<CIS2030CP01>

FlashPark

SERGIO POSADA HENAO
DANIEL ESTEBAN VALERO GALEANO
VIVIANA ANDREA AMARIS BALDA

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA DE SISTEMAS
BOGOTÁ, D.C.
2021

<CIS2030CP01>

FlashPark

Autor(es):

Sergio Posada Henao

Daniel Esteban Valero Galeano

Viviana Andrea Amaris Balda

MEMORIA DEL TRABAJO DE GRADO REALIZADO PARA CUMPLIR UNO DE LOS REQUISITOS PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

Director

Ing. Ricardo Jorge Naranjo Faccini, MSc.

Jurados del trabajo de grado

Ing. Carlos Castañeda Marroquín, PhD.

Admr. Oscar Mauricio Amado Castaño

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

SYSTEMS ENGINEERING PROGRAM

BOGOTÁ, D.C.

Noviembre, 2021

PONTIFICIA UNIVERSIDAD JAVERIANA

FACULTAD DE INGENIERÍA

SYSTEMS ENGINEERING PROGRAM

Rector Pontificia Universidad Javeriana

Padre Jorge Humberto Peláez Piedrahita, S.J.

Decano de la Facultad de Ingeniería

Ing. Lope Hugo Barrero Solano Ph.D.

Directora del Programa de Ingeniería de Sistemas

Ing. Alexandra Pomares Quimbaya Ph.D.

Director del Departamento de Ingeniería de Sistemas

Ing. César Julio Bustacara Medina Ph.D.

Artículo 23 de la Resolución No. 1 de Junio de 1946

"La Universidad no se hace responsable de los conceptos emitidos por sus alumnos en sus proyectos de grado. Sólo velará porque no se publique nada contrario al dogma y la moral católica y porque no contengan ataques o polémicas puramente personales. Antes bien, que se vean en ellos el anhelo de buscar la verdad y la Justicia"

1. AGRADECIMIENTOS

Queremos agradecer a Dios por la fortaleza que nos ha dado para sacar este trabajo adelante a pesar de las adversidades. A nuestra familia, amigos y en especial a nuestro director, gracias, porque siempre estuvieron a nuestro lado acompañándonos con amor y guiándonos con paciencia.

Adicionalmente queremos agradecer grandemente a Carlos, sin ti, nada de esto hubiera sido posible.

2. CONTENIDO

3. ABSTRACT

Proposed solution to the inconvenience that can occur when a driver is looking for a place to park his vehicle in the city of Bogotá, the solution is composed of a group of applications. The purpose of the first application is to provide the user with information about the parking lots near their destination, so they can organize their way without setbacks, avoid bad times and situations of finding a parking lot with available spaces and adequate rates. With the second application, the aim is to facilitate the administration of the parking lots, from here the operator or administrator of the parking lot will be able to see the available spaces, the new reservations that arrive, and statistics of the application. The third application is intended to support the first two applications and will be for the use of system administrators.

With these three applications, we intend to provide an attractive solution for drivers of cars, motorcycles, bicycles, and electric scooters in such a way that they can easily find a place to park their vehicles.

4. INTRODUCCIÓN

Propuesta de solución al inconveniente que se puede dar cuando un conductor se encuentra buscando un lugar donde estacionar su vehículo en la ciudad de Bogotá, la solución está compuesta por un grupo de aplicaciones. La finalidad de la primera aplicación es brindar al usuario información de los parqueaderos cercanos a su sitio de destino, y así, éste pueda organizar su camino sin contratiempos, evitar los malos momentos y situaciones donde luego de salir de un trancón comienza la dificultad de encontrar un parqueadero con espacios disponibles y tarifas adecuadas. Con la segunda aplicación, se busca facilitar la administración de los parqueaderos, desde aquí el operario o administrador del parqueadero podrá evidenciar los espacios disponibles, las nuevas reservas que llegan y estadísticas de la aplicación. La tercera aplicación tiene como objetivo brindar un soporte a las dos primeras aplicaciones y será para el uso de los administradores del sistema.

Con éstas tres aplicaciones pretendemos brindar una solución atractiva para los conductores de automóviles, motos, bicicletas y patinetas eléctricas de tal forma que puedan encontrar de forma más sencilla un lugar donde estacionar su vehículo.

5. DESCRIPCIÓN GENERAL

5.1. Oportunidad, Problema

5.1.1. Contexto del problema

En la actualidad los sistemas de reservas funcionan por medio de métodos poco automatizados o en algunos casos inexistentes. Ciertas aplicaciones móviles funcionan únicamente para algunos parqueaderos en Bogotá, ya que son creadas con el propósito de ubicar únicamente los parqueaderos del dueño de la aplicación y no los restantes.

Comprendiendo la movilidad como una actividad de la cual dependen las necesidades básicas de las comunidades y su impacto en el desarrollo de estas, se decide desarrollar una encuesta relacionada con la movilidad en Bogotá, Colombia. La encuesta fue aplicada a personas con las siguientes características: hacen uso de vehículos propios, poseen pase para manejar, tienen acceso a un carro para movilizarse y que disponen de un dispositivo móvil. A partir de esto se identificaron las siguientes preguntas que se generan cuando alguna persona desea desplazarse por la capital colombiana:

- ¿Qué medio de transporte voy a utilizar?
- ¿Debería llevar mi propio vehículo?
- ¿Habrá un lugar donde estacionar mi vehículo?
- ¿A qué horas debo salir para llegar a tiempo?

Se evidenció que la movilidad es un factor generador de estrés constante en los individuos, llevándolos a tener un excesivo control del tiempo; también, se encuentra como razón los parqueaderos, puesto que carecen de un sistema por el cual los usuarios tengan la posibilidad de obtener información de estos, como su ubicación, precio y disponibilidad. A raíz de esto los usuarios tienen problemas para ubicar parqueaderos y sólo se basan en experiencias anteriores esperando que tengan suerte y encuentren un parqueadero disponible.

