínelo cuando diligencie su información

Historia de usuario	Caso - Descripción prueba	Integraciones	Data Test Id	Automatización	Tipo de prueba	Descripción	Valor	Resultado esperado	Mensaje / Respuesta
HU1_C1_Login	Case1: Prueba de autenticación asignando valores erróneos. Se busca validar al final diálogo con mensaje de error, además de los componentes mínimos que deben estar presentes en la vista de autenticación. Interacción con vistas: NA	Comp. Botón de configuración.	test-buttonIconGoConfig	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Logo imagen FVL.	test-logoImageFvl	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Título inicio sesión.	test-titleTextLogin	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Tipo de documento.	test-dropDownPicker	appium	Caja negra	Validar que el componente exista y hacer Clic.	NA	Correcto	Debe estar definido y permitir Clic.
HU1_C1_Login		Back. servicio /patient/documents-types-all.	test-cc	appium	Caja negra	Validar que el componente exista y se selecciona opción CC haciendo Clic.	CC	Correcto	Debe estar definido y permitir Clic.
HU1_C1_Login		Comp. Número de documento.	test-documentNumber	appium	Caja negra	Validar que el componente exista y asignar el valor.	11223344556677	Correcto	Debe estar definido y permitir asignación del valor.
HU1_C1_Login		Comp. Contraseña.	test-password	appium	Caja negra	Validar que el componente exista y asignar el valor.	fvltesting	Correcto	Debe estar definido y permitir asignación del valor.
HU1_C1_Login		Comp. Imagen del Captcha.	test-svg	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Valor código Captcha.	test-captchaCode	appium	Caja negra	Validar que el componente exista y asignar el valor.	9999	Correcto	Debe estar definido y permitir asignación del valor.
HU1_C1_Login		Comp. Enlace de crear cuenta.	test-textNewHereQuest	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Enlace recuperar contraseña.	test-textForgottenPasswordQuest	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Pie de página FVL.	test-footerLogin	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C1_Login		Comp. Botón de inicio de sesión.	test-loginButton	appium	Caja negra	Validar que el componente exista y hacer Clic.	NA	Correcto	Debe estar definido y permitir Clic.
HU1_C1_Login		Back. Servicio /patient/login	NA	appium	Caja negra	Se envía el formulario al backend.	NA	Correcto	Debe retornar un mensaje y código de error.
HU1_C1_Login		Comp. Título de diálogo.	test-cusDialogTitle	appium	Caja negra	Validar que el componente exista.	1- Cargando... 2- Error al ingresar	Correcto	Debe estar definido con los valores esperados.
HU1_C1_Login		Comp. Botón Ok de diálogo.	test-okButtonDialog	appium	Caja negra	Validar que el componente exista y hacer Clic.	NA	Correcto	Debe estar definido y permitir Clic.

Figura 38: Prueba integral: Historia de usuario 1, Caso 1, validación de error al introducir datos incorrectos.

- **Caso 2, Autenticación con datos correctos:** Este caso prueba el inicio de sesión con credenciales correctas para confirmar que la autenticación funciona adecuadamente. Este proceso verifica la comunicación efectiva entre la aplicación móvil de la Super App y el backend del aplicativo. La especificación de este caso se ilustra en la Figura 39.

FUNDACIÓN VALLE DEL LILI

PLAN DE PRUEBAS INTEGRALES

La información contenida en color azul son las indicaciones para el diligenciamiento, por favor elimínelo cuando diligencie su información

Historia de usuario	Caso - Descripción prueba	Integraciones	Data Test Id	Automatización	Tipo de prueba	Descripción	Valor	Resultado esperado	Mensaje / Respuesta
HU1_C2_Login	Case2: Prueba de autenticación asignando valores correctos. Se busca validar al final el título de la vista de menú que contiene las Apps / Micro Apps, indicando que el inicio de sesión fue correcto. Interacción con vistas: HU14_MicroApps.	Comp. Tipo de documento.	test-dropDownPicker	appium	Caja negra	Validar que el componente exista y hacer Clic.	NA	Correcto	Debe estar definido y permitir Clic.
HU1_C2_Login		Back. servicio /patient/documents-types-all.	test-cc	appium	Caja negra	Validar que el componente exista y se selecciona opción CC haciendo Clic.	NA	Correcto	Debe estar definido y permitir Clic.
HU1_C2_Login		Comp. Número de documento.	test-documentNumber	appium	Caja negra	Validar que el componente exista y asignar el valor.	1130682656	Correcto	Debe estar definido y permitir asignación del valor.
HU1_C2_Login		Comp. Contraseña.	test-password	appium	Caja negra	Validar que el componente exista y asignar el valor.	fvltesting@	Correcto	Debe estar definido y permitir asignación del valor.
HU1_C2_Login		Comp. Imagen del Captcha.	test-svg	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.
HU1_C2_Login		Comp. Valor código Captcha.	test-captchaCode	appium	Caja negra	Validar que el componente exista y asignar el valor.	9999	Correcto	Debe estar definido y permitir asignación del valor.
HU1_C2_Login		Comp. Botón de inicio de sesión.	test-loginButton	appium	Caja negra	Validar que el componente exista y hacer Clic.	NA	Correcto	Debe estar definido y permitir Clic.
HU1_C2_Login		Back. Servicio /patient/login	NA	appium	Caja negra	Se envía el formulario al backend.	NA	Correcto	Debe retornar un mensaje y código de éxito.
HU1_C2_Login		Comp. Título de diálogo.	test-cusDialogTitle	appium	Caja negra	Validar que el componente exista.	Cargando...	Correcto	Debe estar definido con los valores esperados.
HU1_C2_Login		Comp. Título de Apps.	test-titleTextMicroApps	appium	Caja negra	Validar que el componente exista.	NA	Correcto	Debe estar definido.

