UCLV
Universidad Central
"Marta Abreu de Las Villas

MFC
Facultad de Matemática
Física y Computación

DEPARTAMENTO: CIENCIA DE LA COMPUTACIÓN

TRABAJO DE DIPLOMA

Título: TICKET – Aplicación para móviles de reservas en línea.

Autor: Jesús Daniel Saura Díaz

Tutores: MSc. Frank Reyes García

Ing. Jandro Daniel Velázquez Hernández

Consultante: MSc. Liset De la Hoz Rojas

Santa Clara, noviembre 2021

PENSAMIENTO

«Solo tú puedes decidir qué hacer con el tiempo que se te da ha dado».

- Dama Galadriel

DEDICATORIA

A mis padres.

AGRADECIMIENTOS

A mi mamá y mi papá. Agradecido eternamente de la educación, las correcciones, el aliento y del amor que me inculcaron para ser hoy el hombre que soy.

A mis abuelos, principalmente al incansable Aníbal por su apoyo incondicional, sus consejos y su ejemplo.

A mis hermanos y a las familias que han formado los momentos y recuerdos de toda una vida.

A mis tíos, todos aportando mucho para mi formación como persona, a mis primos que los quiero como hermanos.

A mis amigos que más que amigos, son familia, a Noslen por ser el primero de verdad, a Yasiel por acompañarme en estos 5 años codo a codo. Agradecer a esa aula tan peculiar, al Muppet, Yeralys, Gilberto, Kevin, Arnaldo, Izquierdo, Ernesto, Hailen, Javi, Yoel, Angelito, Yandro, Julio, Félix, Víctor, Giraldo y a las niñas, las sobrevivientes, Doris, Yaremi, Mary y Karina. Todos compañeros de viaje, repleto de momentos que compartimos y sufrimos, con los que entré, con los que se incorporaron y con los que me gradúo.

A mi doctora bella, por decir que sí, por ayudarme, por apoyarme, por estar a mi lado y poner orden en mi vida.

A todas esas personas que de alguna forma u otra aportaron, a los profesores, en especial a Lorgio y a Escobar, por su contribución e infinita disposición.

A la XETID y sus miembros, que me brindó la oportunidad de superarme y de ser parte de algo importante trabajando cada día como una familia.

A Frank por aceptar ser mi tutor y darme la confianza que necesitaba desde el día uno en KaleydoBit.

A cada crítica, cada aliento, cada obstáculo, cada puerta que se cerró y las otras muchas que se abrieron.

Quiero agradecerme por creer en mí, por no tomarme días libres, por nunca haber renunciado, por intentar tener más aciertos que errores y, sobre todo, por ser yo mismo en todo momento.

Orgulloso de lo que logré y ayudé a construir, de todo lo que hice y también de lo que decidí no hacer. Gracias a todos por el sacrificio, dedicación y por dejarme ser parte de sus vidas.

RESUMEN

La información brindada por distintas empresas de viajes, instituciones de trámites y negocios, muestran un incremento sustancial de las reservaciones en línea acompañado por la situación excepcional que propicia la pandemia de la COVID-19 en el mundo. Para la Empresa de Tecnologías de la Información para la Defensa (XETID) es de vital importancia contar con una aplicación móvil que complemente el sistema de reservaciones previamente elaborado.

En el presente trabajo de diploma se describe el diseño y desarrollo de una aplicación móvil para dispositivos con sistema operativo Android, que permite la realización de reservaciones de forma remota para negocios y servicios. Se analiza la situación e impacto de las aplicaciones móviles en la actualidad y su utilización en el sector de las reservaciones. Se recurre a la metodología de desarrollo Mobile-D para el diseño, análisis e implementación de la propuesta de solución con los artefactos correspondientes en cada etapa del desarrollo. Se tratan conceptos básicos de las tecnologías dentro del marco de trabajo de Android, funcionamiento de una API REST y diseño de la Base de Datos. Se utiliza Kotlin como lenguaje de programación y el patrón de arquitectura MVVM respaldada por la colección de bibliotecas Jetpack desarrollada por Google.

Como resultado se logró el desarrollo de la aplicación móvil para dispositivos Android en el tiempo planificado, dejando como consecuencia un producto de software capaz de la autenticación y registro de usuarios, búsqueda de servicios y realización de reservaciones de forma remota.

Palabras claves: Android, aplicación, móvil, desarrollo, reservación, Kotlin, Mobile-D.

ABSTRACT

The information provided by different travel companies, processing institutions and businesses, show a substantial increase in online reservations accompanied by the exceptional situation that propitiates the COVID-19 pandemic in the world. For the Defense Information Technology Company (XETID) it is vitally important to have a mobile application that complements the previously developed reservation system.

This diploma work describes the design and development of a mobile application for devices with Android operating system, which allows making reservations remotely for businesses and services. The current situation and impact of mobile applications and their use in the reservation sector are analyzed. The Mobile-D development methodology is used for the design, analysis, and implementation of the solution proposal with the corresponding artifacts at each stage of development. Basic concepts of the technologies within the Android framework, the operation of an API REST and the design of the Database are covered. Kotlin is used as the programming language and the MVVM architecture pattern supported by the Jetpack library collection developed by Google.

As a result, the development of the mobile application for Android devices was achieved in the planned time, resulting in a software product capable of authenticating and registering users, searching for services, and making reservations remotely.

Keywords: Android, application, mobile, development, reservation, Kotlin, Mobile-D.

TABLA DE CONTENIDO

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

Las aplicaciones y las nuevas tecnologías móviles de comunicación están presentes en la vida cotidiana e influyen continuamente sobre el desarrollo de la sociedad. Cada día nacen aplicaciones que ofrecen nuevas oportunidades de negocio, comunicación, ocio, accesibilidad e inclusión. Softwares que, instalados en los teléfonos inteligentes (smartphones) facilitan las tareas diarias de trabajo, posibilitan una mayor y mejor interconexión. El presente y futuro debe dirigirse a aportar soluciones, información veraz, desarrollo de servicios y entretenimiento, dejando atrás malas praxis como el uso indebido de datos del usuario, creación de adicciones o propuestas sin valor.

Los dispositivos móviles se han instaurado en la vida de tal forma que prácticamente todas las personas llevan en su bolsillo un terminal móvil, la gran mayoría con sistema operativo Android por la increíble cuota mundial de mercado que representa de un 72.73% (Stats, 2021c), teniendo en Cuba hasta el 90.62% (Stats, 2021b) del mercado.

La portabilidad de los sistemas orientados a dispositivos móviles es muy beneficiada por la capacidad de realizar tareas remotas o automatizadas, muchas de ellas sin necesidad de la intervención directa de la persona como, por ejemplo; una transferencia bancaria ya no requiere asistir a un banco, la firma digital puede sustituir al proceso manual de firmar un documento, así como la acción de comprar productos se facilita mediante los comercios electrónicos (e-commerce). Los ejemplos son disímiles y las ventajas muy prometedoras.

La pandemia de COVID-19 obligó a reestructurar las relaciones tanto profesionales como personales. En medio de la crisis mundial resultante, las empresas debieron reinventar la atención al cliente, debido al cierre parcial o total de negocios que impactó de manera negativa su estabilidad. Para cualquier negocio su premisa principal es la de captar y sumar clientes; pero el confinamiento obligatorio de la sociedad colocó una enorme barrera en el mundo físico, impidiendo, de algún modo, mantener las comunicaciones con ellos. Sin embargo, las nuevas tecnologías de la información y las comunicaciones rompen todas las barreras espaciotemporales, permitiéndole a los negocios seguir manteniendo contacto con sus clientes, sin necesidad de compartir el mismo espacio

físico. Al mismo tiempo, ayuda a captar nuevo personal y fidelizar a los que ya están mediante el uso del internet. (Digital, 2021)

La Empresa de Tecnologías de la Información para la Defensa, conocida como XETID, es una empresa de origen cubano, fundada como empresa en el 2013 y dedicada al sector del software, la automática y las comunicaciones. (EcuRed, 2018)

Como parte del proceso de informatización por el cual que transcurre el país, constituye una necesidad la creación de un sistema de reservaciones en línea, que facilite el acceso a los distintos servicios que brindan las entidades o negocios, haciéndolos accesibles a todo público. Como parte fundamental de este sistema, es de gran importancia disponer de una aplicación para dispositivos móviles que permita automatizar los procesos de reservación y simplificar el trabajo de los proveedores de los servicios.

Problema de investigación

No se cuenta con una aplicación para dispositivos móviles, que permita las reservaciones de forma remota (online) en los distintos negocios o establecimientos.

Objetivo General

Desarrollar una aplicación móvil que permita realizar reservaciones de forma remota en negocios y servicios para utilización en Cuba.

Para dar cumplimiento al objetivo general se trazan los **objetivos específicos** siguientes:

- Determinar las bases teóricas existentes sobre el desarrollo de aplicaciones móviles que contribuyan a la realización de reservaciones en internet.
- Diseñar la aplicación móvil que permita las reservaciones en los negocios y servicios disponibles en el sistema *ticket*, desarrollado por XETID.
- Implementar la aplicación móvil en base a las técnicas y herramientas identificadas.

Preguntas de investigación

1. ¿Cuáles son las bases teóricas existentes acerca del desarrollo de aplicaciones móviles que contribuyen a la realización de reservaciones en línea?
2. ¿Cómo diseñar una aplicación móvil que posibilite la creación de reservaciones de forma remota para los negocios y servicios del sistema *ticket*?
3. ¿Cómo implementar la aplicación para dispositivos móviles que sea capaz de realizar las reservaciones en línea?

Justificación

Las tecnologías móviles representan a un campo de constantes cambios que debe ser profundamente estudiado por su impacto directo en la vida cotidiana de la sociedad. Las investigaciones de las nuevas técnicas, herramientas y marcos de trabajo son de vital importancia para toda la comunidad de desarrolladores del sector porque, son estas, las que dan lugar a las mejoras de software, las nuevas tecnologías y modernos métodos de trabajo.

La barrera en el mundo físico que implantó la COVID-19 afecta directamente a todos los procesos presenciales que se realizan para la obtención de diferentes servicios, por lo que se hace necesario un software orientado a los móviles, que contribuya a la agilización y automatización de estos procedimientos.

En Cuba no se dispone de un sistema de reservas en línea de carácter nacional para una variedad de servicios tales como los relacionados con: instituciones estatales, transporte, gastronomía, alojamiento, recreación, entre otros, que permita la obtención de citas, realización de pagos y valoraciones del servicio recibido. Por lo que, se hace necesario el desarrollo de una aplicación, que se integre a un sistema de reservaciones y facilite el acercamiento entre clientes y proveedores de servicios.

Hipótesis

Si se implementa la aplicación móvil, posibilitará a los usuarios obtener reservaciones de forma remota en los distintos servicios que se ofertan.

1 CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

En este capítulo se exponen conceptos y aspectos generales en el desarrollo de aplicaciones para dispositivos móviles mediante el análisis del uso de estas aplicaciones, así como el impacto en la sociedad. Se describen los principales métodos, tecnologías y aplicaciones existentes, relacionados con la posible solución del problema de investigación planteado. Para ello, se realiza un estudio detallado de las metodologías de software relacionadas al desarrollo de aplicaciones móviles, se identifican las tendencias actuales en el mercado en cuanto a tecnologías y se alcanzan conclusiones sobre herramientas y lenguajes de programación a utilizar, que permiten un sistema escalable, correctamente diseñado y acorde a los avances tecnológicos brindados por los marcos de trabajo actuales.

1.1 OBJETO DE ESTUDIO

La mayoría de las reservaciones en Cuba se realizan de forma presencial, mediante llamada telefónica al local, contactando a través de las redes sociales o utilizando el correo electrónico. El cliente llega a conocer del servicio a causa de que algún familiar, amigo o conocido se lo hace saber, quizás hasta le ofrece su valoración, en caso de contar con alguna experiencia personal.

Para un negocio que brinde servicios, realizar procesos manuales o presenciales afectan directamente a la eficacia y rentabilidad de este. A medida que el negocio crece y aumenta la cantidad de clientes, se hace necesario tener más trabajadores y más tiempo para poder gestionarlos. Las redes sociales son un medio muy empleado para el marketing y publicidad, pero carece de herramientas de gestión y de plataformas de pago integradas. La visibilidad de los servicios que se prestan no es la mejor porque solo llega a públicos específicos, no incluye algoritmos de búsquedas avanzadas o de recomendaciones a partir de las preferencias del usuario, por lo que encontrar lo que se busca se torna difícil.

1.1.1 Procesos objeto de automatización

Se hace necesario automatizar distintos procesos relacionados con la obtención de tickets (reservaciones) para los establecimientos.

La realización de búsquedas de servicios y establecimientos es una funcionalidad muy importante ya que es la acción desencadenante de la mayoría de las demás funcionalidades. Para ello se hace elemental el empleo de filtros de búsqueda que faciliten y agilicen la localización del servicio para su reservación. Como parte de este proceso se dispone de una serie de pasos como pueden ser la elección de día, la realización de pagos por medio de distintas plataformas, la selección del turno y el beneficiario a quien irá dirigida la reservación.

También se requiere automatizar la valoración de los servicios por parte de los usuarios, la cancelación de los tickets y la gestión de favoritos en establecimientos y servicios de modo que se pueda acceder rápidamente a ellos.

Para una mejor comunicación entre usuarios y proveedores de servicios, se pretende implementar un sistema de notificaciones con alertas, ofertas e informaciones relacionadas con las reservaciones.

1.2 CONSIDERACIONES SOBRE EL ESTADO DE LAS APLICACIONES MÓVILES

Una **aplicación móvil**, también llamada app móvil, es un tipo de aplicación diseñada para ejecutarse en un dispositivo móvil, que puede ser un teléfono inteligente o una tableta. Incluso si las aplicaciones suelen ser pequeñas unidades de software con funciones limitadas, se las arreglan para proporcionar a los usuarios servicios y experiencias de calidad. (Herazo, 2019)

Debido a los recursos de hardware limitados de los primeros dispositivos móviles, las aplicaciones evitaban la multifuncionalidad. Sin embargo, incluso si los dispositivos que se utilizan hoy en día son mucho más sofisticados, las aplicaciones móviles siguen siendo funcionales. Así es como los propietarios de aplicaciones móviles permiten a los consumidores seleccionar exactamente las funciones que deben tener sus dispositivos. (Herazo, 2019)

A diferencia de la web, las aplicaciones móviles están disponibles para su ejecución en forma instantánea: basta que el usuario toque su ícono para que ésta se inicie, sin necesidad de abrir un navegador, escribir direcciones web o usar buscadores. En un mundo donde la inmediatez es primordial, la app está ahí para ser usada sin más demora. (Talent, 2017)

Con el uso de aplicaciones móviles los usuarios disminuyen procesos manuales, errores en la captura de datos y tareas ineficientes. Además, se comunican mejor y comparten información con mayor fluidez, lo que resulta en incrementos en los niveles de productividad y ganancia.

1.3 IMPACTO DE LAS APLICACIONES MÓVILES EN LA SOCIEDAD Y LOS NEGOCIOS.

El porcentaje de tráfico web global en teléfonos móviles ha aumentado durante la última década. En julio de 2021, el 56,75% de todo el tráfico web procedía de teléfonos móviles. En el año 2011, esta cifra era de un escaso 6,09%. Luego de cinco años, en el 2016, el porcentaje de tráfico web en dispositivos móviles ya se había multiplicado por siete, hasta un 45,91%. (Oberlo, 2021)

En Cuba las estadísticas son incluso aún más determinantes, representando un 72.33% (Stats, 2021a) el acceso a internet por medio de los dispositivos móviles frente al 27.67% (Stats, 2021a) que representa el acceso por computadoras de escritorio.

En la actualidad resulta un imperativo para las empresas comprender la forma de pensar y actuar de sus consumidores al adquirir sus productos/servicios, por lo que éstas deben enfocarse a responder preguntas como: qué compran, qué los motiva a comprar, cuándo lo compran o de qué manera. Son varios los factores que han modificado el comportamiento del consumidor, siendo las nuevas tecnologías de la información y la comunicación (TIC) y el E-Commerce, especialmente los principales responsables del cambio en las necesidades de adquisición y consumo de las sociedades. (School, 2018)

El ritmo de vida acelerado de las personas ha generado un cambio significativo en la manera de adquirir productos/servicios, los consumidores son cada vez más exigentes no solamente con la calidad de los objetos que adquieren, sino también con los precios que están dispuestos a pagar por ellos, los servicios exclusivos y personalizados que las

empresas puedan ofrecerles, y especialmente la inmediatez en que los puedan recibir. (School, 2018)

Las empresas buscan que sus aplicaciones estén disponibles en los dispositivos móviles sin que ello implique una cantidad excesiva de tiempo de desarrollo. Hay una variedad de estrategias de desarrollo para diseñar aplicaciones móviles, desde soluciones móviles y preempaquetadas que no requieren código hasta soluciones completamente personalizadas y plataformas de desarrollo integradas en los dispositivos móviles. (Hat, 2021)

Para muchas empresas, la estrategia de desarrollo basada en servicios con funciones móviles integradas se está convirtiendo en el método estándar. La integración del desarrollo de aplicaciones móviles en una estrategia más amplia basada en microservicios nativos de la nube ofrece muchos beneficios, como el aumento de la productividad, la reducción de los costos y la mejora de la seguridad, así como mayores niveles de visibilidad y control. (Hat, 2021)

La creación de una aplicación móvil para una empresa o negocio permite establecer vínculos con el público y comunicar información rápidamente. Se puede llegar a más clientes potenciales y ayudará a obtener beneficios adicionales. Sin importar el sector, invertir en una aplicación móvil puede ser un salto significativo en el alcance o impacto de cualquier sistema.