Por otro lado, el manejo y administración de parqueaderos tampoco es tarea sencilla. Actualmente en Bogotá existen diferentes tipos de parqueaderos con diferentes niveles de sistematización en sus instalaciones. Algunos cuentan con una tecnología muy avanzada lo que permite poca interacción con la persona que lo administra mientras que en muchos otros la tecnología no es tan abundante y los procesos son muy manuales obligando a mucha interacción humana en todas las actividades desde que llega un vehículo hasta que este abandona el establecimiento.

Desde esta perspectiva, podemos evidenciar que hay muchos factores que pueden influir en que tan sencillo y rápido se hace este proceso, no solamente para la persona que llega buscando el servicio sino para la persona que lo ofrece. Considerando el punto de vista del administrador se generan las siguientes cuestiones que pueden surgir durante el proceso:

- ¿Cuál es la capacidad de vehículos que tiene el parqueadero?
- ¿Cuántos vehículos han ingresado en el día?
- ¿Cuántos vehículos han salido?
- ¿Cómo realizar el registro del flujo de vehículos?
- ¿Cuánto se debe cobrar por vehículo?
- ¿Qué horario de atención es beneficioso en la zona en la que se encuentra?

Esto nos da una idea de las necesidades que pueden tener los administradores de los parqueaderos y nos ayuda a entender hacia dónde se debería enfocar la búsqueda de una solución teniendo en cuenta sus necesidades y preocupaciones diarias.

5.1.2. Formulación del problema

¿Cómo facilitar la consulta, ubicación y reserva de establecimientos para estacionar vehículos en la ciudad de Bogotá?

5.1.3. Propuesta de la solución

Con el fin de encontrar una respuesta a esta dificultad ha surgido la idea de una aplicación móvil que tiene como objetivo la gestión, consulta y planificación de establecimientos, orientada a

resolver las necesidades de localizar y reservar parqueaderos en Bogotá, esta tendrá tres perfiles de acuerdo con los stakeholders que se describirán a continuación:

• Aplicación Usuario: esta aplicación será para el usuario final donde este tendrá su perfil y podrá gestionar sus reservas.

• Aplicación Administrador Parqueadero: esta aplicación será para el administrador del establecimiento donde este tendrá su perfil y podrá gestionar sus parqueaderos.

• Aplicación Administrador General: esta aplicación será para la administración y gestión de las otras dos aplicaciones.

Nuestra propuesta es la creación de una aplicación móvil que permita al usuario realizar toda la gestión para estacionar su vehículo, además, que sea una herramienta útil para los administradores a la hora de realizar la gestión de su parqueadero.
Para obtener más información sobre el diseño se puede revisar el Anexo B "Descripción de Diseño del Software - FlashPark (SDD)".

5.1.4. Justificación de la solución

En la actualidad el teléfono móvil se ha convertido en un elemento indispensable para la gestión de las actividades diarias, existen innumerables aplicaciones que nos ayudan en diferentes situaciones del día a día.

El uso de parqueaderos en Bogotá es un tema que se ha convertido en algo que genera mucho estrés en los ciudadanos pues su gestión no es sencilla, comenzando con la difícil localización de ellos, las diferentes tarifas que manejan cada uno además de la falta de seguridad que se puede presentar a la hora de encontrar disponibilidad de cupos.

Actualmente en el mercado existen un gran número de aplicaciones para la reserva de parqueaderos en la ciudad, sin embargo, la calidad de estas no es la adecuada para el usuario

final. Por tal motivo consideramos que la creación de una aplicación orientada a solventar estos inconvenientes puede servir como base a un modelo de negocios rentable.

5.2. Descripción del proyecto

5.2.1. Objetivo general

Desarrollar una aplicación móvil que tiene como objetivo la gestión, consulta y planificación de establecimientos, orientada a resolver las necesidades de localizar y reservar parqueaderos.

5.2.2. Objetivos específicos

- Diseñar el plan de proyecto y el modelo de ingresos.
- Realizar la recolección, análisis y especificación de requerimientos.
- Desarrollar un prototipo funcional de la aplicación enfocado en la gestión de parqueaderos.
- Diseñar y ejecutar pruebas funcionales y de aceptación del prototipo.

6. CONTEXTO DEL PROYECTO

6.1. Soluciones implementadas

Actualmente en el mercado se encuentran varias opciones de aplicación que brindan soluciones parciales a la hora de buscar y reservar parqueaderos, a continuación, describiremos brevemente algunas de estas soluciones:

• Parkopedia es una aplicación para encontrar aparcamiento, conseguir el camino más corto hasta el aparcamiento, ver la disponibilidad de espacio de aparcamiento en tiempo real, comprobar las horas de funcionamiento, precios actualizados, métodos de pago, búsqueda avanzada de aparcamiento usando filtros. Esta aplicación se encuentra en la Play Store y en App Store.

• Parkot es una aplicación que permite buscar, encontrar y reservar un parqueadero. Esta aplicación se encuentra en la Play Store y en App Store.

• Ruedaz es una aplicación que permite disfrutar de la red más amplia de parqueaderos en el país, buscando crear una comunidad de parqueaderos que permita a sus miembros estacionar de manera ilimitada, por un bajo costo mensual. Esta aplicación se encuentra en la Play Store y en App Store.

• Parking App es una aplicación que ofrece las opciones de buscador, localizador, cómo llegar, favoritos, reservas y mensualidades de una cadena específica de parqueaderos. Esta aplicación se encuentra en la Play Store y en App Store.

• Rent a parking es una aplicación donde los usuarios ofrecen y alquilan parqueaderos fácilmente. Esta aplicación se encuentra en la Play Store.

• Queo es una aplicación de ingreso a edificios, oficinas o residencias. Esta aplicación se encuentra en la Play Store y en App Store.

• Nidoo es una aplicación que permite encontrar un parqueadero cercano a la ubicación. Esta aplicación se encuentra en la Play Store y en App Store.

• Wheels house está pensado para ofrecer y encontrar parqueaderos. Tiene página web pero no se encuentra en funcionamiento.