Figura 39: Prueba integral: Historia de usuario 1, Caso 2, 2. autenticación con datos correctos.

Ambos casos también involucran la interacción del backend de la Super App con el microservicio de autenticación, que se conecta mediante PI/PO con los servicios ERP de la institución, donde se gestiona la información de los pacientes. En la Figura 40 se puede observar el código de ejemplo del Caso 1. Adicionalmente, en el Anexo A.9 se incluye un video que muestra la aplicación en funcionamiento y realizando las pruebas de ambos casos descritos.

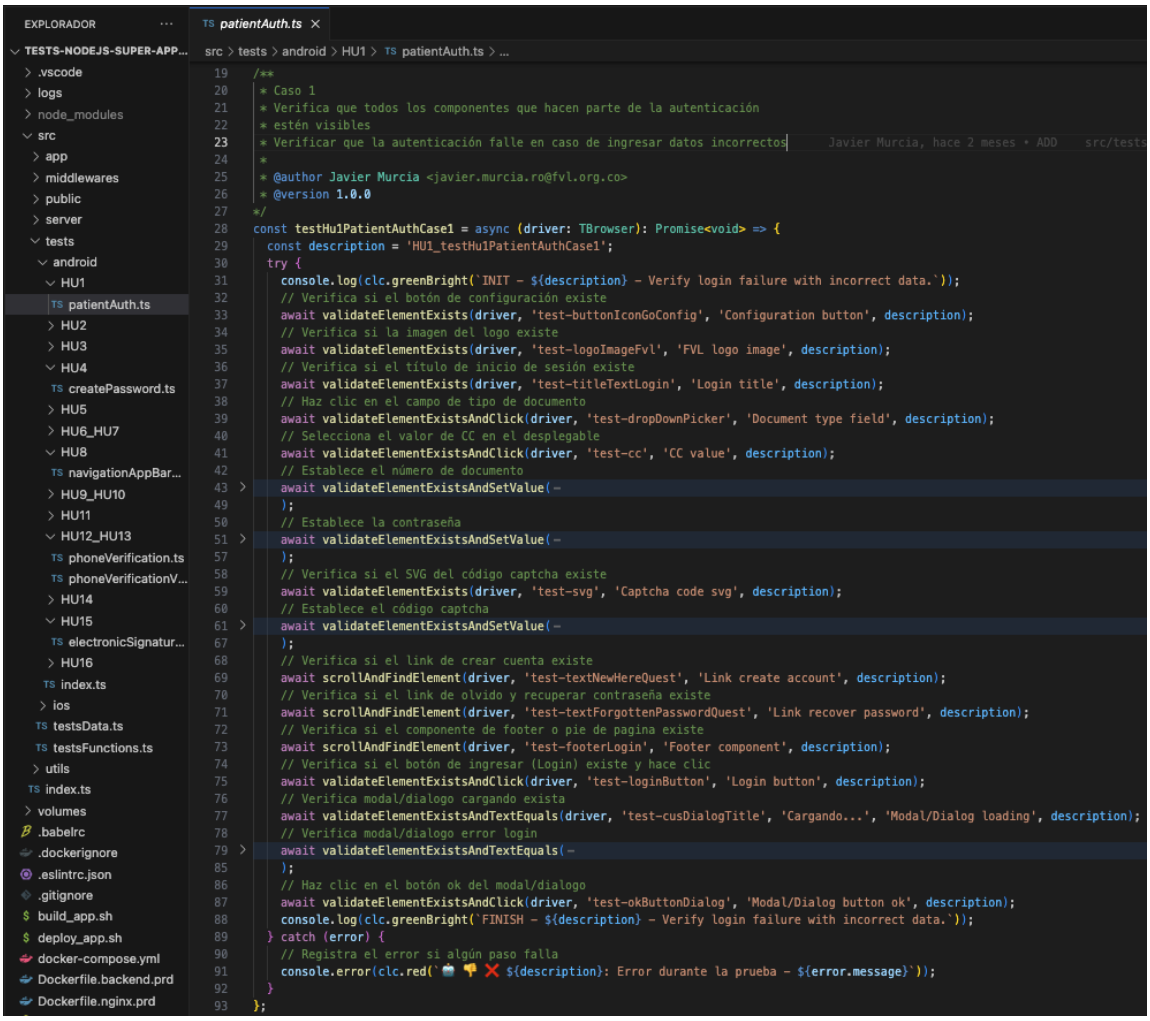


Figura 40: Código de ejemplo: historia de usuario 1, Caso 1.