1.4 SECTOR DE LAS RESERVAS EN LÍNEA

Las tendencias de los consumidores digitales continúan cambiando. El uso de escritorio está disminuyendo continuamente y el móvil resulta más recurrente. Los sistemas de reservas online son unas de las mejores formas de aumentar las ventas en un negocio que ofrece servicios en internet, ya que los clientes pueden acceder desde cualquier dispositivo. (Lab, 2019)

Empresas con alta experiencia en el mundo de las reservas como Bookitit, SimplyBook.me, Planyo, incluso Upbooking poseen aplicación móvil para que los clientes puedan reservar fácilmente sus servicios, cambiar una reserva, ver el historial de reservas, realizar pagos, comprar tarjetas de regalo y demás.

Es importante conocer todos los recursos que se pueden utilizar. Un sistema de reservas online es una plataforma que permite que los clientes y el negocio puedan comunicarse de forma muy rápida y eficaz. (Lab, 2019)

Las ventajas de tener un sistema de reservas online son muchas.

Accesibilidad: El cliente puede acceder en cualquier momento y a cualquier hora al sistema de reservas online. Por lo tanto, no es necesario que esperen a que el negocio esté abierto. Así, el negocio no pierde la oportunidad de satisfacer a un cliente en ningún momento. (Lab, 2019)

Organización: De un solo vistazo, el proveedor del servicio puede controlar la organización de su local, las reservas realizadas y la cantidad de clientes que acudirán a sus instalaciones. Por lo tanto, puede organizar y planificar mejor el trabajo del servicio y los horarios, con y sin reserva, según vayan llegando. (Lab, 2019)

Eficiencia: Aunque todavía hay clientes que llaman para hacer sus reservas, cada vez son menos los que lo hacen. De esta manera los empleados no están continuamente respondiendo llamadas ni reorganizando las disponibilidades. (Lab, 2019)

Promoción: Poder promocionarse o hacer descuentos es muchísimo más fácil. Se puede utilizar para ofrecer cupones de descuento o códigos promocionales. Y así, en temporadas bajas, por ejemplo, atraer a más clientela. (Lab, 2019)

Competencia: Es importante no quedar obsoleto e ir avanzando al ritmo de los competidores. Cuantas más facilidades se dé a los clientes, más satisfechos quedan y mejor es la experiencia que viven. Por lo tanto, las posibilidades de que vuelvan aumentan considerablemente. (Lab, 2019)

La digitalización de las reservaciones ha demostrado ser un proceso que, además de facilitar el trabajo, ayuda a aumentar las ganancias de los negocios. (Lab, 2019)

1.5 SISTEMAS AUTOMATIZADOS EXISTENTES VINCULADOS A LAS RESERVAS EN LÍNEA.

En el mundo hay varios sistemas de reservas online comparables con el que se desea trabajar. Estos son muy potentes y se pueden tomar como ejemplo para lograr conformar un software de calidad. En Cuba actualmente, este tipo de sistema no tiene destacados exponentes. Solo existen simples funcionalidades para reservaciones muy específicas, como pueden ser, reservar un hotel, una cita en un establecimiento, pero siempre ajustándose al sector o categoría a la que pertenece el negocio, sin llegar a conformar un sistema de reservaciones genérico que se pueda contratar de manera independiente y que posibilite la gestión detallada de las planificaciones, turnos, servicios y clientes. Puede verse ejemplificado en sitios webs como en los de grupos hoteleros Islazul (Islazul, 2021), Cubanacan (S.A, 2021a) o Havanatur (S.A, 2021b). Internacionalmente existen soluciones más inclusivas con el enfoque del sistema de reservaciones que tiene como objetivo el presente trabajo.

A continuación, se analiza solo algunos de los sistemas más relevantes mundialmente.

SimplyBook.me

SimplyBook.me es un excelente sistema de reserva de citas online para cualquier empresa que quiera digitalizar la programación de sus servicios profesionales. Impresiona sus numerosas funcionalidades, así como la interfaz intuitiva y fácil de usar. Los clientes de los negocios pueden programar y reservar citas online en el sitio web corporativo a través de cualquier dispositivo y lugar, como puede ser la página de Facebook. (Terol, 2015)

Lo más destacado de este sistema radica en la brevedad del proceso de reservas de citas, la posibilidad del envío de recordatorios a los clientes y se le puede ofrecer a los usuarios del negocio que compartan su experiencia en el servicio una vez finalizado. A diferencia de otros sistemas de reservas de citas online, SimplyBook.me ofrece un nivel de personalización muy avanzado. Mediante el código HTML y CSS se puede diseñar a gusto la apariencia del sitio web de reservas de citas online. Posee además muy buena integración con distintas herramientas como los calendarios de Google y Outlook, con

las redes sociales, los sitios creados con WordPress y con importantes plataformas de pagos como PayPal y Stripe. (Terol, 2015)

Bookitit

El software online Bookitit es ideal para aquellos negocios que desean ofrecer a los clientes un calendario donde realizar reservas para citas online, y así conseguir rentabilizar al máximo la presencia en internet. Una de sus grandes ventajas es que proporciona una aplicación móvil nativa y gratuita, disponible para iOS y Android, para que se puedan gestionar todas las reservas online. Con este sistema se ofrece un canal online de atención al cliente, posibilita la automatización del proceso de citas, la reducción de los gastos empresariales y el tiempo de gestión. (Terol, 2015)

En este caso, al ser Bookitit un motor de reservas para citas online no existen planes de precios con una suscripción mensual o anual. Hay diferentes tarifas en función del número de agendas que sean necesarias utilizar para el negocio. (Terol, 2015)

Planyo

El software de reservas online Planyo tiene la gran ventaja que permite su uso para numerosos tipos de negocios y necesidades empresariales. También posee funcionalidades que permiten a los negocios sacar la máxima rentabilidad a la presencia corporativa en internet y ofrecer un rápido servicio de atención al cliente. Aquí se tiene la posibilidad de poder contratar entre sus tres tipos de planes de precios, incluyendo la versión gratuita, en función de las necesidades empresariales y funcionalidades que se deseen utilizar para el negocio. (Terol, 2015)

Algunas de las características más destacadas en este sistema de reservas online son la gran cantidad de integraciones con otras plataformas (Facebook, WordPress, Joomla, Airbnb, etc), la disponibilidad de las aplicaciones móviles nativas, generación de facturas automatizadas e integración con Google Analytics para estadísticas y reportes. (Terol, 2015)

Upbooking

El sistema de reservas online Upbooking es ideal para pequeños negocios turísticos que tienen pocos recursos económicos. Es un motor de reservas online gratuito de Marca Blanca para hoteles, por tanto, no depende de cuotas, ni comisiones, ni tampoco gastos de gestión. (Terol, 2015)

La principal ventaja que ofrece es que permite personalizar las tarifas, los descuentos, la disponibilidad, y opciones de garantías, tanto para las agencias, empresas o grupos individuales. Tiene la posibilidad de definir las promociones especiales para atraer nuevos clientes, como por ejemplo para largas estancias en hoteles, reservas anticipadas o reservas de última hora, así como crear beneficios específicos para fidelizar a los clientes con precios o tarifas especiales. (Terol, 2015)

Luego del análisis de algunas de las plataformas existentes y más relevantes en el sector de las reservaciones en línea, constituye de gran importancia evaluar las ventajas, aportes y beneficios que podría tener un nuevo sistema. Como uno de los puntos a resaltar estaría la independencia y soberanía tecnológica que brinda un producto o solución informática de factura nacional, único en el país a la escala que se pretende trabajar y adaptado a las condiciones de la sociedad cubana. Otro aspecto que cabe destacar que ninguno de los sistemas anteriormente analizados posee un apartado de “sala de espera” o “cola virtual” para aquellos usuarios que no puedan alcanzar a las disponibilidades en un momento, por lo que resulta innovadora una funcionalidad con estas características.

1.6 TENDENCIAS Y TECNOLOGÍAS ACTUALES

Para comprender el proceso de compilación de una aplicación móvil es necesario conocer las diferentes consideraciones tecnológicas que existen, los sistemas operativos predominantes y los tipos de desarrollos que existen.

1.6.1 Sistemas Operativos

Un sistema operativo móvil también se conoce como SO móvil y es un conjunto de datos y programas que se ejecuta en una computadora o dispositivo móvil. Gestiona todo el hardware y optimiza la eficacia del software de aplicación en el dispositivo. Un sistema operativo móvil gestiona funciones multimedia móviles, conectividad móvil e Internet, etc.

Los sistemas operativos móviles se ven en teléfonos inteligentes, asistentes digitales personales (PDA), tabletas y dispositivos de información o dispositivos inteligentes que pueden incluir sistemas. El éxito definitivo de una plataforma móvil depende completamente de su adaptabilidad a las aplicaciones de terceros. (V.V. Subrahmanyam, 2011)

Android

Android es el líder en la actualidad de los sistemas operativos de los teléfonos inteligentes, tiene su origen en el sistema Linux. Su creación en un principio fue con el objeto de realizar acciones para cámaras fotográficas profesionales. El sistema fue vendido a Google quien realizó algunos ajustes y pudo adaptarlo a los dispositivos telefónicos. Es usado también en tabletas que son versiones de teléfonos inteligentes en formatos mayores. Los desarrolladores están buscando la forma de adaptarlos a los equipos PC de escritorio y ordenadores laptops. Quien llevan a cabo las actualizaciones y desarrollos de interfaz Android es la empresa Google. (González, 2020)

Creado por Andy Rubin en el año 2003, el sistema operativo lo adquirió Google en el 2005. La primera aparición tal cual como lo conocemos hoy fue en el año 2007, cuando algunos Smartphone comenzaban a tomar el mercado de los teléfonos móviles. Motorola y Samsung fueron los primeros en confiar sus dispositivos a Google para implementar ese sistema operativo. (González, 2020)

Android dispone de un kernel que permite de manera virtual realizar las acciones entre el software y el hardware. Lo revolucionario de este tipo de sistemas operativos móviles, es que permite a los usuarios realizar las acciones tocando la pantalla y dejando a un lado el sistema de teclas físicas. El kernel permitió establecer los códigos necesarios de Java para que las aplicaciones pudieran implementarse al momento de ser solicitadas por los usuarios. Esta forma de llevar a cabo las acciones permite al sistema Java tomar posesión de las aplicaciones y mostrarlas en la pantalla. (González, 2020)

Ha servido de referencia en las actualizaciones de otros sistemas operativos móviles. Las licencias con que cuenta Google permiten establecer modificaciones a los desarrolladores y fabricantes de celulares. De manera que pueden llevar a cabo algunas

variantes siempre y cuando permitan el avance y desarrollo de las acciones del sistema. (González, 2020)

iOS

Constituye el sistema operativo móvil para los dispositivos iPhone, el iPad, el iPod Touch y el sistema Apple TV. Consiste en un sistema sencillo el cual mantiene una cantidad de aplicaciones que mantienen a los usuarios encantados con su funcionamiento. Se considera la segunda empresa de sistemas operativos móviles más importantes después de Android. (González, 2020)

Es un sistema simple y muy fácil de operar. En el mundo existen millones de usuarios que buscan este tipo de dispositivos ya que la versatilidad permite otorgar diversas aplicaciones. El hardware es muy eficiente y las funciones se realizan con rapidez. Anualmente la empresa actualiza el sistema y se consiguen versiones que trae algún tipo de innovación. (González, 2020)

1.6.2 Tipos de aplicaciones según el desarrollo

Al igual que el desarrollo de aplicaciones web, el desarrollo de aplicaciones móviles tiene sus raíces en el desarrollo de software más tradicional. Sin embargo, una diferencia fundamental es que las aplicaciones móviles a menudo se escriben específicamente para aprovechar las características únicas de un dispositivo móvil en particular. (David, 2021)

En los primeros años de las aplicaciones móviles, la única forma de garantizar que una aplicación pudiera funcionar de manera óptima en cualquier dispositivo era desarrollar la aplicación de forma nativa. Esto significaba que se tenía que escribir un nuevo código específicamente para el procesador específico de cada dispositivo. Antes de desarrollar una aplicación, se debe determinar qué tipo creará. (David, 2021)

Aplicaciones Nativas

Estas aplicaciones están diseñadas para un único sistema operativo móvil. Por eso se denominan nativos: son nativos de una plataforma o dispositivo en particular. La mayoría de las aplicaciones móviles actuales están diseñadas para sistemas como Android o iOS.

En pocas palabras, no se puede instalar ni usar una aplicación de Android en iPhone o viceversa. (Herazo, 2019)

El principal beneficio de las aplicaciones nativas es su alto rendimiento y excelente experiencia de usuario (UX). El acceso a una amplia gama de API también ayuda a acelerar el trabajo de desarrollo y ampliar los límites del uso de la aplicación. Estas aplicaciones solo se pueden descargar de las tiendas de aplicaciones e instalarlas directamente en los dispositivos. Es por lo que primero deben pasar por un estricto proceso de publicación. (Herazo, 2019)

El inconveniente más importante de las aplicaciones nativas es su costo. Para crear, respaldar y mantener una aplicación para Android y iOS, básicamente se necesitan dos equipos de desarrollo, lo que puede hacer que el proyecto tenga más gastos. (Herazo, 2019)

Aplicaciones Web

Las aplicaciones web son aplicaciones de software que se comportan de manera similar a las aplicaciones móviles nativas y funcionan en dispositivos móviles. Sin embargo, existen diferencias significativas entre las aplicaciones nativas y las aplicaciones web. Para empezar, las aplicaciones web utilizan navegadores para ejecutarse y, por lo general, están escritas en CSS, HTML5 o JavaScript. (Herazo, 2019)

Dichas aplicaciones redirigen al usuario a la URL y luego les ofrecen la opción de instalarla. Simplemente crean un marcador en su página. Por eso requieren una memoria mínima del dispositivo. (Herazo, 2019)

Dado que todas las bases de datos personales se guardarán en el servidor, los usuarios solo pueden usar la aplicación si tienen una conexión a internet. Este es el principal inconveniente de las aplicaciones web: siempre requieren una buena conexión a internet. De lo contrario, corre el riesgo de ofrecer una experiencia de usuario insatisfactoria. Además, los desarrolladores no tienen tantas API que funcionen, a excepción de las funciones más populares, como la geolocalización. El rendimiento también estará vinculado al trabajo del navegador y la conexión de red. (Herazo, 2019)

Aplicaciones Híbridas

Estas aplicaciones se crean utilizando tecnologías web como JavaScript, CSS y HTML5. Las aplicaciones híbridas funcionan básicamente como aplicaciones web disfrazadas de un contenedor nativo. Son fáciles y rápidas de desarrollar, lo cual es un claro beneficio. También obtiene una única base de código para todas las plataformas, ello reduce el costo de mantenimiento y agiliza el proceso de actualización. (Herazo, 2019)

Los desarrolladores también pueden aprovechar muchas API para funciones como giroscopio o geolocalización. Por otro lado, las aplicaciones híbridas pueden carecer de velocidad y rendimiento. Además, es posible que experimente algunos problemas de diseño, ya que es posible que la aplicación no tenga el mismo aspecto en dos o más plataformas. (Herazo, 2019)

1.6.3 API REST

Una API de REST, o API de RESTful, es una interfaz de programación de aplicaciones (API o API web) que se ajusta a los límites de la arquitectura REST y permite la interacción con los servicios web de RESTful. El informático Roy Fielding es el creador de la transferencia de estado representacional (REST). (Hat, 2020)

Figura 1. Funcionamiento básico de una API REST.