• Parkcero App para publicar, encontrar, reservar y pagar parqueaderos. Esta tiene página web, dice que la aplicación está próxima a salir, pero no se encuentra ni en la Play Store ni App Store.

• Parkiando se trata de una aplicación móvil que permite ubicar un parqueadero fácil y rápidamente, según el sector donde uno se encuentre. Esta aplicación fue la ganadora de Apps.co en 2012 pero actualmente no está en funcionamiento.

6.2. Análisis poblacional

Se definió como población objetivo a personas residentes en la ciudad de Bogotá que tengan un teléfono móvil y un vehículo. El teléfono móvil es indispensable para descargar la aplicación, obtener una ruta y reservar en el parqueadero a donde se dirigirán a estacionar su vehículo.

7. FACTIBILIDAD TÉCNICO-ECONÓMICA

Este trabajo de grado se basa y busca dar continuidad al trabajo de grado de ingeniería industrial denominado “TECHMOB Solutions: empresa de desarrollo de soluciones móviles para ubicación, manejo de información y reservas de establecimientos comerciales tales como parqueaderos” realizado por Maria y Bernal (2016). En este se plantea el montaje de un startup estableciendo además un modelo financiero con el fin de mostrar la viabilidad del proyecto. De acuerdo con el análisis financiero realizado en dicho trabajo de grado la rentabilidad del proyecto es de un 38.76% dando así un margen de ganancia bueno a los inversionistas.

8. DISEÑO DE LA SOLUCIÓN

8.1. Roles y responsabilidades

La asignación de roles se ha establecido de acuerdo con las habilidades y preferencias de cada uno de los integrantes del equipo de trabajo. Así mismo, se desglosan las responsabilidades asociadas a cada rol pues

es importante que cada integrante tenga pleno conocimiento de ellas, logrando así, que se desarrolle cada actividad sin ambigüedades ni discusiones al momento de asignarlas.

Tabla 1: Roles y responsabilidades

ROL	RESPONSABILIDADES	RESPONSABLE
Gerente de Proyecto	- Coordinación, administración y asignación de actividades, planes y recursos. - Planeación y verificación de cronograma.	Daniel Esteban Valero Galeano
Director de calidad y manejo de riesgos	- Revisión y verificación de la documentación y plan de calidad. - Administración de riesgos.	Daniel Esteban Valero Galeano
Administración de configuración y documentación	- Verificar la organización de documentación y su actualización. - Creación y actualización del repositorio, como el control y definición de acceso a los archivos contenidos en él. - Creación de plantillas y formatos.	Viviana Andrea Amaris Balda
Director de Diseño	- Generación del diseño y prototipos del sistema.	Viviana Andrea Amaris Balda
Jefe de Desarrollo	- Conversión de especificaciones y requerimientos en código ejecutable con su debida documentación. - Reducción de complejidad del software. - Modificación de errores del código. - Elección de ambiente en el que se desarrollará la aplicación. - Desarrollo de la aplicación.	Sergio Posada Henao
Arquitecto	- Desarrollo de la arquitectura del proyecto.	Daniel Esteban Valero Galeano

Administrador de Pruebas	- Detección y eliminación de errores y defectos del sistema. - Asegurar la calidad del código a entregar. - Construcción, ejecución y documentación de planes de prueba.	Daniel Esteban Valero Galeano y Sergio Posada Henao

Fuente: Autoría propia

Tabla 2: Proceso de cambios

ACTIVIDAD	DESCRIPCIÓN	RESPONSABLE
Integración de Secciones	Al inicio del proyecto se realiza la división de secciones del PMP. Esta tarea revisa e integra las mismas.	Gerente de Proyecto, Administración de configuración y documentación
Revisión	Se realizarán 2 revisiones, una finalizando la integración del documento y la segunda en el cierre de la entrega.	Director de calidad y manejo de riesgos, Administración de configuración y documentación
Modificaciones	Se realizarán modificaciones teniendo en cuenta las conclusiones u observaciones realizadas en las revisiones.	Director de calidad y manejo de riesgos, Administración de configuración y documentación, Jefe de desarrollo

Fuente: Autoría propia

8.2. Levantamiento de requerimientos

El levantamiento de requerimientos detallado se podrá encontrar en el Documento de Especificación de requerimientos (SRS), en donde se especifican los requerimientos de software definiendo el comportamiento del sistema de reservas FlashPark. Dentro de este documento se encontrará el alcance de cada una de las aplicaciones que harán parte del proyecto, así como una explicación detallada de cada una de las funcionalidades que el MVP debe cumplir para que el desarrollo del proyecto sea satisfactorio. Con el fin de realizar este documento de una manera eficiente se evaluaron cada una de las aplicaciones y su comportamiento deseado con el fin de separarlas por partes y lograr identificar cada uno de los requerimientos pertinentes. En el documento se plantean 38 casos de uso, 3 subsistemas, 135 requerimientos funcionales; 44 de los cuales corresponden al administrador del parqueadero, 75 que corresponden al usuario dueño de vehículo que busca el servicio de parqueadero y 45 que corresponden al gestor de aplicaciones.

8.3. Elección de arquitectura

El auge de las tecnologías móviles ha permitido dar solución a problemas y necesidades básicas de los individuos y/o poblaciones, aumentando así el número de usuarios alrededor del mundo y generando de manera simultánea múltiples oportunidades de emprendimiento, creación e innovación alrededor de estas aplicaciones y sistemas inteligentes. Basados en esto y teniendo en cuenta que se estima que el 96% de los conductores de vehículo portan un dispositivo móvil, generalmente celular o Tablet se ha tomado la decisión de hacer una aplicación móvil tomando en cuenta además que como el usuario se va a estar desplazando una solución móvil es más conveniente para él pues puede estar haciendo seguimiento tanto de la ruta como de la reserva en todo momento.

8.4. Metodología de implementación

El desarrollo del producto se basará en dos metodologías ágiles: Lean Startup para la parte del plan de negocios y Scrum que se aplicará a la creación del prototipo y del producto final de la aplicación.