8. Conclusiones y trabajos futuros

8.1. Conclusiones

Inicialmente, el proyecto se planteó como un prototipo de software para dispositivos móviles tipo Super App, con el objetivo de dar solución a una problemática en una institución de salud de alta complejidad. Sin embargo, tras completar el desarrollo, el proyecto dejó de considerarse un prototipo para convertirse en una iniciativa clave y prioritaria para el año en curso, programada para ser lanzada en un entorno productivo. El software no solo ha resuelto la problemática inicial, sino que también ha proporcionado múltiples beneficios a la institución, al haber abordado con éxito otras problemáticas. Entre los beneficios concretos que ha aportado el software, se nombran algunos:

- **Unificación de contraseñas para pacientes:** La institución cuenta con múltiples desarrollos propios y de terceros que apoyan su operación. Cada uno de estos sistemas manejaba de forma independiente la autenticación de los usuarios, lo que frecuentemente resultaba en datos duplicados o inconsistentes. En muchos casos, los pacientes terminaban teniendo hasta dos o tres contraseñas diferentes, causando confusión y una experiencia de usuario deficiente. Con el desarrollo de un microservicio aislado para la Super App de pacientes, estas plataformas ahora pueden integrarse a través de APIs. Esto permite una gestión unificada y centralizada de la información, respetando estándares y políticas de privacidad, según el usuario y los roles asignados.

- **Unificación y Actualización de Correo Electrónico y Número Celular:** Para asegurar el correcto funcionamiento de la Super App, que busca contar con canales de comunicación efectivos con los pacientes, es crucial verificar su identidad mediante su correo electrónico y número celular. Sin embargo, se detectó que muchos de estos datos estaban desactualizados o incorrectamente registrados en las tablas maestras del ERP. Además, la institución no contaba con el consentimiento necesario para emplear esta información en actividades de marketing relacionadas con sus servicios. A través de la Super App, estos datos

ahora se actualizan y verifican electrónicamente. Adicionalmente, al aceptar los términos y condiciones antes de enviar el formulario, se proporciona una base legal para que el área de marketing pueda contactar a los pacientes y ofrecer nuevos servicios orientados a su salud y bienestar.

Se logró un funcionamiento correcto en iOS y Android con un único código mantenible gracias al uso de React Native. Además, se garantiza la modularidad, integralidad y seguridad en la gestión de la información usando la metodología de arquitectura de software propuesta durante el proyecto. Esta metodología está enfocada en realizar pasos sistemáticos basados en fundamentos teóricos y prácticos, centrándose inicialmente en estos atributos de calidad. De igual manera, permitirá de forma iterativa mejorar estos atributos o incorporar nuevos, según sea necesario para el crecimiento de la solución y los requerimientos de los servicios de la institución. Al observar los resultados obtenidos y el enfoque teórico-práctico de esta nueva metodología, el área de informática ha decidido utilizarla no solo para el software de la Super App, sino también para todos los proyectos actuales y futuros desarrollos. Además, en caso de necesitarse desarrollos por parte de terceros, estos deberán guiarse por los documentos de arquitectura, que son resultado directo de la metodología aplicada y que serán proporcionados por el área de desarrollo de la institución.

Durante el desarrollo de este proyecto, se tuvo la oportunidad de poner a prueba un nuevo formato de documento de especificación de requerimientos, el cual generó resultados que superaron las expectativas del área. Las personas encargadas del área de gestión de proyectos realizaron una revisión y seguimiento de los artefactos y procesos generados, concluyendo que con este nuevo formato se agilizó el levantamiento de requerimientos y la definición de solución. Además, contar con nuevos artefactos, como los prototipos de bajo nivel, permitirá visualizar de manera anticipada los desarrollos a realizar, alineando expectativas en el desarrollo de funcionalidades, mitigando reprocesos debido a diferencias entre lo planteado o requerido frente a lo ejecutado. Teniendo en cuenta todo lo anterior, se aprobó este nuevo formato como el documento oficial para los requerimientos de soluciones de software de toda la institución.

La Micro App de firma electrónica, seleccionada como la primera para integrarse con la Super App, no solo permitió al área de desarrollo de software realizar las validaciones de capacidad técnica y funcional de la Super App para integrarse con el ERP de la institución y las particularidades de la infraestructura, sino que también permitió evidenciar que el enfoque seleccionado de integración mediante WebViews permitirá que estas aplicaciones puedan ser accedidas y ejecutadas fuera de la Super App para otros casos de uso con el mismo propósito. En otras palabras, esta Micro App quedó accesible mediante una URL pública para la firma electrónica de documentos, la cual puede ser enviada por SMS o correo electrónico al interesado, permitiendo su ejecución desde cualquier dispositivo (móvil, tablet y escritorio) debido a su diseño responsivo. Esta misma URL es consumida por la Super App, enfocándose en el usuario que ha iniciado sesión y en los documentos asignados para firmar. Esto representa una gran ventaja, dado que optimiza el mantenimiento y consumo de infraestructura tanto para las Apps como para las Micro Apps. Además, gracias a esta primera integración se pudo generar un documento inicial titulado "Integración con Super App Pacientes FVL". Este documento no solo busca estandarizar y asegurar el uso adecuado de la API para la correcta integración entre la Super App y las Micro Apps, sino que también ofrece indicaciones y sugerencias para el uso de la imagen de la institución, colores, tipografía y un stack tecnológico seleccionado para el desarrollo de estas. Esto asegura que las aplicaciones puedan ser soportadas por el equipo de desarrollo de la institución, independientemente de si son desarrolladas internamente o por terceros.