Las API REST se comunican a través de solicitudes HTTP para realizar funciones estándar de base de datos, como crear, leer, actualizar y suprimir registros (también conocidos como CRUD) dentro de un recurso. Por ejemplo, una API REST utilizará una solicitud GET para recuperar un registro, una solicitud POST para crearlo, una solicitud PUT para actualizarlo y una solicitud DELETE para suprimirlo (Ver **Figura 1**). Todos los métodos HTTP se pueden utilizar en llamadas API. Una API REST bien diseñada es

similar a un sitio web que se ejecuta en un navegador web con funcionalidad HTTP incorporada. (Education, 2021)

1.7 FUNDAMENTACIÓN DE LA METODOLOGÍA UTILIZADA.

En el mundo del desarrollo de software existen variados métodos de desarrollo, cada uno con sus puntos fuertes y débiles. En el caso del desarrollo de aplicaciones móviles sucede lo mismo, y se debe saber escoger en función de las necesidades. (Vique, 2011)

Algunos de los métodos más conocidos son los siguientes:

- Modelo waterfall
- Desarrollo rápido de aplicaciones
- Desarrollo ágil (cualquiera de sus variantes)
- Mobile-D

Una de las características importantes de la gran mayoría de los desarrollos móviles es su corta duración. Esto se debe a factores como la gran competencia en el sector, los cambios en el mismo con la aparición, casi constante, de novedades tanto software como hardware, el hecho de que muchas aplicaciones nacen con un desarrollo precoz en forma de prototipo (y van evolucionado después) o incluso la simplicidad de las aplicaciones, que no requieren grandes desarrollos. Esta suele ser, salvo algunas excepciones, la norma de los desarrollos de aplicaciones para dispositivos móviles. (Vique, 2011)

1.7.1 Modelo waterfall

El modelo waterfall es el modelo más estático y predictivo. Es aplicable en proyectos en los que los requisitos están fijados y no van a cambiar durante el ciclo de vida del desarrollo. Esta aproximación divide el proyecto en fases estancas totalmente secuenciales como se puede apreciar en la **Figura 2**. En este modelo, el desarrollo se interpreta como el agua que va cayendo de un estanque al siguiente. Se le da mucho énfasis a la planificación, a los tiempos, a las fechas límite y al presupuesto (Vique, 2011). En el contexto del desarrollo de aplicaciones móviles, el modelo waterfall puede ser aplicable a proyectos realmente controlados y previsibles, en los que no hay mucha

incertidumbre por lo que se desea hacer y para los que no son importantes los cambios constantes en la industria. (Vique, 2011)

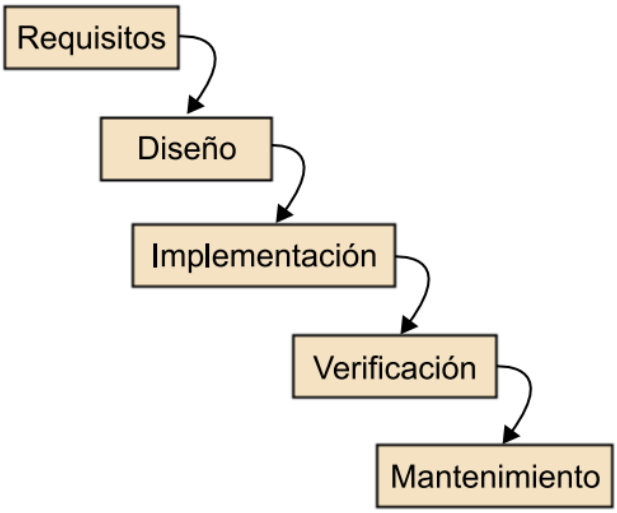

Figura 2. Ejemplo de fases de un proyecto de software en un modelo waterfall. (Vique, 2011)

1.7.2 Desarrollo rápido de aplicaciones

El desarrollo rápido de aplicaciones es un método de desarrollo iterativo cuyo objetivo es conseguir prototipos lo antes posible para mejorarlos después, poco a poco. Se suele priorizar la implementación sobre la planificación, y se utilizan muchos patrones de diseño conocidos para poder adaptarse de la mejor manera a cambios en los requerimientos. El desarrollo rápido de aplicaciones es un método muy útil para el desarrollo de proyectos realmente urgentes con tiempos de entrega muy cortos. (Vique, 2011)

1.7.3 Desarrollo ágil

El desarrollo ágil es un modelo de desarrollo basado en iteraciones, donde en cada iteración se realizan todas las fases del ciclo de desarrollo. (Vique, 2011)

El manifiesto ágil fue publicado en el 2001 por diecisiete desarrolladores de software, quienes representaban entonces los métodos de desarrollo más populares, que pasarían a conocerse como ágiles (Extreme Programming, Crystal Clear, DSDM o ASD, entre otros). El manifiesto define doce principios y cuatro valores éticos para los desarrolladores. El desarrollo ágil se basa en los principios del manifiesto ágil y sus

valores éticos, que tratan de dar más valor a algunos conceptos, pero sin dejar de lado los demás. (Vique, 2011)

- Dar más valor a los individuos y a sus interacciones que a los procesos y herramientas.
- Dar más valor al software que funciona que a la documentación exhaustiva.
- Dar más valor a la colaboración con el cliente que a la negociación contractual.
- Dar más valor a la respuesta al cambio que al seguimiento de un plan.

Con estos valores se intenta conseguir, entre otras cosas, entregar algo lo más pronto posible y evitar problemas originados por cambios de requisitos. Esto es muy apropiado para proyectos cambiantes, ya sean grandes o pequeños, ya que mediante estos valores se pueden mitigar los riesgos. Para conseguir proyectos que puedan cambiar fácilmente, se pone especial atención en la calidad de los productos conseguidos, cosa que es realmente importante en proyectos de software para dispositivos móviles. Para conseguir esto, se basan en las pruebas de la aplicación y, a menudo, las automatizan. (Vique, 2011)

Los métodos ágiles suelen ser muy adecuados para el desarrollo de aplicaciones móviles por las siguientes razones:

- Alta volatilidad del entorno: Con cambios en entornos de desarrollo, nuevos terminales y tecnologías a un ritmo mucho más elevado que en otros entornos de desarrollo.
- Equipos de desarrollo pequeños: Dado que los desarrollos móviles suelen ser proyectos relativamente pequeños, los equipos no suelen ser muy grandes. Generalmente son llevados a cabo por desarrolladores individuales o por PYME.
- Software no crítico: No suelen ser aplicaciones de alto nivel de criticidad, dado que suelen ser aplicaciones para entretenimiento o gestión empresarial no crítica.
- Ciclos de desarrollo cortos: Dada la evolución constante de la industria, se requieren ciclos de vida realmente cortos para poder dar salida a las aplicaciones a tiempo. (Vique, 2011)

1.7.4 Mobile-D

El método Mobile-D se desarrolló junto con un proyecto finlandés en el 2004. Fue realizado, principalmente, por investigadores de la VTT (Instituto de Investigación Finlandés) y, a pesar de que es un método antiguo, sigue en vigor (se está utilizando en proyectos de éxito y está basado en técnicas que funcionan). El objetivo es conseguir ciclos de desarrollos muy rápidos, en equipos muy pequeños (de no más de diez desarrolladores) trabajando en un mismo espacio físico. Según este método, trabajando de esa manera se deben conseguir productos totalmente funcionales en menos de diez semanas. Se trata de un método basado en soluciones conocidas y consolidadas: Extreme Programming (XP), Crystal Methodologies y Rational Unified Process (RUP), XP para las prácticas de desarrollo, Crystal para escalar los métodos y RUP como base en el diseño del ciclo de vida. (Vique, 2011)

Cada fase (excepto la inicial) tiene siempre un día de planificación y otro de entrega como se aprecia en la **Figura 3**. Las fases son:

- Exploración
- Inicialización
- Productización o Producción
- Estabilización.
- Pruebas y reparación

Figura 3. Ciclo de desarrollo de Mobile-D. (Vique, 2011)

1.7.5 Selección de la metodología de desarrollo

Luego de exponer las metodologías de desarrollo más utilizadas y orientadas a dispositivos móviles, se ha decidido optar por la metodología **Mobile-D** para definir el diseño e implementación de la aplicación. Dicha elección está justificada sobre las ventajas y facilidades que ofrece con relación a la cantidad de miembros del equipo de trabajo (pocos), la utilización de prácticas conocidas y consolidadas, la rapidez con la que se crea y se estabiliza la aplicación para tener un producto funcional que se podrá ir mejorando con cada iteración del ciclo de desarrollo. También cuenta con una gran comunidad en el mundo de los móviles sobre el cual está enfocada la metodología.

1.7.6 Marco de trabajo (framework)

Antes de desarrollar una aplicación, es importante tener en cuenta que existen diferentes tecnologías que se pueden emplear. Ya sea para crear una aplicación nativa, híbrida o multiplataforma, cada una de estas tecnologías tiene sus ventajas y desventajas, según la naturaleza del proyecto.

1.7.6.1 Nativo

El desarrollo móvil nativo implica la creación de aplicaciones adecuadas para trabajar en sistemas operativos para móviles particulares. Para lograr esto, existen tecnologías seleccionadas y lenguajes de programación que utilizan los desarrolladores. Por ejemplo, si se está buscando crear una aplicación para iOS, se necesitará aprender Objective-C o Swift. Por el contrario, los desarrolladores de Android confían en Java o Kotlin. (Luetić, 2021)

Android y iOS son los dos principales sistemas operativos móviles que se utilizan en la actualidad y para desarrollar aplicaciones nativas, los desarrolladores de Android y iOS utilizan un conjunto específico de herramientas de desarrollo llamado Software Development Kit (SDK) y un Integrated Development Environment (IDE). (Luetić, 2021)

Debido a que están dirigidas específicamente a sistemas operativos predeterminados, las aplicaciones nativas son generalmente más seguras, intuitivas, funcionan mejor y otorgan a los desarrolladores acceso completo a las funciones del dispositivo de destino. Sin embargo, el desarrollo nativo también suele ser más intensivo en capital tanto inicialmente como después del lanzamiento. (Luetić, 2021)

Android: Android Studio

En el desarrollo de Android, el IDE oficial y más utilizado por los desarrolladores es Android Studio. La creación de aplicaciones de Android con Android Studio significa que no solo se obtiene un potente editor de código respaldado por interesantes herramientas para desarrolladores, sino que también se tiene acceso a funciones avanzadas que potencian la experiencia del desarrollador. (Luetić, 2021)

Algunas de estas características incluyen:

- Un emulador repleto de funciones que ofrece un gran rendimiento.
- Un sistema de construcción flexible basado en Gradle.
- Amplia gama de frameworks y herramientas de prueba.
- Soporte para la importación de código de muestra y características de aplicaciones reutilizables a través de plantillas de código e integración de GitHub.

- Integración de Google Cloud Platform.
- Compatibilidad con C ++ y Native Development Kit (NDK).

Otra ventaja de desarrollar con Android Studio es que, como IDE, permite crear aplicaciones nativas en prácticamente cualquier sistema operativo, ya sea Windows, Mac o Linux. Sin embargo, para hacerlo, se necesitará instalar un kit de desarrollo de Java (JDK). Entre otras herramientas de desarrollo, un JDK incluye Java Runtime Environment (JRE), un intérprete de código (Java) y un compilador (javac). Con el JDK configurado, se podrá instalar Android Studio y comenzar a construir la aplicación. (Luetić, 2021)

Cuando se trata de la elección del marco de trabajo, Android Studio es una elección común entre los desarrolladores de Android. Teniendo en cuenta su amplia oferta de funciones, no es difícil ver los motivos.

Algunas de las aplicaciones nativas más populares creadas con Android Studio incluyen WhatsApp Messenger, LinkedIn, Netflix, Evernote, Uber, etc.

<u>Ventajas</u>

- Software de código abierto de uso gratuito.
- Admite la traducción automática de Java-Kotlin.
- Diseño de editor intuitivo e interfaz flexible (admite la edición de temas).
- Eficiente actualización de las bibliotecas.
- Excelente modo de depuración.
- Gran apoyo de la comunidad de desarrolladores.
- Desarrollo más seguro.
- Desarrollo intuitivo.
- Mejor presentación.
- Otorga a los desarrolladores acceso completo a las funciones del dispositivo de destino.

<u>Desventajas</u>

- Gran demanda de recursos de hardware.
- El emulador se inicia con bastante lentitud.

- Manejo de base de código dedicada solo para la plataforma Android.(Luetić, 2021)

iOS: XCode

XCode es el equivalente directo de Android Studio. Es el IDE oficial de Apple y tiene una larga historia en el mundo del desarrollo de aplicaciones. Dado que es nativo para el desarrollo de iOS, es compatible con Objective-C, Swift.

XCode incluye prácticamente todo lo que se necesita para desarrollar una aplicación iOS: compiladores, depuradores, editores de Storyboard y XML Interface Builder (XIB).

En el estilo propietario habitual de Apple, algunas funcionalidades de creación de aplicaciones de iOS están bloqueadas exclusivamente en XCode. Por ejemplo, incluso con plataformas no nativas como Flutter o React Native, es imposible editar guiones gráficos sin XCode. Además, otros IDE alternativos aún requieren que tenga XCode instalado. (Luetić, 2021)

Algunas de las características principales:

- Amplia compatibilidad con lenguajes de programación: C, C ++, Objective-C, Java, Python, AppleScript, etc.
- Utilidad para construir archivos binarios pesados, incluido código compatible con diferentes arquitecturas en formato ejecutable Mach-O.
- Compilación y depuración de aplicaciones iOS en procesadores de arquitectura ARM.
- Herramienta de interfaz gráfica de usuario (GUI) que facilita el diseño y la creación de prototipos de la interfaz de usuario sin código.
- Filtro de código, editor de versiones, asistente de codificación, control de fuente, etc.

Cada una de estas características viene con un gran impacto que posiciona a XCode como su ventanilla única para todo el desarrollo de iOS. La biblioteca de ayuda incluye varios recursos, incluida la documentación del SDK, referencias de API, código de muestra, guías de codificación y casi todo lo que un desarrollador necesita para

solucionar problemas de manera efectiva. Toda esta información está disponible en el visor de documentación. (Luetić, 2021)

La mayoría de las aplicaciones nativas de iOS se crean con XCode, por ejemplo: Firefox, LinkedIn, SlideShare, Asana, Lyft y WordPress.

Ventajas

- Una interfaz fluida y fácil de usar que hace que la creación de la interfaz de usuario sea perfecta.
- El emulador imita a un iPhone y permite probar la aplicación mientras la construye.
- Excelente función de finalización de código (ahorro de tiempo para desarrolladores de todos los niveles).
- Amplia documentación.
- Disponibilidad de múltiples esquemas de trabajo.
- Desarrollo más seguro.
- Desarrollo intuitivo.
- Mejor presentación.
- Otorga a los desarrolladores acceso completo a las funciones del dispositivo de destino.

Desventajas

- Exclusivo solo para los sistemas operativos de Apple.
- Sin soporte para entornos de trabajo con pestañas.
- Exportar la aplicación a un dispositivo directamente es bastante complicado.
- Manejo de una base de código dedicada solo para la plataforma iOS. (Luetić, 2021)

1.7.6.2 Multiplataforma

Las aplicaciones multiplataforma se caracterizan por ser creadas bajo un único lenguaje de programación que facilita su exportación y por tanto su visualización en cualquier tipo de dispositivo independientemente de su sistema operativo. Al ser desarrolladas con un

mismo lenguaje, sólo son necesarios unos cambios mínimos para su completa adaptación a cualquier dispositivo, ya sea móvil, ordenador o tablet, entre otros.

El desarrollo multiplataforma es muy cómodo para los profesionales ya que en el lado contrario se pueden encontrar las aplicaciones nativas. Estas aplicaciones móviles se desarrollan por separado para cada sistema operativo, cada una tiene su propio lenguaje de programación. Aunque tienen un muy buen rendimiento y un gran nivel de personalización, cada proyecto y cada idea necesitan ser analizados para saber cuál es la mejor opción.

Flutter

En general, Flutter es un marco de trabajo móvil versátil, de código abierto que se basa en el lenguaje de programación de Google, Dart. Este lenguaje de programación es un componente muy crítico de la arquitectura de Flutter. Dart se compila Ahead-Of-Time (AOT) en código nativo para varias otras plataformas. Esta capacidad de eliminar la necesidad de un puente de JavaScript (como en React Native) le da a Flutter una ventaja significativa en términos de tiempos de inicio y rendimiento general.

Algunas de las características distintivas que se puede esperar al usar Flutter incluyen: *Hot Reload* (actualización inmediata en cambios del código), jerarquía de *Widgets* para componentes e integración de código (facilita la interoperabilidad con código nativo). Las principales ventajas que ofrece es el buen rendimiento en relación con otros frameworks multiplataforma y una curva de aprendizaje muy suave para principiantes. Sin embargo, las aplicaciones suelen ocupar mucho espacio de almacenamiento, en comparación con otras tecnologías populares, es posible que sea necesario acostumbrarse a Dart (lenguaje de programación) y que las aplicaciones nativas son 10 veces más rápidas que las aplicaciones de Flutter. (Luetić, 2021)

React Native

React Native es la alternativa de Facebook a Flutter de Google. Se basa en una poderosa biblioteca de JavaScript llamada React. El marco de trabajo utiliza la funcionalidad de pequeños fragmentos de código denominados “componentes” para crear una interfaz de usuario de aplicación móvil compleja y completa. Estos componentes de React son

reutilizables, lo que hace que el desarrollo sea más rápido y menos estresante con rendimiento casi nativo. También es una tecnología de código abierto y no solo es compatible con aplicaciones de Android y iOS, sino también con aplicaciones web. Otra ventaja importante que presenta React Native es su dependencia de un lenguaje de programación ampliamente utilizado, JavaScript. (Luetić, 2021)

Como desventajas se destacan que el desarrollo de ciertos módulos específicos de la plataforma aún requiere un componente nativo y los problemas de compatibilidad y depuración son bastante comunes. (Luetić, 2021)

La ejecución de la aplicación móvil de React Native se puede ver en aplicaciones como Artsy, Bloomberg y Delivery.com.