1) Lean Startup: Es una metodología que tiene como objetivo la definición de un modelo de negocio de calidad y escalable por medio de una serie de pasos, estos pasos tienen como objetivo

maximizar recursos y evitar desperdicios. Estos pasos se describen a continuación y se realizan de manera iterativa:

• Primer paso: consiste en una lluvia de ideas, posterior a seleccionar la idea esta se debe plasmar en el modelo Canvas, para así lograr identificar recursos, herramientas y características de dicha idea.

• Segundo paso: se fundamenta en la construcción, en este se define un producto mínimo viable (MVP), el cual es el producto que cumple con un conjunto mínimo de funcionalidades que permite validar el producto.

• Tercer paso: este se centra en el producto, consiste en el desarrollo del MVP basándose en la documentación, se aplicará la metodología Scrum.

• Cuarto paso: se concentra en la medición y verificación del producto por medio de preguntas y encuestas de satisfacción a los clientes y usuarios.

• Quinto paso: el foco es el aprendizaje, se aprende acerca de lo realizado y se propone funcionalidades futuras de dicho producto.

2) Scrum: Es una metodología ágil en donde se aplican un conjunto de buenas prácticas con el fin de trabajar colaborativamente y de esta forma poder obtener el mejor resultado posible. En esta metodología se realizan entregas parciales y continuas del producto incluyendo las siguientes características:

• Pila de producto: se componen de una lista de actividades y tareas cortas que se tienen que alcanzar durante el proyecto, estas se organizan por orden de prioridad.

• Sprint: es el tiempo en que se debe cumplir las actividades planteadas en la pila, su función es ver la fecha en la que se cumple dicha actividad.

• Daily Scrum (Scrum Diario): son todas las actividades que se hacen a diario.

• Producto Final: en la parte final de cada Sprint se realiza una reunión donde se analiza y se muestra lo que se lleva hasta el momento.

8.5. Elección de lenguajes y herramientas base para el desarrollo de la solución -

8.5.1. Flutter y Dart como herramientas de implementación

Flutter es un framework de Dart para crear aplicaciones multiplataforma con un único código, uno de nuestros requerimientos más importantes. A diferencia de otros frameworks multiplataforma, el código de una aplicación de Flutter se compila a código nativo, por lo que el rendimiento alcanzado es superior a aplicaciones basadas por ejemplo en web-views (Ionic).

Además, Flutter no utiliza componentes nativos, sino que viene con sus propios componentes, llamados widgets, por lo que la misma aplicación se verá igual en cualquier dispositivo, independientemente de su sistema operativo o la versión ("¿Qué es el lenguaje de programación Dart?", 2021).

Gracias a ello, FlashPark tiene un bajo consumo de recursos de memoria y procesamiento lo cual permite que sea utilizando adecuadamente en dispositivos antiguos, ahorrando tiempo de desarrollo en renderización, cabe destacar que su estructura es fácil de comprender y gracias a su lenguaje flexible y orientado a objetos (Dart) se pueden aplicar muchos conceptos aprendidos a lo largo de nuestra formación.

8.5.2. Firebase como base de datos de la aplicación

Firebase es una plataforma en la nube para el desarrollo de aplicaciones web y móviles. Está disponible para distintas plataformas (iOS, Android y web), con lo que es más rápido trabajar en el desarrollo. Además, no es un simple motor de bases de datos, ya que Firebase proporciona servicios que facilitan aún más el desarrollo de aplicaciones móviles, algunas como ("Qué es Firebase: funcionalidades, ventajas y conclusiones", 2021):

Realtime database: Firebase proporciona servicio de base de datos en tiempo real en la nube, es No SQL y almacena los datos como JSON.

Autenticación de usuarios: debido a que FlashPark depende de la creación de cuentas y autenticación de usuarios Firebase es una excelente opción, ya que provee este servicio ya sea por medio de email y contraseña o recuperando la información de plataformas externas

como lo son Google, Twitter y Facebook, además de mantener las sesiones gracias a su almacenamiento en tiempo real, adicionalmente, provee servicios de autenticación de cuenta, recuperación de contraseña, entre otros.

Crash reporting: FlashPark es una aplicación que responde a los errores gracias a este servicio de Firebase en el cual se realiza seguimiento de errores y el funcionamiento general de la aplicación, este servicio detecta los errores y genera un informe muy detallado y organizado ya que los agrupa por similitud y por gravedad.

Test lab: Gracias a esta funcionalidad, hemos garantizado el buen funcionamiento de FlashPark apps, ya que permite realizar pruebas en diferentes dispositivos con el fin de detectar errores y poder solucionarlos.

8.5.3. Cuadro comparativo de herramientas

Se realizó un estudio de los frameworks y lenguajes más populares teniendo en cuenta el desarrollo, la seguridad, la experiencia de usuario y que permitan el despliegue multiplataforma, tras el estudio, se destacaron 3 frameworks: Flutter, React Native y Xamarin. A continuación, se puede apreciar un cuadro comparativo para obtener más detalle:

Framework	Ventajas	Desventajas
Flutter	**Rendimiento de la aplicación:** Flutter usa Dart, un lenguaje compilado anticipadamente (AOT) que permite que la aplicación se comunique directamente con la plataforma nativa en lugar de pasar por un puente JavaScript como en React Native. Esto permite a los desarrolladores crear aplicaciones complejas sin afectar el rendimiento y los tiempos de inicio. **Recarga en caliente:**	**Proyecto joven:** La primera y mayor desventaja es lo reciente del framework. Aunque Flutter se puso a disponibilidad de uso en producción en mayo de 2018, el proyecto aún necesita un mayor período de asentamiento. No hay forma de saber con certeza si el framework tendrá éxito finalmente o no, y por lo tanto, no hay forma de saber hacía que dispositivos estarán enfocadas las aplicaciones