La propuesta realizada al área de seguridad informática de la institución respecto a la encriptación de la información al momento de plantear los Requisitos Arquitectónicamente Significativos (ASR) ha sido aceptada y se encuentra en implementación. Sin embargo, se ha recibido una sugerencia importante con relación con el riesgo de dependencia del Backend de la Super App. Este riesgo es evidente en la arquitectura de la solución, ya que en caso de que el Backend deje de funcionar por motivos de fuerza mayor, la aplicación debería tener capacidad de respuesta adecuada. Para abordar este riesgo, se recomienda investigar e implementar métodos que lo mitiguen. Desde el área de desarrollo se está planteando realizar una iteración

enfocada en mejorar el atributo de calidad de disponibilidad y otra en el manejo de errores. Estas medidas pueden incluir la creación de mecanismos de redundancia y recuperación automática, así como la implementación de mensajes de error claros y procedimientos alternativos para que los usuarios puedan continuar utilizando la aplicación con mínima interrupción. De esta manera, se buscará minimizar los riesgos y evitar una mala experiencia del usuario en un escenario como este.

La implementación de pruebas unitarias e integrales automatizadas dentro del proyecto de la Super App buscaba no solo mejorar la calidad del software en la institución, sino también mostrar un enfoque innovador que agiliza estos procesos y desarrollos. Inicialmente, se creó un documento estándar para la documentación de pruebas unitarias utilizando Jest. Posteriormente, este documento se modificó para la documentación de pruebas integrales, realizándolas de manera automatizada con Appium. Se llegó a la conclusión de que realizar ambos tipos de pruebas aumenta la carga laboral y el tiempo de desarrollo para el equipo encargado. Por ello, se decidió estandarizar un único documento para las pruebas integrales con Appium, enfocándose exclusivamente en el desarrollo de estas pruebas. Esta decisión se basó en la funcionalidad demostrada al validar, por ejemplo, que un componente de selección de tipo de documento carga con la información esperada, probando así esta unidad de manera integral. La decisión fue tomada en conjunto por el área de desarrollo de software y el área de calidad de software, evidenciando que el trabajo realizado en temas de prueba en este proyecto ha aportado un valor agregado significativo. Inicialmente, solo se había planteado revisar el plan de pruebas y validación, pero al final se estableció una metodología para realizar las pruebas de manera automatizada, buscando optimización de recursos y eficiencia del proceso.

Como última conclusión, y teniendo en cuenta los resultados obtenidos y las decisiones de adoptar lo aquí planteado por parte de la institución, este proyecto ha demostrado ser una pieza fundamental en la transformación digital de la institución. No solamente proporcionó una solución a un problema en específico, sino que también se generaron diferentes entregables metodológicos que dieron solución a otras problemáticas y mejoramiento de ciertos procesos desde un enfoque teórico y práctico. De igual manera, lo planteado en este documento podrá ser aplicado a cualquier

tipo de industria, no solamente a la de salud.

8.2. Trabajos Futuros

La Super App para pacientes FVL sigue en desarrollo y está programada para ser lanzada antes de finalizar el año 2024 para el uso de los pacientes de la institución y se han definido nuevas historias de usuario para incluir funcionalidades básicas como:

- El paciente pueda realizar la actualización de su correo electrónico.
- El paciente pueda realizar la actualización de su número celular.
- El paciente pueda realizar actualización de su dirección de residencia.
- El paciente pueda establecer y/o actualizar un contacto de emergencia.
- Funcionalidad de mensajería push a los dispositivos móviles.

En cuanto a las Apps/Micro Apps que se integrarán con la Super App, se han programado reuniones con los diferentes servicios para liderar el desarrollo funcional de las aplicaciones adicionales propuestas en la Sección 5.2. Actualmente, se continua trabajando en el documento de especificación de requerimientos para permitir que el área de informática realice los desarrollos de estas Micro Apps. Adicionalmente, desde informática se ha iniciado la propuesta y especificación de Micro Apps con funcionalidades básicas que se consideran esenciales para los pacientes, incluyendo:

- Búsqueda de servicios y directorio médico, permitiendo filtrar por sede. (Corto plazo).
- Opción "¿Cómo llegar?", que permitirá a los pacientes seleccionar la sede a la que desean dirigirse y abrir la ubicación en aplicaciones de navegación GPS en tiempo real, como Waze, Google Maps o Mapas de iOS. (Corto plazo).
- Seguimiento de citas médicas programadas, mostrando detalles como lugar, fecha, hora, especialidad y nombre del profesional. (Mediano plazo).