Ionic

Con un framework como Ionic, se pueden crear aplicaciones móviles híbridas utilizando tecnologías de desarrollo web existentes. Específicamente, los desarrolladores usan códigos únicos escritos en tecnologías front-end como Javascript, Angular, CSS y HTML. Se basa en Apache Cordova, que implementa la parte nativa de la aplicación. La creación de prototipos es rápida, sencilla con una amplia variedad de elementos de la interfaz de usuario y presenta una gran documentación y comunidad. La aplicación móvil resultante no está completamente basada en la web ni es completamente nativa. (Luetić, 2021)

Es importante tener en cuenta que el uso Ionic se quedan detrás en el rendimiento, en iOS puede no ofrecer una experiencia pulida como en Android y las aplicaciones suelen ser pesadas ya que requieren una gran cantidad de código en la integración de complementos. (Luetić, 2021)

Las aplicaciones híbridas populares desarrolladas con el marco de trabajo de Ionic incluyen Sanvello, TD Trading, Sworkit, MarketWatch, etc.

1.7.6.3 Selección del marco de trabajo

Para la elección del marco de trabajo a utilizar, se hace necesario identificar primero el tipo de marco trabajo (nativo o multiplataforma) que responda mejor a la implementación del software. Para esto se ha decidido un desarrollo **nativo** que potencie el rendimiento

de la aplicación, permita el acceso a todas las funcionalidades del dispositivo sin depender de paquetes de terceros y ofrezca al usuario final una mejor experiencia.

Luego se tienen a disposición los frameworks nativos de Android y iOS. Basado en las estadísticas de utilización de los dispositivos móviles en el mundo y en Cuba con respecto a sus sistemas operativos, Android se muestra como líder del mercado. El dominio de Android en la industria, la licencia Open Source que mantiene, la inmensa comunidad de la cual dispone, el rendimiento que brinda su sistema y el gran soporte que ofrece Google como encargado de la tecnología, hacen que **Android** sea una elección segura para desarrollar la aplicación para el sistema de reservaciones en línea.

1.7.7 Entorno de desarrollo

Las actividades más comunes de la escritura de software son la edición de código fuente, la creación de ejecutables y la depuración. Un entorno de desarrollo integrados (IDE) es un conjunto de aplicaciones donde puede realizar todas esas actividades en un solo lugar. Puede realizar todo el ciclo de vida del desarrollo de software en un IDE. Como resultado, se han creado muchos entornos diferentes para el desarrollo de aplicaciones móviles. Están surgiendo muchos nuevos y los antiguos están evolucionando. (Melnichuk, 2020)

Android no se queda atrás y es soportado por muchos entornos de desarrollo nativo donde se mencionan los más relevantes a continuación.

Eclipse

Según PYPL Index, el IDE de código abierto de Eclipse es el segundo más popular del mundo, durante varios años, una versión de Eclipse con un complemento de Android (ADT) fue el entorno de desarrollo recomendado para la plataforma Android. Aunque Google ha dejado de admitir ADT, muchos desarrolladores de dispositivos móviles continúan usando Eclipse para crear aplicaciones de Android.

IntelliJ IDEA

El sitio web de IntelliJ IDEA lo describe como un "IDE capaz y ergonómico para JVM". Tiene numerosas funciones para el desarrollo de Android y proporciona la base para

Android Studio. Este IDE también ha ganado numerosos premios y PYPL Index lo clasifica como el sexto más popular, justo detrás de NetBeans. Viene tanto en una edición "Community" gratuita de código abierto como en una edición "Ultimate" de pago.

Android Studio

Es el IDE oficial de Google para Android, que se basa en Intellij IDEA con funciones diseñadas a medida para Android. Reemplazó las anteriores herramientas de desarrollo de Android (ADT) basadas en Eclipse como el IDE preferido para la plataforma. Según PYPL Index, Android Studio es el tercer IDE más popular del mundo con una participación del 9,8%. Es la opción predeterminada para los desarrolladores de Android y, como oferta de Google, se integra fácilmente con Google Cloud Platform. (Melnichuk, 2020)

Selección del entorno de desarrollo

El entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android es el **Android Studio**. Es el indicado para la implementación de la aplicación de reservas en línea, ya que, además tener el potente editor de códigos y herramientas para desarrolladores de IntelliJ, Android Studio ofrece incluso más funciones que aumentan la productividad (algunas exclusivas del IDE) como las siguientes:

- Un sistema de compilación flexible basado en Gradle.
- Un emulador rápido y cargado de funciones.
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android.
- Aplicación de cambios para insertar cambios de código y recursos a la app en ejecución sin reiniciarla.
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de aplicaciones comunes y también importar código de muestra.
- Variedad de marcos de trabajo y herramientas de prueba.
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros.
- Compatibilidad con C++ y NDK.

- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google. Cloud Messaging y App Engine. (Google, 2021c)

1.7.8 Lenguaje de programación

Las diferencias entre los distintos lenguajes de programación de Android pueden ser un poco complejas y matizadas. Pero si se trata de un desarrollo nativo sin tener que “bajar” a la capa de C++ que contiene Android, pues entonces quedan dos opciones fundamentales, Java y Kotlin, con las cuales se crean la mayoría de las aplicaciones de la plataforma.

Java

Cuando llega el momento de desarrollar aplicaciones de Android, Java sigue siendo una de las dos opciones oficiales, incluso cuando Google ha hecho evidente su preferencia por Kotlin, Java está tan arraigado y es tan familiar que muchos equipos de desarrollo han optado por seguir con él. (Sims, 2019)

Desafortunadamente, Java también es complicado y no es un gran “primer idioma”. Es un lenguaje de programación orientado a objetos con temas confusos como constructores, excepciones de puntero nulo, excepciones comprobadas y más. No es muy legible y se usará mucho código repetitivo para hacer cosas simples. El desarrollo que utiliza esta ruta también requiere una comprensión básica de conceptos como Gradle, el Manifiesto de Android y el lenguaje de marcado XML. Según la tabla PYPL (Popularidad de lenguajes de programación), Java es el lenguaje de programación más buscado entre los empleadores. (Sims, 2019)

Kotlin

Kotlin ha sido un lenguaje oficial para el desarrollo de Android desde hace un tiempo, y Google incluso ha ido tan lejos como para convertirlo en la opción preferida para el desarrollo de Android. (Sims, 2019)

Al igual que Java, Kotlin se ejecuta en la máquina virtual de Java (JVM). También es completamente interoperable con Java y no ralentiza ni aumenta el tamaño de los archivos. La diferencia es que Kotlin requiere menos código estándar, lo que significa

que es un sistema optimizado y fácil de leer. También elimina errores como las excepciones de punto nulo e incluso le exime de terminar cada línea con punto y coma. Es genial si se está aprendiendo a desarrollar aplicaciones de Android. (Sims, 2019)

Selección del lenguaje de programación

Java fue durante mucho el lenguaje preferido de la mayoría de los desarrolladores para Android, pero Kotlin fue creado con el fin específico de reemplazarlo y ser el lenguaje de programación oficial de la plataforma. En este proceso que corrigió muchos defectos de Java, añadió increíbles funcionalidades como las corrutinas (coroutines) para el trabajo asíncrono. Ayuda a aumentar la productividad, la satisfacción de los desarrolladores y la seguridad del código. Es expresivo y conciso ya que gracias a las modernas funciones que posee, se podrá enfocar en expresar las ideas y escribir menos líneas de código estándar. Contribuye a un código más seguro al incluir las anotaciones @Nullable y @NonNull para ayudar a evitar excepciones del tipo NullPointerExceptions. Las aplicaciones para Android que usan Kotlin prometen tener un 20% menos de probabilidades de fallar. Es 100% interoperable con el lenguaje de programación Java, por lo que se puede implementar en los proyectos en la medida que se requiera.

Estas características además del rápido crecimiento de del lenguaje y la comunidad, hacen que **Kotlin** sea la mejor opción para utilizar como lenguaje de programación en el desarrollo de la aplicación.

1.7.9 Base de Datos

La base de datos (DB) es la forma más común de almacenar y administrar datos. Desde hace bastante tiempo, las bases de datos se manejan en el lado del servidor o en la nube y los dispositivos móviles solo se comunican con ellos a través de la red. Sin embargo, para hacer que las aplicaciones sean más receptivas y menos dependientes de la conectividad de la red, la tendencia del uso fuera de línea o la menor dependencia de la red está ganando popularidad. Hoy en día, las aplicaciones mantienen la base de datos localmente o hacen una copia en la nube y se sincronizan con ella una vez al día o cada vez que hay una conectividad de red.

En Android hay varias posibilidades para trabajar este tipo de recurso y se puede utilizar un sistema embebido como SQLite u otro como Room.

SQLite

Si bien estas API de SQLite son potentes, se caracterizan por ser bastante específicas y su uso requiere de mucho tiempo y esfuerzo. No hay verificación en tiempo de compilación de las consultas de SQL sin procesar. A medida que cambia el grafo de datos, se debe actualizar manualmente las consultas de SQL afectadas. Este proceso puede llevar mucho tiempo y causar errores. Se debes usar mucho código estándar para convertir entre consultas de SQL y objetos de datos. (Google, 2021e)

Por estos motivos, se recomienda enfáticamente usar la biblioteca de persistencias Room como una capa de abstracción para acceder a la información de las bases de datos SQLite en una aplicación Android. (Google, 2021e)

Room

Room proporciona una capa de abstracción sobre SQLite que permite acceder a la base de datos sin problemas y, al mismo tiempo, aprovechar toda la potencia de SQLite. (Google, 2021d)

Las aplicaciones que controlan grandes cantidades de datos estructurados pueden beneficiarse con la posibilidad de conservar esos datos localmente. El caso práctico más común es almacenar en caché datos relevantes. De esa manera, cuando el dispositivo no puede acceder a la red, el usuario de todos modos puede explorar ese contenido mientras está desconectado. Cualquier cambio de contenido iniciado por el usuario se sincroniza con el servidor una vez que el dispositivo vuelve a estar en línea. (Google, 2021d)

Selección del tipo de Base de Datos

Room facilita el uso de los distintos elementos de una base de datos, es la recomendación de Google como empresa a cargo de la plataforma, respeta los patrones de diseño más utilizados como el *Dao* y el *Repositorio*, los cuales son parte fundamental en la arquitectura de cualquier aplicación Android moderna.

Para la elección del tipo de base de datos a utilizar se analiza las ventajas que ofrece cada tecnología y las recomendaciones de la comunidad. En este caso **Room** resulta la mejor opción para la aplicación de reservaciones que se pretende desarrollar.

1.8 CONCLUSIONES PARCIALES

Las consideraciones teóricas planteadas y el impacto de las nuevas tecnologías en el sector de los negocios promueven la necesidad de una aplicación móvil, que complemente el sistema de reservas en línea de la empresa XETID. Se pretende diseñar e implementar para su despliegue con carácter nacional y que se adapte a las características de la sociedad cubana.

Se analizó el impacto de las aplicaciones móviles en los negocios y sectores de la sociedad, así como el estado actual de las reservaciones en línea en el ámbito nacional e internacional.

La metodología de desarrollo seleccionada para regir todo el proceso de confección de la aplicación fue Mobile-D, se optó por el marco de trabajo nativo que brinda el entorno de desarrollo Android Studio, utilizando a Kotlin como lenguaje de programación principal.

2 CAPÍTULO II. ANÁLISIS Y DISEÑO DE LA APLICACIÓN

En este capítulo se documentan los artefactos de la aplicación *ticket*, de acuerdo con las fases de la metodología ágil Mobile-D, metodología seleccionada para la concepción y desarrollo de la aplicación. Se describen los *storyboards*[1] y *storycards*[2], los cuales representan la principal documentación de la metodología Mobile-D, estos muestran visualmente cada pantalla de la aplicación. Se confecciona el mapa de navegación, el cual permite conocer cómo acceder a cada pantalla. Se describen los métodos y clases de más interés de la implementación.

2.1 EXPLORACIÓN

La fase de Exploración se corresponde a la fase de Análisis dentro de la metodología Mobile-D.

En la primera fase (Exploración) el equipo de desarrollo debe generar un plan y establecer las características del proyecto. Esto se realiza en tres etapas: establecimiento de actores, definición del alcance y el establecimiento de proyectos. Las tareas asociadas a esta fase incluyen el establecimiento del cliente (los clientes que toman parte activa en el proceso de desarrollo), la planificación inicial del proyecto y los requisitos de recogida, y el establecimiento de procesos. (Rafael, 2018)

2.1.1 Establecimiento de los Grupos de Interés

Se refiere a las personas o grupos afectados por un proyecto de desarrollo de software. Las partes interesadas existen tanto dentro como fuera de la organización. Pueden ser usuarios finales o simplemente pueden verse afectados por el proceso. De cualquier manera, tienen intereses creados en el producto final. Los aportes de las partes

[1] Serie de diagramas que muestra una secuencia de visualizaciones. Es un documento de planificación. Está creado antes de que se desarrolle el producto final y se utiliza para ilustrar una historia o mostrar el cambio de escenario. OCR 2013. The OCR Guide to Storyboards. *In:* OCR (ed.). OCR.

[2] El propósito de una *storycard* es escribir cómo un proyecto devolverá valor al usuario. Entonces es el trabajo del equipo de desarrollo ocuparse de cómo desarrollar el código que satisfaga los requisitos de la historia del usuario. En el mejor de los casos, los desarrolladores colaboran estrechamente con los propietarios de negocios y las partes interesadas para aclarar los detalles a medida que se desarrolla el código. MUÑOZ, C. A. M. 2020. *Aplicación de la metodología Mobile-D en el desarrollo de una app móvil para gestionar citas médicas del Centro JEL Riobamba.* Universidad Nacional de Chimborazo.

interesadas le dicen a la empresa qué tipo de software se necesita, sugiriendo ideas para funciones o problemas que necesita resolver. (Technologies, 2018)

Desarrollador: Será el encargado de todo el proceso de creación del producto (análisis, diseño e implementación).

Proveedores de servicio: Persona natural o jurídica que pretende publicar algún tipo de negocio en el sistema.

Usuarios: Toda aquella persona (externa) que necesita realizar reservaciones en algún tipo de establecimiento.

2.1.2 Definición de los requisitos

El término *requerimientos* (o requisitos) no se utiliza de forma consistente en la industria de software. En algunos casos, un requerimiento se visualiza como una declaración abstracta de alto nivel de un servicio que debe proveer el sistema o como una restricción de este. Por otro lado, es una definición matemática detallada y formal de una función del sistema. (Sommerville, 2001)

A menudo, los requerimientos de sistemas de software se clasifican en funcionales y no funcionales.

Requisitos funcionales

Un requisito funcional define una función del sistema de software o sus componentes. Una función es descrita como un conjunto de entradas, comportamientos y salidas. Los requisitos funcionales pueden ser: cálculos, detalles técnicos, manipulación de datos y otras funcionalidades específicas que se supone, un sistema debe cumplir. Los requisitos de comportamiento para cada requisito funcional se muestran en los casos de uso. (Roger S. Pressman, 2010)

Los requisitos funcionales para la aplicación móvil *ticket* se especifican en la **Tabla 1**.

Tabla 1. Requisitos funcionales de ticket.

NOMBRE	BREVE DESCRIPCIÓN

RF1 Autenticar usuarios	La aplicación debe permitir registrar nuevos usuarios, así como iniciar sesión a los que ya posean una cuenta en el sistema.
RF2 Gestionar perfil	Se debe permitir consultar y editar los datos personales del perfil de usuario.
RF3 Consultar Servicios	Se hace necesario listar los servicios y ver detalladamente los mismos.
RF4 Consultar Establecimientos	Se hace necesario listar los establecimientos y ver detalladamente los mismos.
RF5 Gestionar tickets	Un aspecto fundamental de la aplicación es obtener tickets (la reservación) para los servicios disponibles. Es requerido listar los tickets obtenidos, consultar el detalle y el historial tickets del usuario.
RF6 Gestionar Salas de Espera	La aplicación debe permitir entrar a salas de espera para servicios que se hayan quedado sin disponibilidades. Se requiere poder listar las salas de espera, así como poder salir de ellas para liberar la posición a otro usuario.
RF7 Pagar tickets	La aplicación debe posibilitar el pago de tickets para aquellos servicios que lo requieran, utilizando las plataformas de pago existentes en el sistema.
RF8 Valorar servicios	Se requiere la opción de opinar y valorar los servicios por parte de los usuarios.
RF9 Gestionar notificaciones	El sistema debe permitir recibir notificaciones, así como eliminarlas.

Requisitos no funcionales

Un requisito no funcional o atributo de calidad es, en la ingeniería de sistemas y la ingeniería de software, un requisito que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos, ya que estos corresponden a los requisitos funcionales. Por tanto, se refieren a todos los requisitos que no describen información a guardar, ni funciones a realizar, sino características de funcionamiento. Por esto, suelen denominarse atributos de calidad de un sistema. Pueden tratarse también como las restricciones o condiciones que impone el cliente al programa que necesita. (Stellman, 2005)

En la **Tabla 2** se especifican los requisitos no funcionales para *ticket* que se comprenden en usabilidad, confiabilidad, localización, soporte y restricciones de diseño.