	Esta característica permite a los desarrolladores reconstruir la aplicación al instante, como si fuera una página web. Además, la recarga en caliente de Flutter tiene estado, lo que significa que los desarrolladores no necesitan reiniciar la aplicación cada vez que cambian algo, simplemente continúa donde se quedó. Eso hace que el diseño de Flutter sea visible de inmediato. **Un conjunto completo de widgets únicos:** Flutter no se basa en componentes de interfaz de usuario específicos de la plataforma, si no que posee sus propios widgets. El framework implementa widgets del kit de Material Design para Android y ofrece widgets de Cupertino para iOS, pero no tiene que ser estrictamente, así como se usen. Si deseas utilizar los widgets de Cupertino sobre android funcionarán y se verán increíblemente y viceversa. **Gráficos 2D:** Flutter también implementa Skia, una biblioteca de gráficos 2D de código abierto, que se encarga de renderizar la biblioteca de los componentes de la interfaz de usuario integrada en todo el framework. **Todo es un widget:** En Flutter, cada elemento de la pantalla es un widget, lo que simplifica enormemente el diseño de la aplicación. Cada widget separado especifica su	desarrolladas con Flutter hoy en día, dentro de algunos años. **Soporte de terceros:** Si bien mencionamos que Flutter ya cuenta con una impresionante colección de bibliotecas y herramientas. Sin embargo, todavía hay muchos servicios de terceros que no han comenzado a admitir Flutter, incluidos los sistemas de pago, el software de automatización de impuestos e incluso Apple y Android TV. Ahora se están desarrollando muchas herramientas y aún quedan muchas más por desarrollar, pero definitivamente tomará tiempo antes de que Flutter se ponga a la altura de React Native en cantidad de herramientas y disponibilidad de paquetes con los que desarrollar.

	propio modelo de diseño, en lugar de tener un solo conjunto de reglas para todos los widgets. Debido a que el diseño de Flutter es relativamente pequeño, es fácil de optimizar, y debido a que cada elemento de la interfaz de usuario es un widget, todo el diseño en sí; se vuelve más manejable. **Gran cantidad de paquetes disponibles:** Aunque Flutter es de creación reciente, los desarrolladores de todo el mundo están realmente entusiasmados con el proyecto; y por ello, ya existen muchos paquetes diferentes disponibles para trabajar con Flutter. A saber; herramientas para manejar imágenes, solicitudes HTTP, conexiones WebSocket, varios clientes de protocolo de red, notificaciones push, bases de datos integradas, acceso a sensores, acceso a las cámaras del dispositivo, etc. **Compatibilidad de dispositivos:** Los widgets nativos de Flutter permiten que las aplicaciones sigan siendo compatibles con las versiones del sistema operativo a partir de iOS 8.0 y Android Jelly Bean. Siempre que Apple o Google presenten un nuevo widget, no interrumpirá su aplicación desde el exterior porque Flutter no toca los widgets de la plataforma nativa.	
React Native	**Una gama amplia de componentes:** Incluye una amplia selección de componentes nativos de Interfaz de Usuario a diferencia de otros frameworks	**El rendimiento de la aplicación:** Debido a que las aplicaciones multiplataforma no están totalmente alineadas con el hardware del dispositivo,

	de desarrollo multiplataforma. Esto significa que las aplicaciones se verán como aplicaciones nativas. Si se hace correctamente, una aplicación React Native apenas se distingue de una aplicación nativa. **Desarrollo simplificado:** El desarrollo móvil nativo implementa el enfoque de programación imperativo, donde los desarrolladores necesitan definir estrictamente una secuencia de acciones que describan cómo funciona la aplicación para crear una interfaz de usuario. Con React Native, los desarrolladores pueden usar programación declarativa, que describe solo lo que el programa tiene que hacer y no cómo debe hacerse. La programación declarativa permite un código más simple y una codificación más fácil, lo que también significa que el producto final será más fácil de mantener. **Arquitectura modular:** React Native admite la modularidad de la arquitectura, lo que permite separar el código de la aplicación en varios bloques independientes. Esto da como resultado un desarrollo flexible, lo que facilita la actualización y modificación del producto final. **Soluciones y bibliotecas listas para usar:** Debido a que React Native ha existido durante bastante tiempo, los desarrolladores han tenido tiempo de	su rendimiento es peor que el de las aplicaciones nativas. La razón es que hay un puente basado en JavaScript entre la capa de aplicación de React Native y los componentes de hardware, y cada interacción con el dispositivo tiene que pasar por ese puente. Cuantas más interacciones haya, peor será el rendimiento de la aplicación. Para preservar el rendimiento de la aplicación, los desarrolladores deben minimizar las interacciones entre la aplicación y el dispositivo, lo que básicamente significa que la aplicación debe ser lo más simple posible sin ninguna interactividad exagerada contra el hardware. **La necesidad de desarrolladores nativos:** React Native es un gran framework multiplataforma, pero tiene sus limitaciones. Cuando encuentre una de esas limitaciones y aún no haya una solución React Native, es necesario involucrar a los desarrolladores acostumbrados a crear aplicaciones nativas, a veces tanto que es imposible justificar el uso de React Native en primer lugar. **Incapacidad para lidiar con la complejidad:** React Native es bueno para aplicaciones simples, pero no es tan bueno para aplicaciones con muchas pantallas, interacciones, transiciones y animaciones complejas.