- Agendamiento de citas con validación cruzada con aseguradoras y calendarios de especialistas. (Largo plazo).

Es fundamental continuar con la elaboración del documento "Integración con Super App Pacientes FVL". Junto con el equipo de desarrollo, se han identificado mejoras significativas, como gestionar un path de versión al acceder y nombrar algunas funciones de las APIs, lo cual garantiza la longevidad de los servicios, asegurando su funcionamiento y compatibilidad hacia atrás. Además, se buscará desarrollar nuevas funcionalidades que aprovechen características del dispositivo como la cámara y el GPS, logrando hacer estas accesibles desde las Micro Apps.

Se buscará finalizar las pruebas integrales automatizadas para dispositivos iOS. Además, dado que la Super App es una aplicación móvil que podría ser utilizada por miles de pacientes diariamente, es crucial llevar a cabo pruebas no funcionales de rendimiento. Las áreas de infraestructura, desarrollo de software y calidad evaluarán cómo responde el sistema ante diferentes cargas de trabajo. Este análisis incluirá pruebas de carga para determinar la respuesta del sistema bajo condiciones de uso intensivo; pruebas de estrés para evaluar su comportamiento en situaciones extremas; y pruebas de escalabilidad para asegurar que el sistema puede expandirse sin comprometer su funcionalidad o rendimiento.

Al contar con mecanismos de automatización de pruebas y teniendo en cuenta que la Super App desde su lanzamiento podría tener más de una docena de Micro Apps, se planteará la utilización de prácticas de CI / CD (Continuous Integration and Continuous Delivery), buscando que al desarrollar estas se mantenga la calidad del software, acelerar los tiempos de entrega mejorando la colaboración y transparencia en el proceso de desarrollo.

La Super App será utilizada por pacientes de diversas edades y estratos socioeconómicos. Debido a esto, es crucial para la institución validar la usabilidad de la aplicación y recibir recomendaciones de sus futuros usuarios antes de su lanzamiento oficial. Actualmente, la institución cuenta con un grupo de pacientes voluntarios dispuestos a participar en este tipo de iniciativas. Se organizarán reuniones de grupo focal y una versión beta de la aplicación será probada por este grupo, lo que per-

mitirá realizar los ajustes necesarios antes de su publicación final en el App Store y Play Store.

La alianza con la empresa proveedora de equipos para laboratorios e imágenes diagnósticas es crucial para la institución. Esta empresa dispone de una plataforma propia donde los pacientes pueden consultar sus resultados médicos. Actualmente, se están efectuando reuniones con dicho proveedor para explorar la integración de esta plataforma con la Super App de la institución. El propósito de esta integración es facilitar el acceso de los pacientes a sus resultados a través de dispositivos móviles, mejorando así la accesibilidad y la calidad del servicio ofrecido.

9. Referencias Bibliográficas

Referencias

Amazon Web Services (2024). ¿qué son las pruebas unitarias? https://aws.amazon.com/es/what-is/unit-testing/. Último acceso: Abril 2024.

Android Open Source Project (2023). Android 7.0 compatibility definition document - section 3.4 web compatibility. https://source.android.com/docs/compatibility/7.0/android-7.0-cdd?hl=es-419#3_4_web_compatibility. Accedido: 2024-04-18.

API Levels Contributors (2024). Android api levels. https://apilevels.com/. Accedido: 2024-04-18.

Appium Contributors (2023a). Appium in a nutshell. https://appium.io/docs/en/2.5/intro/. Último acceso: Abril 2024.

Appium Contributors (2023b). Install appium - appium documentation. https://appium.io/docs/en/2.2/quickstart/install/. Último acceso: Abril 2024.

Appium Contributors (2023c). Install the uiautomator2 driver - appium documentation. https://appium.io/docs/en/2.2/quickstart/uiauto2-driver/. Último acceso: Abril 2024.

Appium Contributors (2024). Write a test (js) - appium documentation. https://appium.io/docs/en/latest/quickstart/test-js/. Último acceso: Abril 2024.

Appium Inspector Contributors (2024). Appium inspector. https://github.com/appium/appium-inspector. Último acceso: Abril 2024.

AppleInc. (2023). Wkwebview. https://developer.apple.com/documentation/webkit/wkwebview. Accedido: 2023-11-15.

ArrowHiTech (2023). React native webview: The comprehensive guide you need to know. https://blog.arrowhitech.com/react-native-webview-the-comprehensive-guide/. Accedido: 2023-11-15.

Atlassian (2024). Historias de usuario: Ejemplos y plantilla. https://www.atlassian.com/es/agile/project-management/user-stories. Último acceso: Abril 2024.

Axios Team, . (2024). Getting started with axios. https://axios-http.com/docs/intro. Accedido: 2024-01-24.

Baquero, A. P. D. (2021). Super apps: Opportunities and challenges. https://dspace.mit.edu/handle/1721.1/139585.