Tabla 2. Requisitos no funcionales de ticket.

USABILIDAD	
RNF1 Diseño Visual	La aplicación deberá poseer un diseño robusto, fácil de entender por usuarios de todas las edades, por lo que tendrá interfaces visuales sencillas y manteniendo viejos estándares de la usabilidad.
CONFIABILIDAD	
RNF2 Tiempo de Accesibilidad	La aplicación deberá estar disponible las 24hrs del día, posibilitando a sus usuarios poder acceder al sistema cuando se requiera.
LOCALIZACIÓN	
RNF3 Multilenguaje	Se requiere la utilización de al menos dos idiomas, español e inglés, para una mejor

	interacción y adaptación de los usuarios al sistema.
SOPORTE	
RNF4 Software y Hardware	La aplicación requerirá un Smartphone con sistema operativo Android con versiones igual o posteriores a la 5.0 y conexión WI-FI o de datos móviles a la red nacional de Cuba.
RESTRICCIONES DE DISEÑO	
RNF5 Servidor Web	Se utilizará la conexión a un servidor remoto mediante el uso de API REST.
RNF6 Herramienta CASE	Se utilizará la herramienta draw.io para el diseño de los diagramas requeridos.

2.1.3 Definición del Alcance

El alcance del proyecto se determina a través de las limitaciones, supuestos y dependencias.

Limitaciones

- La aplicación sólo puede ser ejecutada en dispositivos con el Sistema Operativo Android en versiones igual o mayores a la 5.0 (Lollipop).
- Para su utilización se hace necesario de una conexión (Wi-Fi o Datos móviles) constante con la API REST que se aloja en el servidor del sistema.

Supuestos y dependencias

- La aplicación está concebida para su uso por visitantes tanto extranjeros como nacionales, por lo que la interfaz de la aplicación es mostrada en dos idiomas: español e inglés, en dependencia del idioma establecido en el dispositivo móvil.

2.2 INICIALIZACIÓN

La Inicialización en Mobile-D comprende la fase de Diseño de producto, es donde se preparan e identifican todos los recursos necesarios. Se establece el entorno técnico que permite el éxito de las próximas fases del proyecto. (Vique, 2011)

2.2.1 Diseño del sistema

El sistema se compone primeramente del cliente móvil (dispositivo Android) que se conecta a través de la red nacional o internet un servicio web (API REST). Es te servicio web está compuesto por una arquitectura orientada a microservicios, motivos por el cual consume a su vez otras API REST. Estos servicios web se alojan en el servidor del sistema, así como los servidores de base de datos. (Ver **Figura 4**)

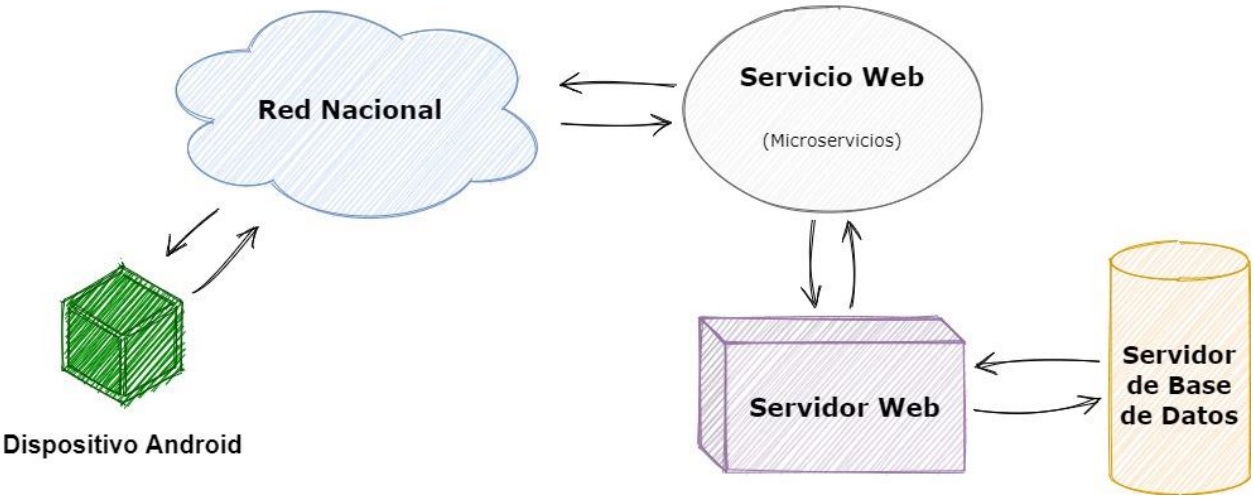

Figura 4. Arquitectura del sistema ticket.

2.2.2 Diseño de la aplicación

En la mayoría de los casos, las aplicaciones de escritorio tienen un solo punto de entrada desde un escritorio o un selector de programas y, luego, se ejecutan como un solo proceso monolítico. Por otro lado, las aplicaciones para Android tienen una estructura mucho más compleja. Una aplicación Android típica consta de varios componentes, actividades, fragmentos, servicios, proveedores de contenido y receptores de emisión. (Google, 2021a)

La mayoría de estos componentes se declaran en el manifiesto de la aplicación. Luego, el Sistema Operativo utiliza ese archivo para decidir cómo integrar la aplicación a la experiencia del usuario general del dispositivo. Debido a que una aplicación de Android escrita correctamente incluye varios componentes y dado que los usuarios suelen interactuar con varias apps en un período breve, las aplicaciones deben adaptarse a distintos tipos de tareas y flujos de trabajo controlados por los usuarios. (Google, 2021a)

Arquitectura MVVM

La arquitectura MVVM es una arquitectura Model-View-ViewModel que elimina el estrecho acoplamiento entre cada componente. Lo más importante es que en esta arquitectura, los hijos no tienen la referencia directa al padre, solo tienen la referencia de los observables. (Shekhar, 2020)

Model (Modelo): Representa los datos y la lógica empresarial de la Aplicación Android. Consiste en la lógica empresarial: fuente de datos local y remota, clases de modelo, repositorio.

View (Vista): Consta del código de la interfaz de usuario (actividad, fragmento), XML. Envía la acción del usuario al ViewModel pero no obtiene la respuesta directamente. Para obtener la respuesta, tiene que suscribirse a los observables que ViewModel le expone. (Shekhar, 2020)

ViewModel (Vista de Modelo): Es un puente entre la Vista y el Modelo (lógica de negocios). No conoce qué Vista tiene que usarlo, ya que no tiene una referencia directa a la Vista. Entonces, básicamente, ViewModel no debe ser consciente de la vista con la que está interactuando. Interactúa con el Modelo y expone lo observable que puede ser observado por la Vista. (Shekhar, 2020)

Para la aplicación Android en desarrollo se pretende utilizar el diseño que se expone en la **Figura 5** y que a su vez es respaldado por el patrón de arquitectura MVVM ya que se adapta perfectamente a los requisitos obtenidos en la fase de Exploración y al marco de trabajo que provee la plataforma Android.

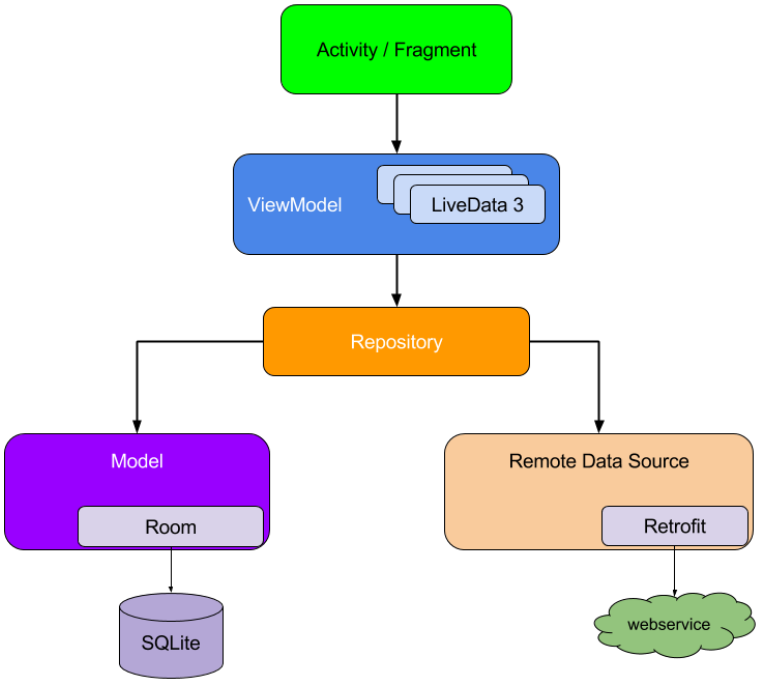

Figura 5. Arquitectura de la aplicación ticket. (Google, 2021a)

2.2.3 Diseño de la Base de Datos

El modelo Entidad-Relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos. Es una técnica de análisis para definir las entidades y las relaciones entre ellas. A continuación, en la **Figura 6**, se muestra el esquema lógico diseñado para la base de datos la aplicación *ticket* la cual representa a una de las fuentes de información y será implementada mediante la biblioteca Room.

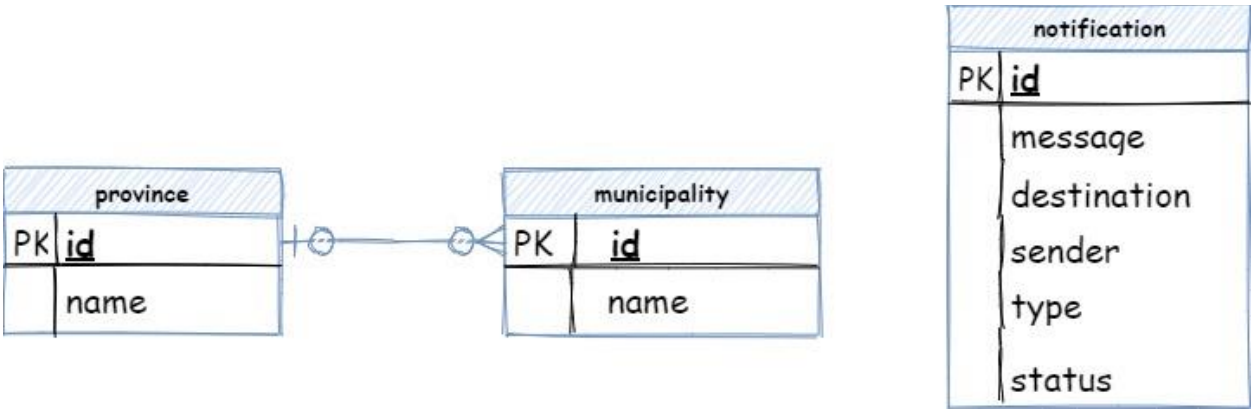

Figura 6. Modelo lógico de los datos para ticket.

2.2.4 Diagrama de componentes

El diagrama de componentes muestra las organizaciones y dependencias lógicas entre componentes de software, sean estos componentes de código fuente, binarios o ejecutables (Ver **Figura 7**). Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, y las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. (Sommerville, 2001)

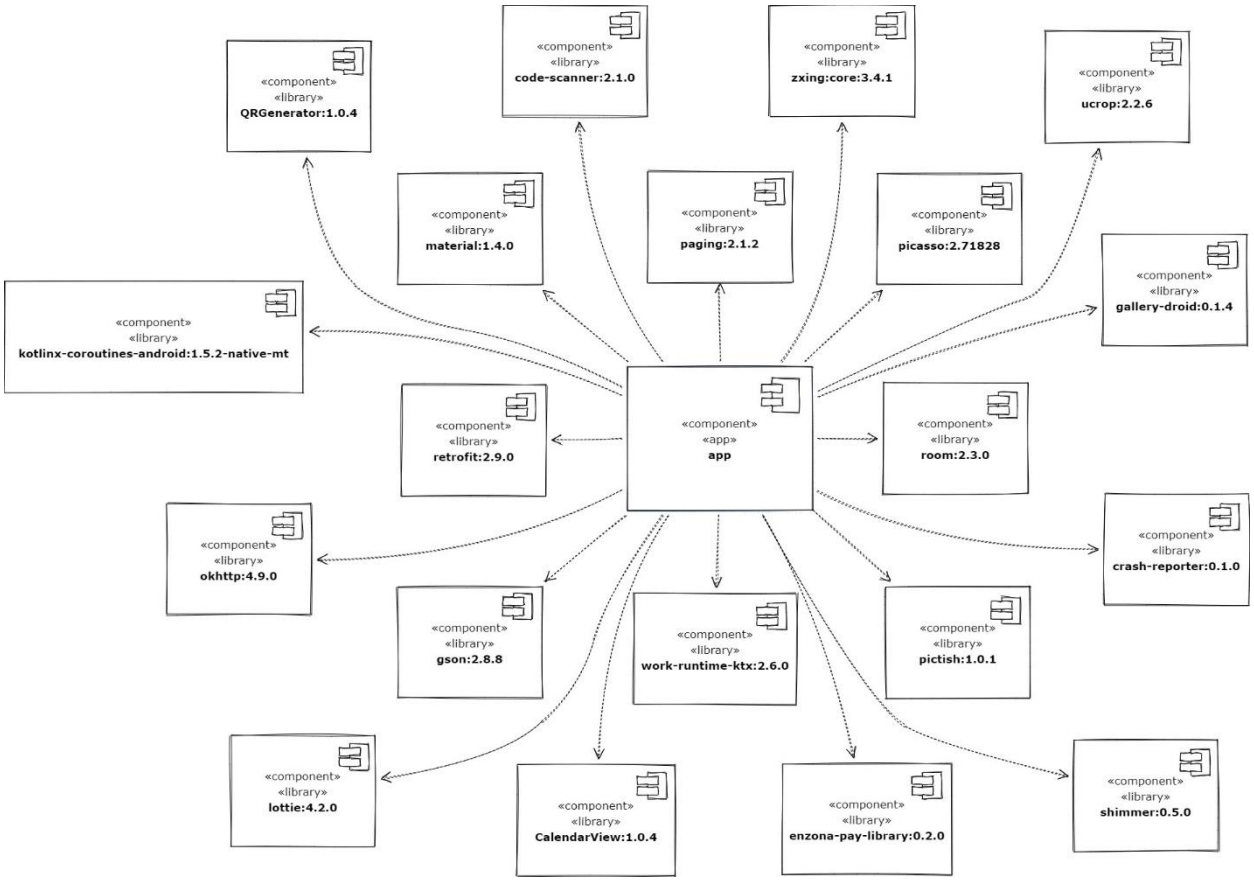

Figura 7. Diagrama de componentes de la aplicación ticket.

2.2.5 Interfaz de Usuario

A continuación, en la **Figura 8**, se describe el *storyboard* o mapa de navegación y los vínculos entre las distintas pantallas que conforman la aplicación.

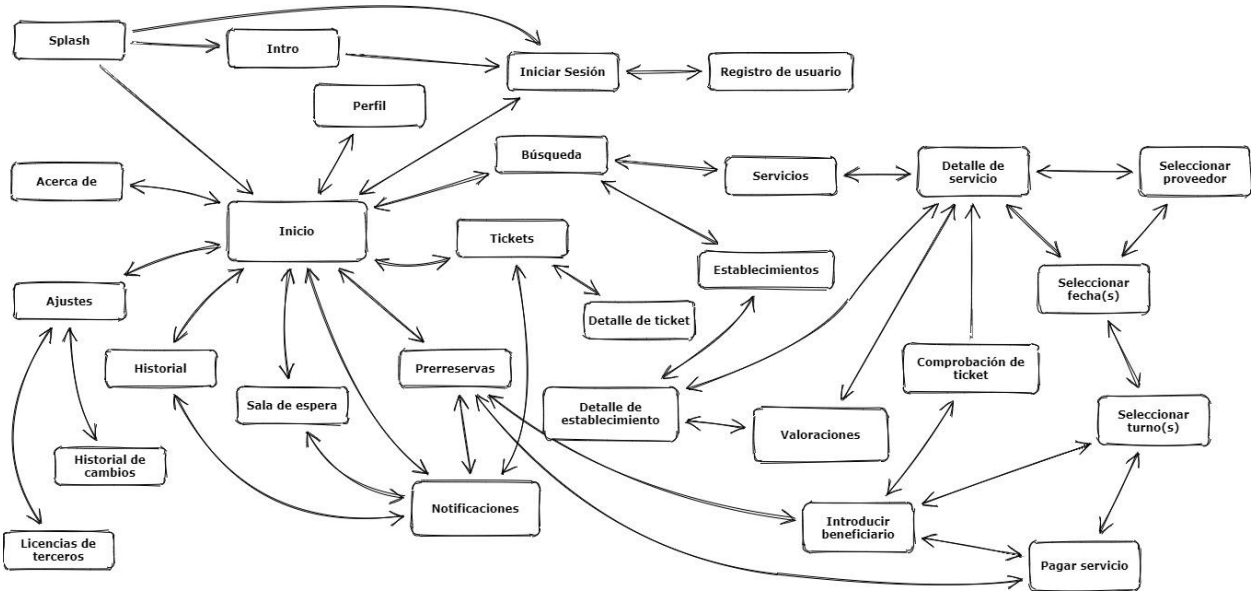

Figura 8. Storyboard de la aplicación ticket.