	crear una variedad de bibliotecas y herramientas para automatizar y ayudar a la mayoría de los procesos de rutina. Existen frameworks de pruebas, herramientas para la verificación de tipos, tecnologías para la configuración del flujo de trabajo, etc. Esto significa que a los desarrolladores les resultará más fácil resolver problemas complicados o mundanos y tendrán más tiempo para perfeccionar el producto. **Soporte de terceros:** Debido a que la comunidad de desarrollo ha respaldado los beneficios de React Native, muchos servicios de terceros han creado API y complementos para las aplicaciones de React Native. Hay soporte de terceros para mapas, sistemas de pago, gráficos y más. **Permite la recarga en caliente:** Al desarrollar una aplicación móvil, es útil mirar la aplicación real con frecuencia (posiblemente después de cada cambio realizado en el código) para ello existe la denominada: recarga en caliente. Esta característica presente en React Native, permite a los desarrolladores ver de inmediato los cambios que se realizan en el código cada vez que se guardan. **Existe un gran ecosistema y comunidad:** Hay una comunidad importante alrededor del framework React Native, que es una plataforma de código abierto; lo que se convierte en una gran ventaja porque significa que es menos probable que los	**Lento al iniciarse:** Las aplicaciones nativas React tardan más en iniciarse, incluso con dispositivos de gama alta, porque el puente de JavaScript también tarda en iniciarse. **Con visión de apps nativas:** Al buscar desarrolladores de React Native, también debe buscar a alguien que tenga experiencia en desarrollo de apps nativas. De lo contrario, las posibilidades ya limitadas de React Native se harán aún más evidentes ya que el equipo no podrá comprender soluciones complejas a los desafíos que involucren hardware nativo que otros desarrolladores han creado con anterioridad. **Actualizaciones de plataforma nativas:** Para mantenerse relevantes, Google y Apple tienen que seguir agregando constantemente nuevas funciones a sus plataformas. Aunque el equipo de React Native ha estado haciendo todo lo posible para mantenerse al día con las nuevas características de hardware, todavía se encuentran un paso por detrás que los proyectos de las anteriores citadas. Aún hoy, lleva mucho más tiempo desarrollar una aplicación React Native que desarrollar una aplicación basada en una solución nativa. Por lo tanto, es probable que además, su aplicación React Native se quede atrás siempre que exista una nueva actualización de hardware.

	desarrolladores se atasquen, ya que siempre hay alguien a quién se le puede pedir ayuda.	
Xamarin	**Rendimiento comparable a las aplicaciones nativas:** Gracias a C#, que es un lenguaje de programación multi-paradigma y fuerte competidor de otros lenguajes utilizados en el desarrollo móvil, lo que permite que las aplicaciones Xamarin se mantengan firmes junto con las aplicaciones desarrolladas en Objective-C, Swift, Java o Kotlin. **Consistencia con el hardware:** La variedad de APIs disponibles para los desarrolladores de Xamarin permite la integración con muchos componentes de hardware nativos, mejorando la experiencia del usuario. **Xamarin.Forms:** Esta tecnología permite a los desarrolladores escribir código de interfaz de usuario que se puede compartir entre aplicaciones de Android e iOS con un conjunto de herramientas de más de 40 controles y diseños que se asignan a controles nativos durante el tiempo de ejecución. **Compatibilidad con la arquitectura MVC y MVVM:** Xamarin admite la implementación de dos patrones de arquitectura populares:	**Los plazos de carga:** Al crear con Xamarin, los desarrolladores deben vincular y hacer referencia al código entre varios sistemas operativos y frameworks .NET. Esto afecta negativamente los tiempos de inicio y descarga de la aplicación, lo que hace que la experiencia de la aplicación sea menos agradable para el usuario final. **Desarrollo lento y complejo de la interfaz de usuario:** Aunque Xamarin.Forms permite compartir algunos componentes de la interfaz de usuario, todavía hay algunos componentes que deben desarrollarse para cada plataforma individualmente. Aunque en algunos casos es mejor que escribir la interfaz de usuario desde cero, el proceso de desarrollo de la interfaz de usuario con Xamarin sigue siendo difícil y requiere de mucho tiempo. **Complejidades en el soporte del código:** Aunque hay muchos desarrolladores .NET profesionales, Xamarin también requiere familiaridad adicional con el desarrollo, los frameworks y las arquitecturas móviles nativos. Esta es una combinación de habilidades difícil de encontrar, ya que no muchos desarrolladores quieren aprender dos

	Model – View – ViewModel (MVVM) y Model – View – Controller (MVC). MVVM es bueno para crear diferentes procesos con la misma base de código, mientras que MVC ayuda a separar la presentación y la lógica de la aplicación, acelerando el proceso de desarrollo. **Soporte de aplicaciones sin conexión:** Debido a sus características de sincronización de datos y nube, Xamarin permite que las aplicaciones trabajen sin conexión, una capacidad que solía estar limitada solo a las aplicaciones nativas.	paradigmas de programación muy diferentes. Como resultado, es posible que existan dificultades para encontrar desarrolladores profesionales que brinden soporte, arreglen, actualicen o desarrollen una aplicación. **Dependencia de la aplicación:** Independientemente de cuánto código comparte entre las diferentes plataformas, los desarrolladores siempre necesitarán escribir código nativo para la personalización específica de cada plataforma. Esto no solo podría ser un problema en términos de costo y de encontrar desarrolladores apropiados: si no que, además, una combinación entre Xamarin y código nativo también resultaría en archivos de gran tamaño, lo que terminará por dañar el rendimiento de la aplicación.

Teniendo en cuenta el cuadro comparativo anterior, el quipo Flashpark toma la decisión de implementar la solución a través de Flutter puesto que sus ventajas favorecen la mayoría de los requerimientos y sus desventajas no afectan a gran escala el desarrollo de este proyecto.

En comparación a los otros frameworks que tuvimos presentes, descartamos React Native por el hecho de que necesitamos una aplicación con bajo consumo de recursos tanto en almacenamiento como en rendimiento, entre una de las desventajas de este Framework está el rendimiento, además de que se recomienda que el desarrollador tenga conocimientos de desarrollo nativo, por lo que representa problemas a largo plazo con los desarrolladores de FlashPark.

Con respecto a Xamarin, otro framework que también consideramos para la implementación de las aplicaciones de FlashPark, fue descartada por que al igual que React presenta problemas en temas de

rendimiento, sumado a que la experiencia del usuario se ve afectada ya que es complejo la implementación de Xamarin Forms y, además, también requiere de experiencia con desarrollo nativo.