Bass, L., Clements, P., and Kazman, R. (2022). *Software Architecture in Practice*. Addison-Wesley Professional, 4 edition.

Bonnie Eisenman (2024). Learning react native. https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html. Último acceso: Julio 2024.

Bootstrap Team (2024a). Bootstrap documentation. https://getbootstrap.com/. Accedido: 2024-04-18.

Bootstrap Team (2024b). Browsers and devices - bootstrap v5.3. https://getbootstrap.com/docs/5.3/getting-started/browsers-devices/. Último acceso: Abril 2024.

Callstack (2023a). Re.pack: Webpack-based toolkit to build react native applications. https://www.callstack.com/open-source/re-pack. Accedido: 2023-11-15.

Callstack (2023b). A step-by-step guide to super app development. https://www.callstack.com/blog/step-by-step-guide-to-super-app-development. Accedido: 2023-11-15.

Ciceri, C., Farley, D., Harmel-Law, N., Keeling, M., Lilienthal, C., Rosa, J., Zitzewitz, A. v., Weiss, R., and Woods, E. (2022). *Software Architecture Metrics: Case Studies to Improve the Quality of Your Architecture*. O'Reilly Media, Inc.

Contributors, M. W. D. (2023). Fetch api - conceptos y uso. `https://developer.mozilla.org/es/docs/Web/API/Fetch_API#conceptos_y_uso`. Accedido: 2024-01-24.

Contributors, M. W. D. (2024a). Window interface. `https://developer.mozilla.org/en-US/docs/Web/API/Window`. Accedido: 2024-04-18.

Contributors, R. N. W. (2024b). Guide to react native webview. `https://github.com/react-native-webview/react-native-webview/blob/master/docs/Guide.md`. Accedido: 2024-04-18.

corp, N. (2020). 2020 naver annual report. `https://www.navercorp.com/en/investment/annualReport`. Accedido: 2023-06-09.

Cox, C., Hicks, B., and Gopsill, J. (2022). Improving mixed-reality prototyping through a classification and characterisation of fidelity. *Proceedings of the Design Society*, 2:353 – 362.

de la Función Pública, D. A. (1993). Ley 100 de 1993. `https://www.funcionpublica.gov.co/eva/gestornormativo/norma.php?i=5248`. [Accedido el: 04 de junio de 2023], ARTÍCULO 156, Literal i.

de salud Colombia, M. (2023). Derechos y deberes en salud. `https://www.minsalud.gov.co/proteccionsocial/promocion-social/Paginas/Calidad-salud-deberes-derechos-mecanismos.aspx`. [Accedido el: 05 de junio de 2023], Numeral 2.

Dick, J., Hull, E., and Jackson, K. (2017). *Requirements Engineering*. Springer International Publishing.

Etim, A. S., Jaiswal, C. P., Subroto, M., and Ortega, V. E. C. (2021). Managing information technology projects using agile methodology. pages 123–146.

Fairley, R., Bourque, P., and Keppler, J. (2014). The impact of swebok version 3 on software engineering education and training. *2014 IEEE 27th Conference on Software Engineering Education and Training (CSEE&T)*, pages 192–200.

Felip, R. (2020). Cómo escribir buenas historias de usuario. https://apiumhub.com/es/tech-blog-barcelona/como-escribir-buenas-historias-de-usuario/. Accedido: 2024-03-22.

Foundation, O. and contributors, N. (2023). Node.js - an open-source, cross-platform javascript runtime environment. https://nodejs.org/en/. Accedido el: 2023-10-20.

Fundación Valle del Lili (2024a). Certificación de excelencia. https://valledellili.org/certificaciones/certificacion-excelencia/. Último acceso: Abril 2024.

Fundación Valle del Lili (2024b). World's best hospitals 2023. https://valledellili.org/worlds-best-hospitals-2023/. Último acceso: Abril 2024.

Google, Play Store (2024). Instacare: Super health app. https://play.google.com/store/apps/details?id=attrayant.design.pat.instacare&hl=es_CL. Último acceso: Febrero 2024.

GoogleLLC (2023). Building web apps in webview. https://developer.android.com/guide/webapps/webview. Accedido: 2023-11-15.

Healthcare IT Today Contributors (2023). Medsi raises usd $10m in debt financing round to onboard first 30,000 mexican customers currently on the waitlist for its health assurance super app. https://www.healthcareittoday.com/2023/02/03/medsi-raises-usd-10m-in-debt-financing-round-to-onboard-first-30000-mexican-custom Último acceso: Febrero 2024.

Hylton, S., Ice, L., and Krutsch, E. (2022). What the long-term impacts of the covid-19 pandemic could mean for the future of it jobs.

i18next Team, . (2024). Introduction to i18next. https://www.i18next.com/. Accedido: 2024-05-18.

IBM (2024a). ¿qué es la prueba de software y cómo funciona? https://www.ibm.com/mx-es/topics/software-testing. Último acceso: Abril 2024.

IBM (2024b). ¿qué es una interfaz de programación de aplicaciones (api)? https://www.ibm.com/es-es/topics/api. Accedido: 2024-04-21.