Para una mejor descripción del funcionamiento de la interfaz de usuario (UI) se hace necesario diseñar y conformar las principales pantallas de la aplicación mediante prototipos de interfaz y *storycards*, ambas representan herramientas muy potentes en las que se basa la metodología de desarrollo Mobile-D. Tanto los prototipos como los *storycards* son derivados de los propios requisitos funcionales o posibilitan el cumplimiento de los mismo de alguna manera.

Pantalla de Inicio de Sesión

Se pretende realizar una interfaz visual sencilla como muestra en el prototipo de interfaz de usuario de la **Figura 9** que posibilite el inicio de sesión en el sistema.

Figura 9. Prototipo de UI: Iniciar Sesión.

Para dar cumplimiento al requisito Iniciar de Sesión se planifica el *storycard* correspondiente en la **Tabla 3**.

Tabla 3. Storycard: Inicio de Sesión.

Numero/Id	**Tipo**	**Dificultad**		**Prioridad**	**Notas**
		Antes	Después		
01	Nuevo Fijo Mejora	Baja Media Alta	Baja Media Alta	Baja Media Alta	
Descripción					
Se requieren los campos de texto para que el usuario introduzca sus credenciales (usuario y contraseña) y un botón continuar para realizar la acción de autenticación con el sistema. Se debe mostrar el botón de *crear cuenta* para los nuevos usuarios y de esta forma posibilitar la navegación a la pantalla de registro.					
Fecha	**Estado**	**Comentario**			
2021/08/14	Definición	Sin comentario.			

Pantalla de Inicio

Para posibilitar la navegación dentro de la aplicación se requiere una pantalla principal donde se muestre el contenido relevante, así como un menú para visitar las distintas funcionalidades que brinda el sistema (Ver **Figura 10**).

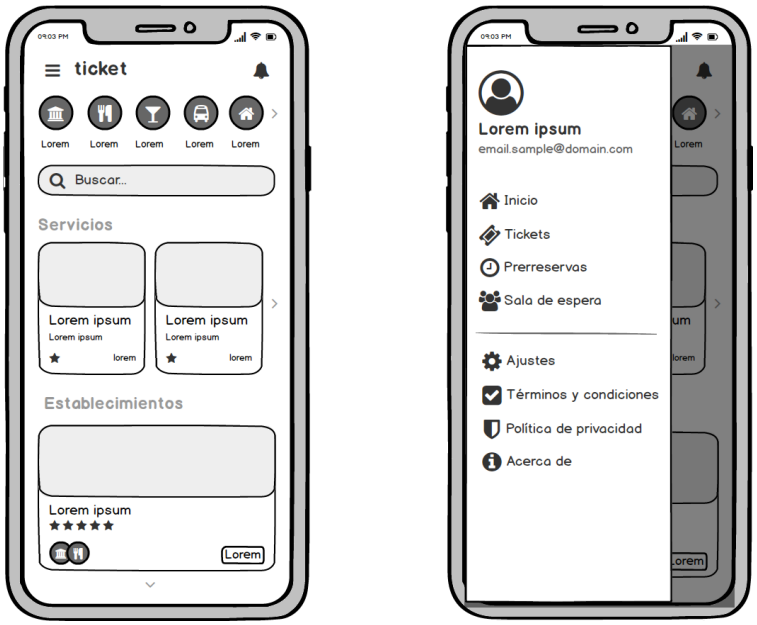

Figura 10. Prototipo de UI: Inicio

Para el desarrollo de esta pantalla, así como su menú lateral para la navegación, se dispone del siguiente *storycard* como se muestra en la **Tabla 4**.

Tabla 4. Storycard: Inicio.

Numero/Id	**Tipo**	**Dificultad**		**Prioridad**	**Notas**
		Antes	Después		
02	Nuevo Fijo Mejora	Baja Media Alta	Baja Media Alta	Baja Media Alta	
Descripción					
Mostrar servicios y establecimientos recomendados por el sistema. Facilitar la búsqueda de servicio y establecimientos a través de un cuadro de búsqueda, así como aplicar filtros rápidos desde una sección de categorías. Permitir una navegación hacia las pantallas de Tickets (reservaciones realizadas), Prerreservas (reservaciones pendientes a confirmar), Sala de espera, Historial de reservaciones, Ajustes, Acerca					

de y Notificaciones sin afectar la experiencia de usuario (UX). También se requiere acceder a la pantalla del perfil desde la misma pantalla principal de la aplicación.		
Fecha	**Estado**	**Comentario**
2021/08/15	Definición	Sin comentario.

Pantalla de Búsqueda

Encontrar fácilmente el servicio o establecimiento que solicita el usuario es de extrema importancia en la aplicación. Para esto se ha formulado el siguiente prototipo de interfaz de usuario (Ver **Figura 11**).

Figura 11. Prototipo de UI: Búsqueda.

En el *storycard* de la **Tabla 5** se especifica la funcionalidad que dará cumplimiento al prototipo de interfaz para la búsqueda.

Tabla 5. Storycard: Búsqueda.

Numero/Id	**Tipo**	**Dificultad**		**Prioridad**	**Notas**
		Antes	Después		
03	Nuevo Fijo	Baja Media	Baja Media	Baja Media	

	Mejora	Alta	Alta	Alta	
Descripción					
Para la búsqueda de servicios y establecimientos se requiere un campo de texto donde introducir sentencias o palabras que posibiliten depurar la búsqueda. Además, se necesita un filtro con los distintos criterios, categorías, valoraciones, precio, moneda, provincia o municipio. Luego se debe permitir seleccionar y ver el detalle del servicio o establecimiento encontrado.					
Fecha	**Estado**	**Comentario**			
2021/08/16	Definición	Sin comentario.			

Pantalla de Detalle de Servicio

En el detalle del servicio se deben encontrar todos los datos relevantes para la reservación, así como la funcionalidad de obtener el ticket para dicho servicio. Se diseña el prototipo de interfaz visual de usuario que se muestra a continuación en la **Figura 12**.

Figura 12. Prototipo de UI: Detalle de Servicio.

Para desarrollar las funcionalidades de la pantalla de Detalle de Servicio se diseña el *storycard* que se muestra en la **Tabla 6**.

Tabla 6. Storycard: Detalle de Servicio.

Numero/Id	Tipo	Dificultad		Prioridad	Notas
		Antes	Después		
04	Nuevo Fijo Mejora	Baja Media Alta	Baja Media Alta	Baja Media Alta	
Descripción					
Se hace necesario mostrar toda la información relevante al servicio como pueden ser; nombre, tipo, valoración, disponibilidad, precio, establecimiento, imágenes de publicidad y su respectiva descripción. Se debe poder acceder al establecimiento si pincha encima del nombre de este. Opción de observar las imágenes a pantalla completa si seleccionan. La acción de valorar el servicio o añadir a favoritos debe estar visible en el menú superior a la derecha. El botón de obtener ticket conviene posicionarlo bien visible al usuario ya que es la acción principal de la pantalla y se debe navegar a la creación de reservación una vez presionado. Si la configuración del servicio lo permite, se debe mostrar el botón de sala de espera para poder ingresar a ella luego de haber completado los datos requeridos.					
Fecha	**Estado**	**Comentario**			
2021/08/17	Definición	Sin comentario.			

Pantallas Obtener Reservación

La reservación es un proceso que se divide en varias pequeñas pantallas, algunas son opcionales como *Proveedores* y *Pago*, en dependencia de la configuración o estado del servicio a reservar. (Ver **Figura 13**)

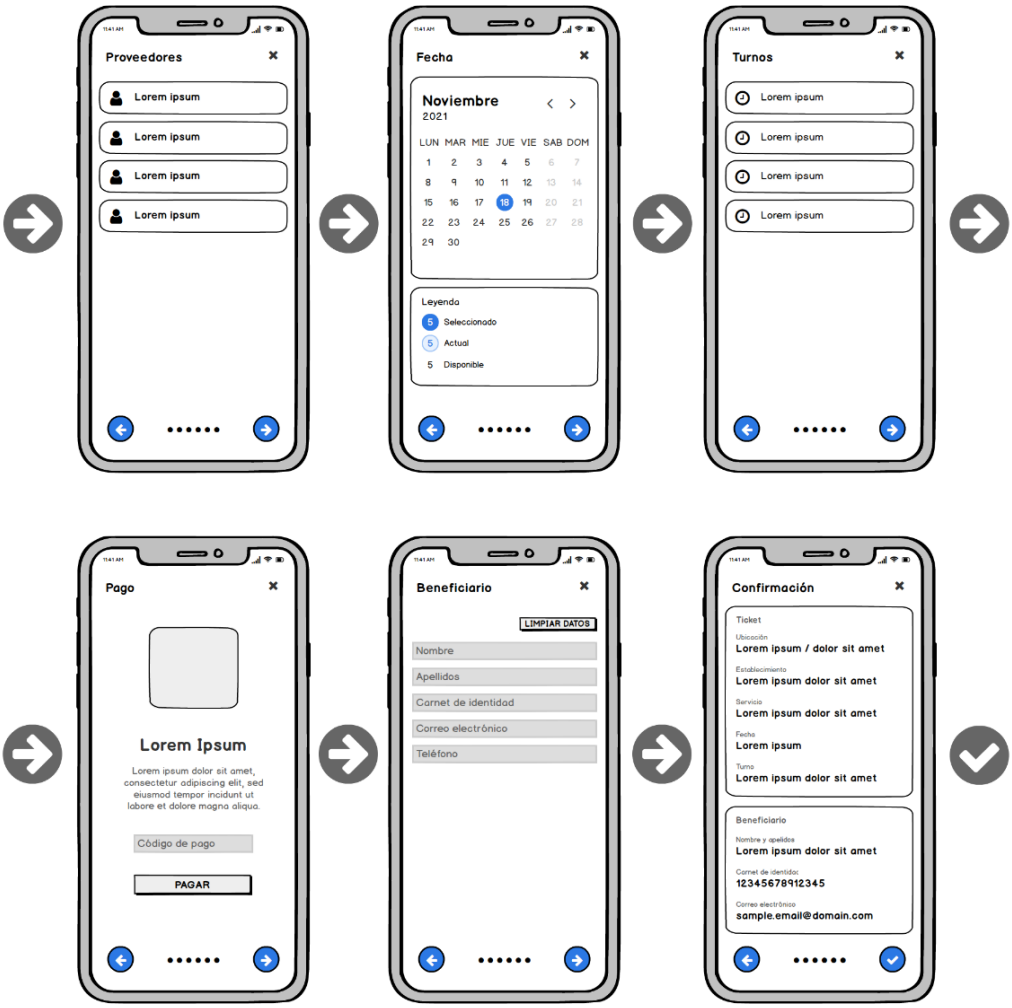

Figura 13. Prototipos de UI: Reservación.

La reservación (obtención del ticket) comprende varios pasos que se abarcan en el *storycard* de la **Tabla 7** a continuación.

Tabla 7. Storycard: Reservación.

Numero/Id	**Tipo**	**Dificultad**		**Prioridad**	**Notas**
		Antes	Después		
05	Nuevo Fijo Mejora	Baja Media Alta	Baja Media Alta	Baja Media Alta	
Descripción					

Para la navegación entre pantallas se utilizará una barra inferior con los botones de *Regresar* y *Continuar*. Se debe mostrar la opción (botón) de cancelar el proceso en todo momento. Se comienza el proceso en la pantalla opcional *Proveedores* (si no es necesario se inicia en selección de *Fecha*) donde el usuario elige que proveedor le facilitará el servicio, una vez seleccionado se debe navegar a la pantalla de *Fecha* donde se eligen el día a reservar. Para la selección del *Turno* en la pantalla siguiente, el usuario debe escoger el horario para su ticket. En caso de que el servicio requiera pago (si no es requerido, pasar a la pantalla *Beneficiario*) se hace necesario escoger un método de los disponibles y luego navegar a la pantalla de *Pago*. Se debe poder crear y confirmar el pago antes de navegar a la siguiente pantalla. Una vez concluido, se navega a la pantalla *Beneficiario* donde se insertarán los datos en sus respectivos capos de texto como son; nombre, apellidos, carnet de identidad, correo electrónico y teléfono. Para la obtención final del ticket se debe navegar a la pantalla de *Confirmación* y verificar por el usuario que todos los datos seleccionados e insertados son correctos. Una vez concluido el proceso de reservación se debe poder navegar hacia la pantalla anterior.		
Fecha	**Estado**	**Comentario**
2021/08/18	Definición	Sin comentario.

Pantalla de Listar Reservaciones

Los tickets (reservaciones) son el resultado del proceso de reservación. Se pueden visualizar accediendo desde el menú lateral de la pantalla principal. (Ver **Figura 14**)

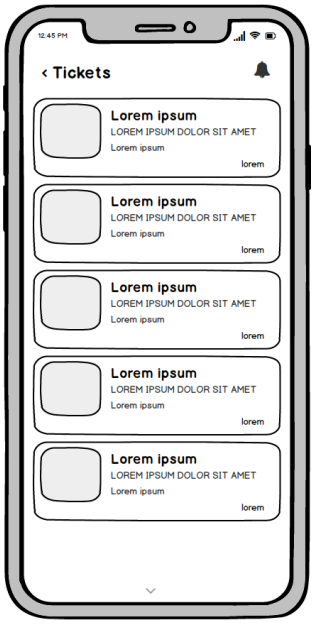

Figura 14. Prototipo de UI: Listar Reservaciones.

En la **Tabla 8** se puede definen el *storycard* que comprende la funcionalidad de listar las reservaciones realizadas.

Tabla 8. Storycard: Listar Reservaciones.

Numero/Id	**Tipo**	**Dificultad**		**Prioridad**	**Notas**
		Antes	Después		
06	Nuevo Fijo Mejora	Baja Media Alta	Baja Media Alta	Baja Media Alta	
Descripción					
Mantener accesible la pantalla de Notificaciones mediante un elemento en el menú superior de la pantalla de reservaciones. Se deben mostrar las reservaciones realizadas en forma de lista vertical, con los datos más relevantes de esta como pueden ser; imagen, nombre de servicio, establecimiento, fecha y turno. En caso de presionar un elemento de la lista se permitirá navegar a la apantalla *Detalle de Ticket*.					
Fecha	**Estado**	**Comentario**			
2021/08/19	Definición	Sin comentario.			

2.3 CONCLUSIONES PARCIALES

En el capítulo, se identificaron los grupos de interés que intervienen de alguna forma en el desarrollo de la aplicación. Se realizó la captura de requisitos funcionales como no funcionales que marcan los objetivos y restricciones para tener en cuenta durante todo el proceso de desarrollo.

Se diseñaron los principales artefactos de la metodología Mobile-D que posibilitaron; conocer la arquitectura del sistema de reservaciones, confeccionar el diseño de base de datos para la aplicación, así como identificar las bibliotecas o dependencias necesarias mediante el diagrama de componentes.

El capítulo viabilizó la realización de prototipados para las principales interfaces de usuario, así como sus respectivos *storycards* para su futura implementación.

3 CAPÍTULO III. IMPLEMENTACIÓN DE LA PROPUESTA DE SOLUCIÓN

En este capítulo, se configura el ambiente de desarrollo elegido y se codifican las funcionalidades tratadas en la etapa de diseño, la cual se distribuye en las fases de Exploración e Inicialización planteadas en la metodología de desarrollo Mobile-D. Se ejemplifican algunos de los patrones de diseño utilizados y la descripción de los componentes más notables.

3.1 PRODUCCIÓN Y ESTABILIZACIÓN

Las fases de Producción y Estabilización del producto están comprendidas dentro de la etapa de Implementación. En esta se repiten iterativamente las subfases, con un día de planificación, uno de trabajo y uno de entrega. (Vique, 2011)

3.1.1 Configuración del Ambiente de Desarrollo

En esta actividad se debe realizar la configuración de los elementos técnicos, en este caso se configura el entorno de desarrollo para aplicaciones de Android Studio.

- Tipo de Proyecto: Android Application Project (con Kotlin).
- Configuraciones: Inclusión de las dependencias (bibliotecas) necesarias donde destacan *Retrofit* para el consumo de la API REST, *Room* para el trabajo con la base de datos, *Picasso* para el manejo de imágenes, *Work Manager* para la realización de tareas en segundo plano y *Material* con su paquete de componentes visuales respetando las líneas de diseño de Material Design.

3.1.2 Estructura de directorios

En la **Figura 15** se puede observar la estructura de los directorios más importantes dentro del proyecto que se desarrolla.