9. IMPLEMENTACIÓN DE LA SOLUCIÓN

9.1. Mockups

9.1.1. FlashPark

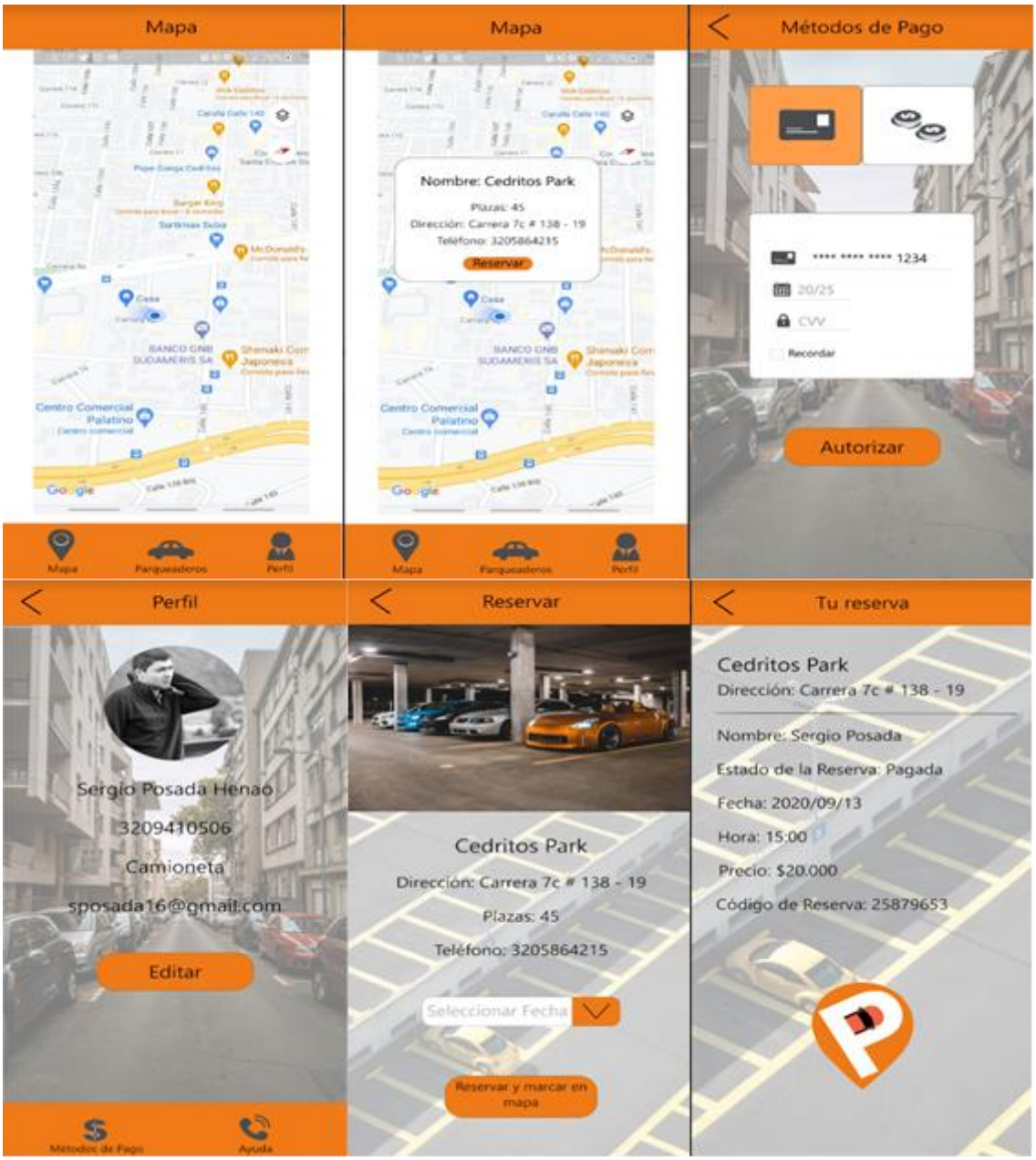

Ilustración 1: Mockups FlashPark

9.1.2. FlashPark Partner

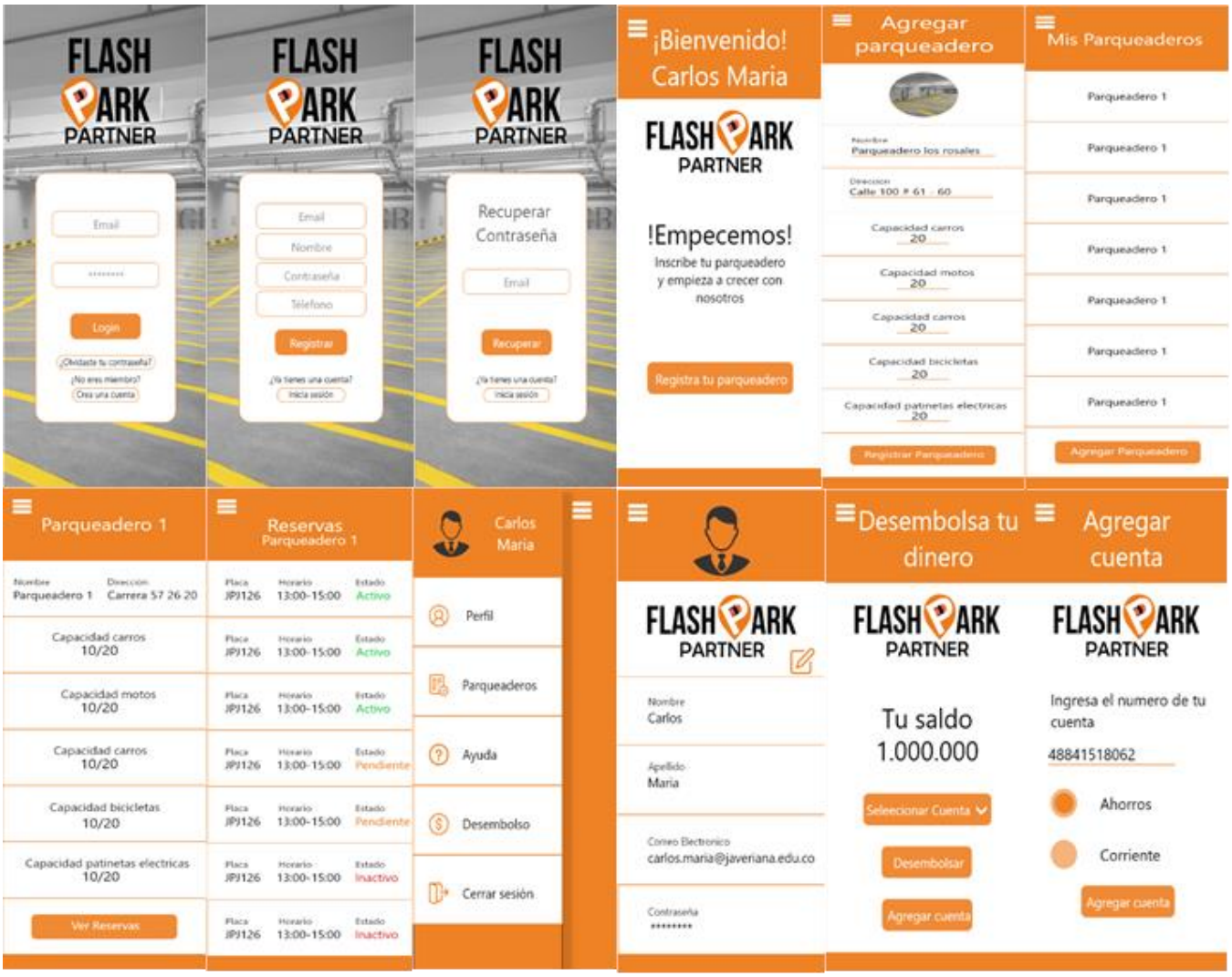

Ilustración 2: Mockups FlashPark Partner

9.2. Aplicaciones

9.2.1. FlashPark

Esta aplicación será dirigida al conductor del vehículo que busca dónde estacionar. Con esta aplicación él podrá encontrar un parqueadero, obtener información de este, además, la opción de reservar un espacio en el establecimiento para poder tener la seguridad de tener un lugar donde estacionar a la hora de llegar a su destino. Una vez seleccionado el parqueadero a donde se dirigirá, la aplicación mostrará un mapa que le permitirá saber la ruta para llegar a su destino.

9.2.2. FlashPark Partner

Esta aplicación será dirigida a los administradores y operarios de parqueaderos. A través de la aplicación podrán recibir solicitudes de reserva de los usuarios de la aplicación FlashPark y tendrán la opción de aceptarla o no. También, tendrá un módulo de control y estadísticas con el cual podrá hacer la administración de espacios de su establecimiento y saber cuánto se ha ganado con la aplicación. En esta aplicación se contará con la opción de mostrar o no ciertos módulos dependiendo del tipo de usuario que la vaya a manejar, se pueden otorgar permisos de acuerdo con si la persona que tiene la aplicación es el administrador o el operario del parqueadero.

9.2.3. FlashPark Admin

Esta última aplicación está destinada para el uso administrativo del producto, desde esta aplicación se controlarán y dará soporte a las aplicaciones FlashPark y FlashPark Partner.

9.3. Pruebas

Para llevar un registro consistente de las pruebas realizadas se procedió a utilizar como base una plantilla enfocada en la documentación de las pruebas para un proyecto del software el cual se puede evidenciar en el Anexo E Plan de pruebas. En este documento se define a profundidad el alcance que se va a tener sobre las pruebas y qué requisitos se están abarcando dentro de estas,

también se explica en detalle la metodología que se realizó sobre estas y los diferentes tipos de prueba que se tuvieron en cuenta. Por último, se muestra una tabla con cada una de las funcionalidades las cuales pasaron por su respectivo proceso de pruebas, y según los resultados obtenidos sobre cada una de las funcionalidades, se decidió si aceptarla o si se rechazaba iniciando nuevamente el proceso de prueba.