IEEE (1999). Ieee standard for application and management of the systems engineering process. *IEEE Std 1220-1998*, pages 1–84.

IEEE (2000). *ISO/IEC standard for systems and software engineering - recommended practice for architectural description of software-intensive systems.*

IEEE (2024). Api - ieee terminology. https://developer.ieee.org/Terminology. Accedido: 2024-04-21.

InstaCare (2024). About instacare. https://instacare.pk/about. Último acceso: Abril 2024.

Investopedia (2024). Revolving credit. https://www.investopedia.com/terms/r/revolvingcredit.asp. Último acceso: Julio 2024.

Işitan, M. and Koklu, M. (2020). Comparison and evaluation of cross platform mobile application development tools. *International Journal of Applied Mathematics Electronics and Computers*, 8:273–281.

Jamali, G. and Oveisi, M. (2016). A study on project management based on pmbok and prince2. *Mathematical Models and Methods in Applied Sciences*, 10:142.

Jest Core Team (2024). Jest - delightful javascript testing. https://jestjs.io/. Último acceso: Abril 2024.

Jest Documentation Contributors (2024). Jest cli options - json output. https://jestjs.io/docs/cli#--json. Último acceso: Abril 2024.

Jiang, Q. (2018). Wechat - a super app from china. https://www.diggitmagazine.com/articles/wechat-super-app-china. Accessed: 2023-06-08.

Katalon (2024). What is integration testing? definition, examples, how-to. https://katalon.com/resources-center/blog/integration-testing. Último acceso: Abril 2024.

Leong, J., Yee, K. M., Baitsegi, O., Palanisamy, L., and Ramasamy, R. (2023). Hybrid project management between traditional software development lifecycle and agile based product development for future sustainability. *Sustainability*.

Lima, A. (2023). Colombian startup unicorn rappi, the first latin american super app? https://labsnews.com/en/articles/business/colombian-startup-unicorn-rappi-super-app/. Accessed: 2023-10-04.

Love, E. (2023). Rappi strengthens its path as a superapp and boosts its fintech take-off. https://fintechmorning.com/2023/05/30/rappi-strengthens-its-path-as-a-superapp-and-boosts-its-fintech-take-off/. Accessed: 2023-06-09.

Malavolta, I., Ruberto, S., Soru, T., and Terragni, V. (2015). Hybrid mobile apps in the google play store: An exploratory investigation. pages 56–59. Institute of Electrical and Electronics Engineers Inc.

Mascetti, S., Ducci, M., Cantù, N., Pecis, P., and Ahmetovic, D. (2021). Developing accessible mobile applications with cross-platform development frameworks. Association for Computing Machinery, Inc.

Material UI Team (2024). Supported platforms - material ui. https://mui.com/material-ui/getting-started/supported-platforms/. Último acceso: Abril 2024.

Medsi (2024). About us. https://www.medsi.mx/about-us. Último acceso: Febrero 2024.

Meta Open Source, Meta Platforms, I. (2023). React: The library for web and native user interfaces. https://react.dev/. Accedido: 2023-11-15.

Microsoft (2023). Integration architecture design. https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/integration. Accessed: 2023-06-07.

Munteanu, V. and Dragoș, P. (2021). The case for agile methodologies against traditional ones in financial software projects. *European Journal of Business and Management*, 6:134–141.

NPM, Jest Core Team (2024). Jest - npm. https://www.npmjs.com/package/jest. Último acceso: Abril 2024.

O'regan, G. (2017). Undergraduate topics in computer science concise guide to software engineering from fundamentals to application methods. http://www.springer.com/series/7592.

Postman (2024). Postman api platform. https://www.postman.com/. Último acceso: Abril 2024.

QPSoftware (2022). Your guide to wechat features. https://qpsoftware.net/blog/your-guide-to-wechat-features. Accessed: 2023-06-08.

React Documentation Contributors (2024). Reusing logic with custom hooks. https://react.dev/learn/reusing-logic-with-custom-hooks. Accedido: 2024-04-21.

React Native Documentation Contributors (2024). Testing overview - react native. https://reactnative.dev/docs/testing-overview. Último acceso: Abril 2024.

React Native Paper Team, . (2024). React native paper documentation. https://reactnativepaper.com/. Accedido: 2024-03-26.

React Native WebView Contributors (2024). React native webview. https://github.com/react-native-webview/react-native-webview. Accedido: 2024-04-18.

ReactNativeCommunity (2023a). React native webview. https://github.com/react-native-webview/react-native-webview. Accedido: 2023-11-12.

ReactNativeCommunity (2023b). React native webview api reference. https://github.com/react-native-webview/react-native-webview/blob/master/docs/Reference.md. Accedido: 2023-11-15.

Roa, L., Correa-Bahnsen, A., Suarez, G., Cortés-Tejada, F., Luque, M. A., and Bravo, C. (2021). Super-app behavioral patterns in credit risk models: Financial, statistical and regulatory implications. *Expert Systems with Applications*, 169.