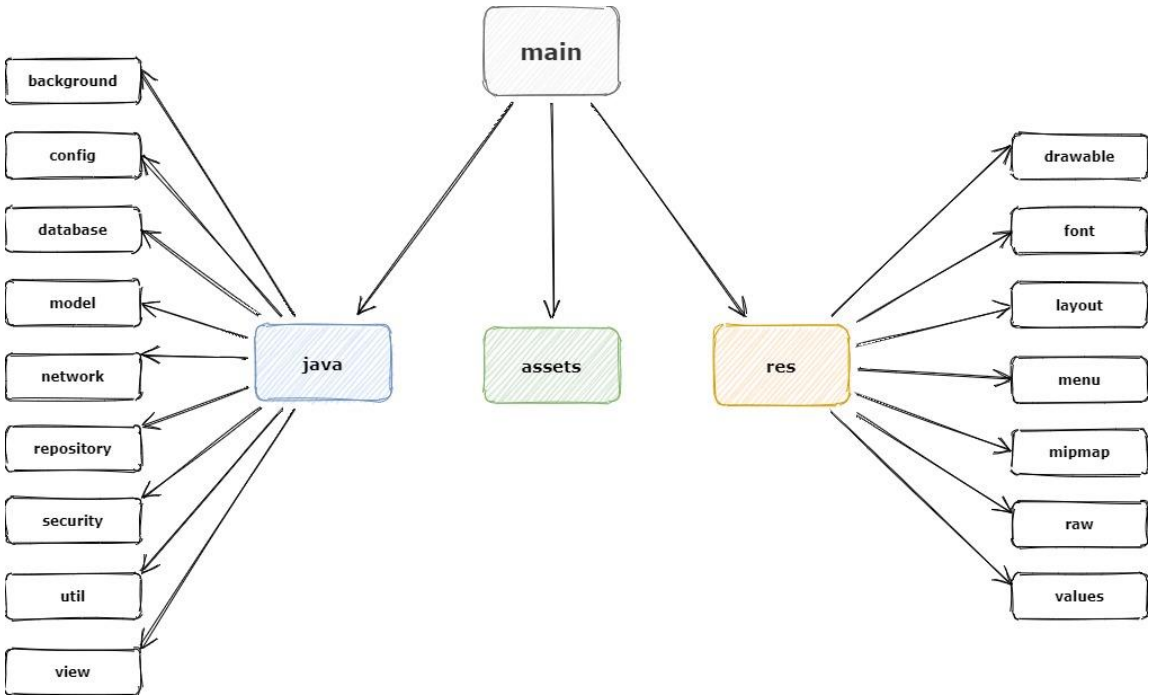

Figura 15. Estructura de directorios del proyecto.

El paquete *main* constituye el directorio principal el cual abarca a los directorios *java*, *assets* y *res*.

El directorio *java* se crea de manera automática al comenzar un proyecto en Android, y aunque *ticket* se desarrolla utilizando el lenguaje de programación Kotlin, esto no interfiere con el normal funcionamiento de la aplicación. El paquete *java* contiene las clases más importantes, desde la capa de datos, controladores de vistas, clases del modelo e incluso de utilidades.

Los *assets* comprenden todos aquellos archivos externos de solo lectura que pueden ser cargados dentro de la aplicación como pueden ser archivos de texto, base de datos, imágenes. En *ticket* este directorio se utiliza para almacenar las provincias y municipios que luego serán cargadas por la base de datos para su futura utilización.

El directorio *res* representa la raíz de todos los recursos que Android provee a la aplicación, como pueden ser; vectores, imágenes, íconos, fuentes y otros archivos XML como pueden las vistas de la interfaz de usuario de la aplicación, así como sus propios menús.

3.1.3 Codificación

La codificación se realizó basándose en cada uno de los *storycards* realizados en la fase anterior de Inicialización.

Inicio de Sesión

Para el inicio de sesión se crea un archivo XML en el directorio main/res/layout/ llamado *activity_login.xml* en el cual se diseñan los componentes visuales de la pantalla. Se hace necesario un campo de texto para el correo de usuario, un campo de texto para la contraseña, un botón para enviar las credenciales hacia el servicio web (API REST), un botón para navegar a la *Pantalla de Registro* y otro para realizar el proceso de recuperación de contraseña. (Ver **Figura 16**)

Figura 16. UI: Inicio de Sesión.

Para hacer funcional la pantalla se dispone fundamentalmente de las siguientes clases:

- AuthRepo: Maneja los datos y peticiones entre el servicio web y el ViewModel.
- LoginViewModel: Contiene los datos relacionados a la vista del inicio de sesión, además de las funcionalidades de autenticación y recuperación de contraseña.

- LoginActivity: Clase que hace uso de LoginViewModel, valida los datos introducidos por el usuario y maneja los componentes del archivo activity_login.xml.

Luego de la autenticación satisfactoria en el sistema se navega hacia la pantalla de *Inicio*.

Inicio

Luego del inicio de sesión se muestra la pantalla de Inicio (pantalla principal). Se hace necesario la creación del archivo *main/res/layout/base_activity_main.xml* que maneja la clase MainActivity. Esta clase controla la navegación hacia varios destinos en forma de fragmentos (Fragments) como son *Inicio*, *Tickets*, *Prerreservas*, *Sala de espera* e *Historial*. En la pantalla de *Inicio* se utilizan varios componentes RecyclerView para mostrar listas horizontales y verticales como son; la vista de categorías en la parte superior, la vista de servicios recomendados y la de establecimientos populares. Se inserta un cuadro de búsqueda que posibilita la navegación hacia la pantalla de *Búsqueda*.

Se construye el menú que contiene la navegación hacia la pantalla de *Notificaciones*, así como el menú lateral en forma de NavigationDrawer. (Ver **Figura 17**)

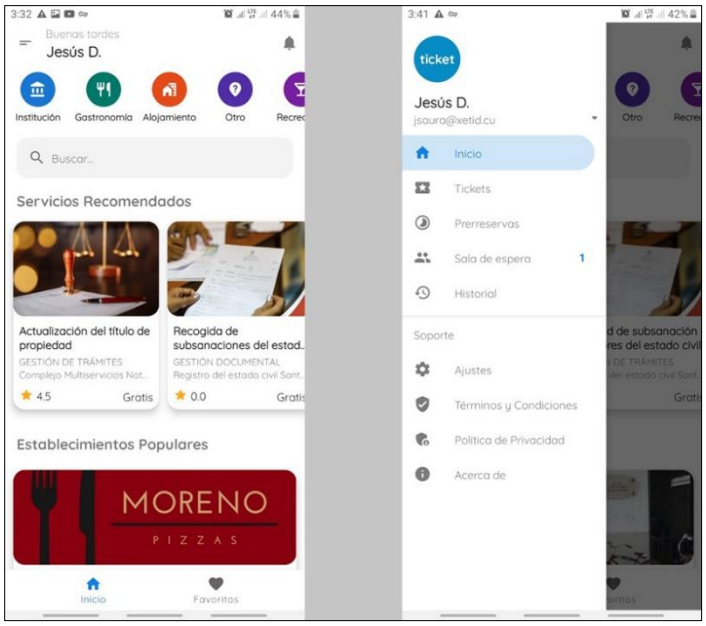

Figura 17. UI: Inicio.

Para hacer funcional la pantalla se dispone fundamentalmente de las siguientes clases:

- MainRepo: Provee los datos desde la API Rest de los servicios y establecimientos que requiere la pantalla principal.
- HomeViewModel: Depende de la clase MainRepo para brindar la información al MainFragment. Contiene los métodos *fetchMainData* y *fetchInitialData*.
- MainFragment: Maneja los listados de categorías, servicios y establecimientos en forma de RecyclerViews y contiene la acción de búsqueda en forma de campo de texto.
- MainActivity: Es el contenedor de los fragments de la navegación principal mediante un menú de tipo NavigationDrawer. También maneja las notificaciones en sus Toolbar en forma de menú.

Búsqueda

Se crea el fichero *main/res/layout/ activity_main_search.xml* para ser utilizado luego por el controlador de la actividad.

La búsqueda de servicios y establecimientos implica dividir los resultados en dos Tabs diferentes. Se utiliza un campo de texto para realizar las búsquedas requeridas por el usuario donde es posible apoyarse del filtro de búsqueda avanzado que despliega el botón flotante (FloatingActionButton) de la parte inferior derecha de la pantalla. Esta búsqueda avanzada comprende los distintos criterios como pueden ser las categorías, valoraciones, precio, moneda, provincia o municipio.

Los resultados se visualizan en forma de listado compuesto por elementos de tipo CardView, un componente muy utilizado en Material Design para casos de uso similares. (Ver **Figura 18**)

Una vez se selecciona el servicio o establecimiento buscado se navega a la pantalla de detalle correspondiente.

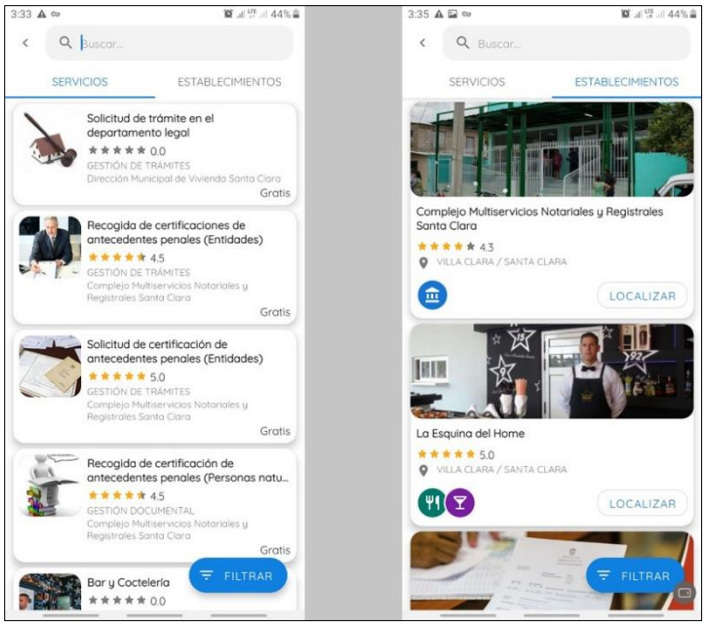

Figura 18. UI: Búsqueda.

Para hacer funcional la pantalla se dispone fundamentalmente de las siguientes clases:

- MainSearchActivity: Actividad que contiene un TabLayout para mostrar los fragments de servicio y establecimientos en cada página. También posee el campo de texto para realizar búsquedas.
- ServicesFragment: Fragmento donde se listan los servicios que coincidan con los criterios de búsquedas.
- EstablishmentsFragment: Fragmento donde se listan los establecimientos que coincidan con los criterios de búsquedas.
- ServiceRepo: Se encarga de realizar las peticiones a la API REST con los filtros de búsqueda confeccionados para la obtención de los servicios en forma paginada.
- EstablishmentRepo: Realiza la misma función que ServiceRepo pero utilizando los establecimientos.
- ServicesViewModel: Envía el filtro confeccionado a ServiceRepo y contiene los datos de respuesta para que luego sean mostrados en ServicesFragment.

- EstablishmentsViewModel: Envía el filtro confeccionado a EstablishmentRepo y contiene los datos de respuesta para que luego sean mostrados en EstablishmentsFragment.

Detalle de Servicio

Cada servicio posee una pantalla de detalle en la que se muestra toda la información relacionada con el mismo. Se crea una su respectivo recurso de tipo layout nombrado *activity_service_detail.xml.*

Para las imágenes se utiliza el componente ImageView de Android con la complementación de las bibliotecas *Pictish* para mostrar el proceso de carga y *GalleryDroid* para manejar la vista a pantalla completa y navegación entre una imagen y otra. Se dispone del botón de obtención del ticket para dar paso al proceso de reservación y de otro botón para los servicios que tengan una sala de espera. Para la valoración y el manejo de favoritos se crea el menú menu_service_detail.xml. (Ver **Figura 19**)

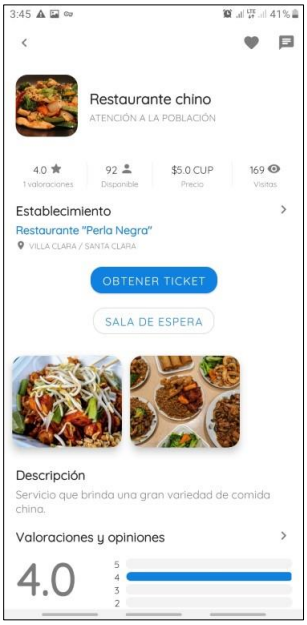

Figura 19. UI: Detalle de Servicio.

Para hacer funcional la pantalla se dispone fundamentalmente de las siguientes clases:

- ServiceRepo: Realiza las peticiones al servidor de los datos del servicio. Gestiona el añadir o remover de los favoritos al servicio.
- ServiceDetailViewModel: Depende de ServiceRepo, contiene y maneja la información que se muestra en pantalla y se encarga de llamar a los métodos que se encuentran en ServiceRepo.
- ServiceDetailFragment: Utiliza a ServiceDetailViewModel para la gestión de acciones y fuente de datos a mostrar. Contiene los componentes visuales que interactúan con el usuario.
- ServiceDetailActivity: Es el contenedor del fragmento ServiceDetailFragment. Dispone del menú en la Toolbar para la gestión de favoritos y la acción de valoración del servicio.

Reservación

El proceso de reservación (obtención del ticket) se compone de varias pantallas de tipo Fragment manejadas desde una actividad CreateBookingActivity. Para la funcionalidad se crea el archivo activity_booking_create.xml en el directorio correspondiente a layout. A su vez se crean también los archivos XML de cada fragmento. Durante todo el proceso se muestra un indicador en la parte inferior de la pantalla para notificar el avance entre las distintas pantallas.

En caso de que el servicio contenga la necesidad de seleccionar proveedores se navega hacia ProvidersFragment compuesto por un RecyclerView listando todos los proveedores disponibles de ese servicio a reservar.

El siguiente paso consiste en elegir la fecha de reservación donde se muestra un calendario haciendo uso de la biblioteca *CalendarView* para facilitar el manejo de los días y disponibilidades.

Luego se requiere la elección los turnos a partir de las disponibilidades del día seleccionado en el paso anterior. Para ello se utiliza un RecyclerView con la denominación del turno en cada elemento.

El pago es una pantalla que depende de la configuración previa del servicio y que luego de la prerreservación del mismo se dispone de un tiempo predeterminado para completar

los datos requeridos, la elección de la plataforma de pago y su confirmación. En caso de la plataforma Enzona se utiliza una WebView para la confirmación del pago ya que es la única vía disponible que se brinda. Se hace uso de la biblioteca *EnzonaPayLibrary* para facilitar el proceso con el servicio web. Una vez concluido se navega a la pantalla de *Beneficiario*.

El fragmento del beneficiario requiere la inclusión de los campos de texto para insertar los datos de la persona a la cual va dirigida la reservación. Cada campo es validado por el controlador de la vista BeneficiaryFragment y se da paso a la verificación de los datos en la pantalla de *Comprobación*.

En la *Comprobación* se hace un resumen de todos los datos seleccionados y recogidos durante el proceso y se espera la ratificación del usuario para realizar finalmente la reservación mediante el FloatingActionButton en la parte inferior derecha de la pantalla.

El resultado de la implementación del proceso de reservación se muestra en la **Figura 20** a continuación.

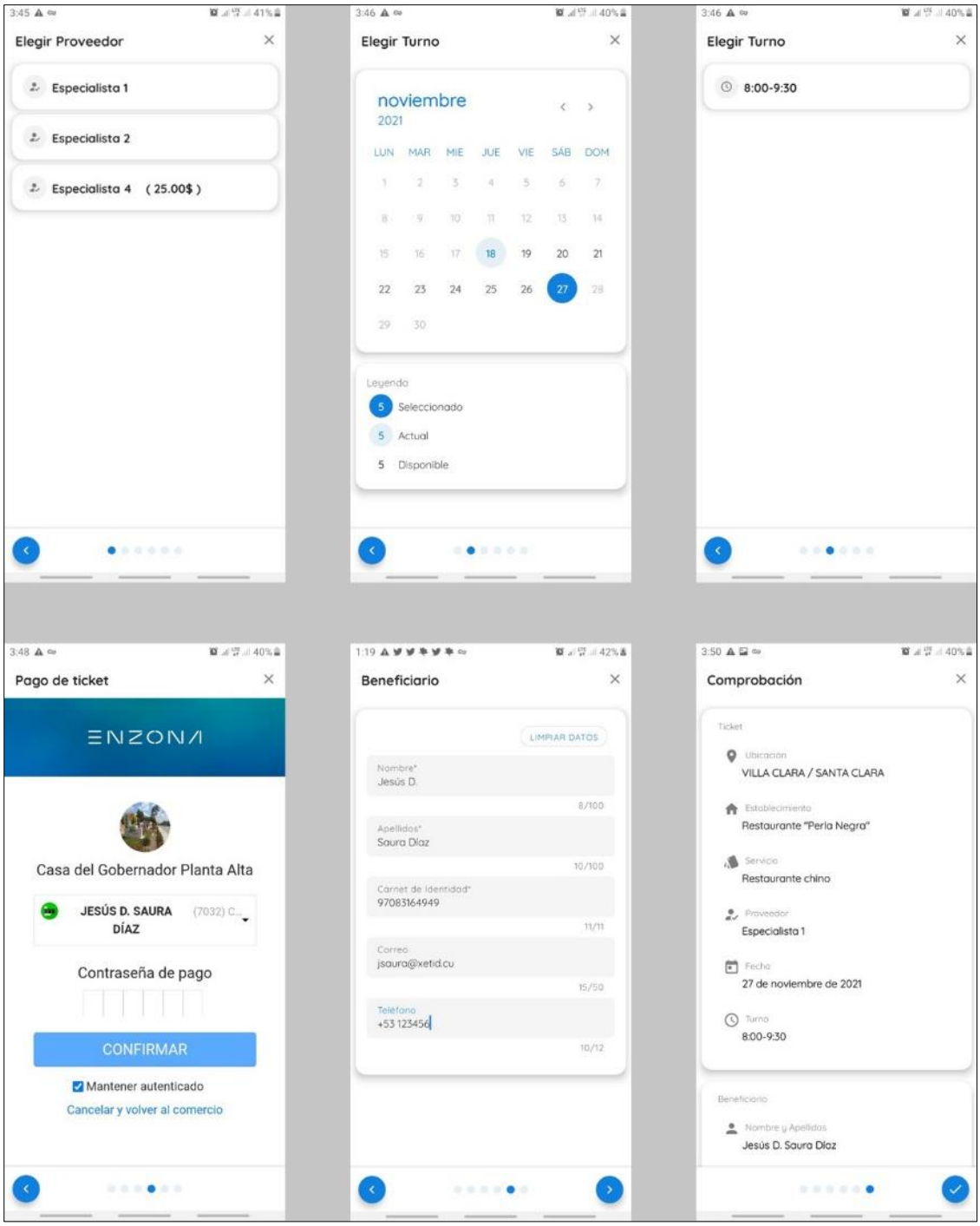

Figura 20. UI: Reservación.

Listar Reservaciones

El usuario comprueba sus tickets al listarlos en la pantalla Tickets accesible desde el menú lateral de la pantalla principal. Se utiliza un fragment con un RecyclerView en forma

vertical para mostrar las distintas reservaciones realizadas y compuestas cada una por un CardView.

Se muestra la información más relevante de la reserva con posibilidad de pasar al detalle del ticket al presionar sobre un elemento. Como acción secundaria se dispone de un FloatingActionButton con las acciones de navegar a la pantalla de *Búsqueda* y de *Historial*. (Ver **Figura 21**)

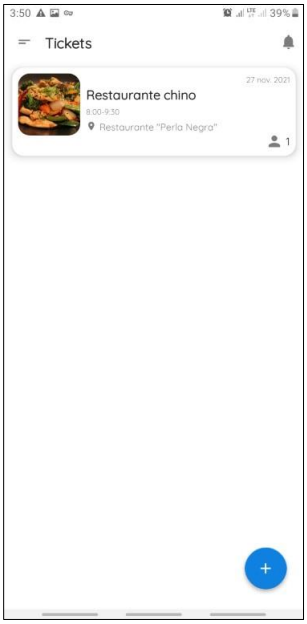

Figura 21. UI: Listar Reservaciones.

Para hacer funcional la pantalla se dispone fundamentalmente de las siguientes clases:

- BookingRepo: Contiene el método *fetchBookings* el cual devuelve de forma paginada todas las reservaciones activas realizadas por usuario. Para cancelar la reservación se dispone del método *deleteBooking*.
- BookingsViewModel: Depende de BookingRepo para la obtención de los datos requeridos que son mostrados en la pantalla BookingsFragment.
- BookingsFragment: Fragmento que contiene el RecyclerView con los tickets obtenidos. Forma parte de la actividad principal de la aplicación MainActivity.

3.1.4 Código de interés

A continuación, se muestran las clases y métodos más importantes que constituyen parte fundamental en el funcionamiento de la aplicación.

Patrón de diseño Singleton de la Base de Datos

Cuando existe un objeto del cual debe existir únicamente una instancia y es deseable un acceso global a esta instancia se utiliza el patrón Singleton. Comúnmente un Singleton se implementa definiendo un tipo de objeto que posee como uno de sus miembros una única instancia de sí mismo y un método estático llamado el cual retorna una referencia a esta única instancia del objeto. (Bermúdez et al., 2012)

Para la Base de Datos se utiliza el patrón Singleton de manera que no se pueda crear otra referencia del mismo objeto de Room. (Ver **Figura 22**)

```
@Database(
    entities = [Province::class, Municipality::class, Notification::class],
    version = DB_VERSION, exportSchema = false
)
@TypeConverters(RoomConverters::class)
abstract class KRoomDatabase : RoomDatabase() {

    abstract fun provinceDao(): ProvinceDao
    abstract fun municipalityDao(): MunicipalityDao
    abstract fun notificationDao(): NotificationDao

    companion object {

        @Volatile
        private var instance: KRoomDatabase? = null

        fun getInstance(context: Context): KRoomDatabase =
            instance ?: synchronized(this) {
                instance ?: buildDatabase(context).also { instance = it }
            }

        private fun buildDatabase(context: Context): KRoomDatabase = Room
            .databaseBuilder(context.applicationContext, KRoomDatabase::class.java, DB_NAME)
            .fallbackToDestructiveMigration()
            .allowMainThreadQueries()
            .build()

    }

}
```

Figura 22. Patrón de diseño Sigleton utilizado para Room.

Parte del Servicio Web implementado en Retrofit

Retrofit es un cliente REST de tipo seguro para Android, Java y Kotlin desarrollado por Square. La biblioteca proporciona un marco poderoso para autenticar e interactuar con API y enviar solicitudes de red con OkHttp. Esta biblioteca hace que la descarga de datos JSON o XML desde una API web sea bastante sencilla. (Codepath, 2015)

A continuación, en la **Figura 23**, se muestra la implementación de los métodos de iniciar sesión (postLogin) y registrar usuario (postRegister) sobre la interfaz que requiere

Retrofit para funcionar. Nótese el modificador suspend que define a ambos métodos para ser ejecutados por las *coroutinas*[3] de Kotlin.

```
interface WSO2Service {

    @FormUrlEncoded
    @POST(URL_POST_LOGIN)
    suspend fun postLogin(
        @Field("scope") clientId: String = OPEN_ID,
        @Field("username") userName: String,
        @Field("password") password: String,
        @Field("grant_type") grantType: String = GRANT_TYPE,
    ): Response<AuthBearerToken?>

    @FormUrlEncoded
    @POST(URL_POST_REGISTER)
    suspend fun postRegister(
        @Field("application") app: String = APP_NAME,
        @Field("uid") uid: String,
        @Field("username") userName: String,
        @Field("mail") mail: String,
        @Field("name") name: String,
        @Field("lastname") lastName: String,
        @Field("password") password: String,
        @Field("dni") dni: String,
        @Field("province") province: String? = null,
        @Field("municipality") municipality: String? = null,
    ): Response<PostResponse?>

}
```

Figura 23. Parte del Servicio Web implementado en Retrofit.

ViewModel para la pantalla Listar Servicios

El ViewModel responsable de poblar de datos a la pantalla *Listar Servicios* contiene el método *fetchServices* (Ver **Figura 24**), el cual hace peticiones de forma paginada a la API mediante la biblioteca Paging integrada a la aplicación.

[3] Una coroutine es un patrón de diseño de simultaneidad que puede usar en Android para simplificar el código que se ejecuta de forma asincrónica. GOOGLE. 2021b. *Kotlin coroutines on Android* [Online]. Google Developers: Google Developers. Available: https://developer.android.com/kotlin/coroutines [Accessed November 15 2021].

```
class ReviewsViewModel(private val app: Application) : LiveCoroutinesViewModel(app) {

    val itemsCount by lazy { MutableLiveData<Int>() }

    fun fetchReviews(reviewType: ReviewType, resourceId: Int): LiveData<PagedList<Review>> {
        statusResource.postValue(StatusResource.loading())
        val config = PagedList.Config.Builder()
            .setPageSize(20)
            .setEnablePlaceholders(false)
            .build()

        val dataSourceFactory = object : DataSource.Factory<Int, Review>() {
            override fun create(): DataSource<Int, Review> =
                ReviewDataSource(
                    app = app,
                    scope = viewModelScope,
                    statusResource = statusResource,
                    reviewType = reviewType,
                    resourceId = resourceId,
                ).apply {
                    countListener = object : CountListener {
                        override fun onCount(count: Int) = itemsCount.postValue(count)
                    }
                }
        }
        return LivePagedListBuilder(dataSourceFactory, config).build()
    }

}
```

Figura 24. ViewModel para la pantalla Listar Servicios.

3.2 CONCLUSIONES PARCIALES

Como parte de las fases de Producción y Estabilización que presenta la metodología Mobile-D se logró implementar cada uno de los *storycards* planificados en la fase de Inicialización, partiendo de la configuración inicial del proyecto y la identificación de los directorios principales de la aplicación.

Se justificó el uso de algunas bibliotecas y patrones de diseños orientadas a los requisitos de cada capa arquitectónica de la aplicación.

Se realizaron las iteraciones requeridas por cada funcionalidad lo que posibilitó mantener un sistema estable y funcional en la mayoría del tiempo.

CONCLUSIONES

Con el presente trabajo se analizó la situación e impacto de las aplicaciones móviles en la actualidad y su utilización en el sector de las reservaciones en línea. Esto permitió la realización de comparativas entre los sistemas ya existentes, teniendo como resultado el perfeccionamiento del software que se pretendía construir.

Se describió los procesos de análisis, diseño e implementación de la aplicación móvil para dispositivos con sistema operativo Android, que posibilita la realización de reservaciones de forma remota para negocios y servicios.

El empleo de la metodología Mobile-D, para el desarrollo del trabajo, implica un tiempo relativamente breve para la entrega del producto final, obteniendo productos funcionales en cada iteración de las fases de producción y estabilización. Es esta una de las metodologías más utilizadas en el desarrollo de aplicaciones para dispositivos móviles en la actualidad.

Resultó ser un producto innovador dentro del país ya que la aplicación no se enfoca en un solo tipo de servicio y permite la inclusión de varios sectores de la sociedad en un mismo sistema al alcance de la población desde su dispositivo móvil.

REFERENCIAS

BERMÚDEZ, G. S., JIMÉNEZ, I. M. & RODRÍGUEZ, L. R. J. I. R. D. L. U. D. C. R. 2012. USO DE PATRONES DE DISEÑO: UN CASO PRÁCTICO. 22**,** 45-59.

CODEPATH. 2015. Consuming APIs with Retrofit. Available: https://guides.codepath.com/android/consuming-apis-with-retrofit [Accessed November 12].

DAVID, M. 2021. Desarrollo de aplicaciones móviles. Available: https://www.computerweekly.com/es/definicion/Desarrollo-de-aplicaciones-moviles [Accessed 2021/11/09].

DIGITAL, M. 2021. *MKT Marketing Digital* [Online]. Marketing Digital. Available: https://mktmarketingdigital.com/atencion-al-cliente-en-tiempos-de-covid/ [Accessed September 08 2021].

ECURED 2018. Empresa de Tecnologías de la Información para la Defensa. *EcuRed.* EcuRed.

EDUCATION, I. C. 2021. API REST. Available: https://www.ibm.com/pe-es/cloud/learn/rest-apis [Accessed November 15].

GONZÁLEZ, A. 2020. *Sistemas operativos móviles y sus diferentes tipos* [Online]. Tecnoinformatic: Tecnoinformatic. Available: https://tecnoinformatic.com/c-sistemas-operativos/sistemas-operativos-moviles/ [Accessed September 11 2021].

GOOGLE. 2021a. *Guide to app architecture* [Online]. Google Developers: Google Developers. Available: https://developer.android.com/jetpack/guide [Accessed November 11 2021].

GOOGLE. 2021b. *Kotlin coroutines on Android* [Online]. Google Developers: Google Developers. Available: https://developer.android.com/kotlin/coroutines [Accessed November 15 2021].

GOOGLE. 2021c. *Meet Android Studio* [Online]. Google Developers: Google Developers. Available: https://developer.android.com/studio/intro/ [Accessed Sepetember 12 2021].

GOOGLE. 2021d. *Save data in a local database using Room* [Online]. Google Developers: Google Developers. Available: https://developer.android.com/training/data-storage/room/ [Accessed September 13 2021].

GOOGLE. 2021e. *Save data using SQLite* [Online]. Google Developers: Google Developers. Available: https://developer.android.com/training/data-storage/sqlite/ [Accessed September 13 2021].

HAT, R. 2020. ¿Qué es una API de REST? Available: https://www.redhat.com/es/topics/api/what-is-a-rest-api [Accessed November 4].

HAT, R. 2021. El concepto de desarrollo de aplicaciones móviles para empresas. Available: https://www.redhat.com/es/topics/mobile.

HERAZO, L. 2019. ¿Qué es una aplicación móvil? Anincubator: Anincubator.

ISLAZUL. 2021. *Hoteles Islazul* [Online]. Islazul. Available: https://www.islazulhotels.com/ [Accessed September 14 2021].

LAB, B. 2019. *Ventajas de tener un sistema de reservas online* [Online]. Bartalent Lab: Bartalent Lab. Available: https://www.bartalentlab.com/degustanews/hosteleria-digital/hosteleria-digital/ventajas-sistema-reservas-online [Accessed September 21 2021].

LUETIĆ, M. 2021. Cross-Platform vs. Native Mobile Development. DECODE: DECODE.

MELNICHUK, A. 2020. Android IDEs for Developers. NCube: NCube.

MUÑOZ, C. A. M. 2020. *Aplicación de la metodología Mobile-D en el desarrollo de una app móvil para gestionar citas médicas del Centro JEL Riobamba.* Universidad Nacional de Chimborazo.

OBERLO. 2021. *What percentage of internet traffic is mobile?* [Online]. Oberlo: Oberlo. Available: https://www.oberlo.com/statistics/mobile-internet-traffic/ [Accessed September 10 2021].

OCR 2013. The OCR Guide to Storyboards. *In:* OCR (ed.). OCR.

RAFAEL, C. M. G. 2018. *Tablero de control de indicadores de desempeño para plataforma Android.* Universidad Central "Marta Abreu" de Las Vilas.

ROGER S. PRESSMAN, P. D. 2010. *Software Engineering. A Practitioner's Approach*, McGraw-Hill.

S.A, C. 2021a. *Hoteles Cubanacan* [Online]. Cubanacan S.A. Available: https://www.hotelescubanacan.com/ [Accessed September 14 2021].

S.A, H. 2021b. *Habanatur S.A - Agencia de turismo en Cuba* [Online]. Habanatur S.A. Available: https://www.havanatursa.com/ [Accessed September 14 2021].

SCHOOL, I. D. 2018. *Impacto de las aplicaciones móviles sobre el comportamiento del consumidor en la decisión de compra online* [Online]. ID Digital School. Available: https://iddigitalschool.com/impacto-de-las-aplicaciones-moviles-sobre-el-comportamiento-del-consumidor-en-la-decision-de-compra-online/ [Accessed September 10 2021].

SHEKHAR, A. 2020. MVVM Architecture - Android Tutorial for Beginners - Step by Step Guide. Available: https://blog.mindorks.com/mvvm-architecture-android-tutorial-for-beginners-step-by-step-guide [Accessed November 10].

SIMS, G. 2019. I want to develop Android apps — What languages should I learn? Android Authority: Android Authority.

SOMMERVILLE, I. 2001. *Software engineering,* Harlow, England, Pearson Education Limited.

STATS, S. G. 2021a. *Desktop vs Mobile Market Share Cuba: 2011 - 2021* [Online]. Statcounter Global Stats: Statcounter Global Stats. Available: https://gs.statcounter.com/platform-market-share/desktop-mobile/cuba#yearly-2011-2021/ [Accessed September 11 2021].

STATS, S. G. 2021b. *Mobile Operating System Market Share Cuba: Aug 2020 - Aug 2021* [Online]. Statcounter Global Stats: Statcounter Global Stats. Available: https://gs.statcounter.com/os-market-share/mobile/cuba/ [Accessed September 08 2021].

STATS, S. G. 2021c. *Mobile Operating System Market Share Worldwide: Aug 2020 - Aug 2021* [Online]. Statcounter Global Stats: Statcounter Global Stats. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide/ [Accessed September 08 2021].

STELLMAN, A. G., JENNIFER 2005. *Gestión de proyectos de software aplicado*, O'Reilly Media.

TALENT, I. D. 2017. Las ventajas de tener una aplicación móvil. Available: https://www.isdi.education/mx/blog/las-ventajas-de-tener-una-aplicacion-movil [Accessed 2021/11/09].

TECHNOLOGIES, C. 2018. How To Define Stakeholders For Your Software Development Project. Available from: https://www.conceptatech.com/blog/how-to-define-stakeholders-for-your-software-development-project [Accessed August 21 2021].

TEROL, C. G. 2015. 10 Sistemas de reservas online para tu negocio en Internet.

V.V. SUBRAHMANYAM, K. S. 2011. A Study on Mobile Operating Systems and their Recent Advances. Published in the Proceedings. Available: https://www.researchgate.net/publication/315108816_A_Study_on_Mobile_Operating_Systems_and_their_Recent_Advances.

VIQUE, R. R. 2011. Tecnología y desarrollo en dispositivos móviles. Available: https://desarrolloappandroid.files.wordpress.com/2013/06/tecnologia-desarrollo-dispositivos-moviles.pdf [Accessed September 12, 2021].