Sehl, K. (2021). ¿qué es la aplicación line? todo lo que las marcas necesitan saber. https://blog.hootsuite.com/line-app/. Accedido: 2023-06-09.

Softesting (2019). Por qué son necesarias las pruebas de software. https://softesting.com/es_es/por-que-son-necesarias-las-pruebas-de-software/. Último acceso: Abril 2024.

StatCounter (2023a). Mobile operating system market share in colombia. https://gs.statcounter.com/os-market-share/mobile/colombia. Consultado el 4 de octubre de 2023.

StatCounter (2023b). Mobile operating system market share worldwide. https://gs.statcounter.com/os-market-share/mobile/worldwide. Consultado el 4 de octubre de 2023.

StrongLoop, I. y. o. c. d. e. (2023). Express - infraestructura web rápida, minimalista y flexible para node.js. https://expressjs.com/es/. Accedido el: 2023-10-20.

Taley, D. (2020). Comprehensive study of software testing techniques and strategies: A review. *International Journal of Engineering Research and*, V9.

Techopedia (2023). What is integration architecture? - definition from techopedia. https://www.techopedia.com/definition/16961/integration-architecture. Accessed: 2023-06-07.

Vailshery, L. S. (2023). Cross-platform mobile frameworks used by global developers 2022. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/. Flutter is the most popular cross-platform mobile framework used by global developers, according to a 2022 developer survey.

Vercel, I. (2023). Next.js - the react framework for the web. https://nextjs.org/. Accedido el: 2023-10-20.

Webpack JS, O. (2023a). Concepts webpack. https://webpack.js.org/concepts/. Accedido el: 2023-11-08.

Webpack JS, O. (2023b). Module federation - webpack. https://webpack.js.org/concepts/module-federation/. Accedido: 2023-11-15.

Ye, S. (2022). The rise of superapps in emerging countries. In *Proceedings of the 2022 4th International Conference on Economic Management and Cultural Industry (ICEMCI 2022)*, pages 1847–1856. Atlantis Press.

Zustand Team, . (2024). Introduction to zustand. https://docs.pmnd.rs/zustand/getting-started/introduction. Accedido: 2024-01-24.

10. Glosario de Términos

Modular: En el contexto del software, se refiere a un diseño de sistema que se divide en unidades independientes de funcionalidad llamadas módulos. Cada módulo puede funcionar de forma autónoma y puede interactuar con otros módulos para formar un sistema completo. Este diseño facilita la reutilización de código, el mantenimiento, la comprensión y la flexibilidad del software, permitiendo que los módulos sean intercambiados o reorganizados según sea necesario.

Escalable: En el contexto del software, se refiere a la capacidad de un sistema para manejar una creciente cantidad de trabajo de manera eficiente, o su capacidad para adaptarse y expandirse en respuesta al aumento de la demanda. Un sistema escalable puede aumentar su capacidad de procesamiento, ya sea agregando más recursos al sistema (escalabilidad vertical), como memoria o CPU, o incorporando más sistemas para compartir la carga de trabajo (escalabilidad horizontal).

Apps / App: Es una abreviatura de "aplicación" y, en el contexto de la tecnología, se refiere a un programa o un software diseñado para cumplir una función específica. Comúnmente se usa para referirse a las aplicaciones móviles y/o web, las cuales están diseñadas para ejecutarse en dispositivos móviles, como smartphones y tabletas, o en navegadores web.

Micro Apps / Micro App: son versiones más pequeñas y más simples de una aplicación, diseñadas para realizar una función específica con eficiencia y rapidez. A diferencia de las aplicaciones tradicionales que pueden tener una amplia gama de funciones y capacidades, una Micro App se centra en una sola tarea o conjunto de tareas. Estas aplicaciones son a menudo más rápidas y más fáciles de usar, ya que están optimizadas para una tarea específica y no tienen las funciones adicionales que pueden complicar las aplicaciones más grandes.

Encriptar: En el contexto de software, se refiere al proceso de convertir información o datos en un código que oculta su verdadero significado para prevenir el

acceso no autorizado. Este proceso se realiza utilizando algoritmos de cifrado y claves secretas. La información encriptada, también conocida como "cifrado", sólo puede ser leída o "descifrada" por aquellos que poseen la clave de descifrado correcta. La encriptación es una herramienta vital en la seguridad de la información, protegiendo los datos en tránsito y en reposo contra las amenazas de piratería y espionaje.

A. Anexos

A.1. Especificación del desarrollo de aplicativo móvil para pacientes (Super App)

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.2. Especificación del desarrollo de firma electrónica (App / Micro App)

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.3. Integrabilidad - Metodología de arquitectura software

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.4. Modularidad - Metodología de arquitectura software

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.5. Seguridad - Metodología de arquitectura software

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.6. Documento de integración Super App pacientes FVL

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.7. Documento de pruebas unitarias Super App pacientes FVL

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.8. Documento de pruebas resultados de pruebas unitarias Super App pacientes FVL

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.

A.9. Video de automatización de pruebas integrales Super App pacientes FVL

Para ver el anexo, puede hacer clic sobre el siguiente texto: Acceder aquí.