9.4. Documentación técnica

Con el fin de llevar a cabo el proyecto de grado FlashPark cumpliendo con las normas y requerimientos establecidos por la dirección de carrera, se realizaron diferentes documentos los cuales fueron la base para un desarrollo y documentación correctos a lo largo de los semestres los cuales se utilizaron para la implementación de este.

Para iniciar, se desarrolló un documento denominado Plan de Administración del Proyecto (PMP) el cual se puede evidenciar en el Anexo C Plan de Proyecto - FlashPark (SPMP). En este documento se incluye todo el proceso de análisis, diseño, implementación y validación del producto que se realizó en el MVP.

Una vez realizado y revisado el documento anterior como base principal para el desarrollo del proyecto, se procedió a definir los diferentes tipos de requerimientos que las aplicaciones necesitarían para cumplir con los objetivos planteados por el grupo de trabajo. Estos requerimientos se realizaron en la Especificación de Requerimientos (SRS), este documento se puede evidenciar en el Anexo D Especificación de Requerimientos - FlashPark (SRS) y este es el que provee información detallada sobre las diferentes interfaces que se realizaron, las restricciones de las aplicaciones, las características de los usuarios que utilizarán el producto y, por último, el modelo de dominio que se implementó.

Después de tener claro el alcance y prioridad de cada uno de los requerimientos se llevó a cabo una etapa de análisis la cual consistió en la producción de diferentes diagramas los cuales muestran la composición del producto a un nivel más técnico con el fin de que estos sirvieran como base para empezar el desarrollo de las diferentes aplicaciones. Con el fin realizar esto se desarrolló el documento denominado Descripción de Diseño del Software (SDD) el cual se puede ver en el Anexo B Descripción de Diseño del Software - FlashPark (SDD) donde se podrá

evidenciar la arquitectura del sistema según diferentes vistas técnicas y también el diseño detallado de toda la estructura que se utilizó para desarrollar el código.

Por último, se desarrolló en el lenguaje de programación Flutter ambas aplicaciones cumpliendo con los requerimientos establecidos previamente según su prioridad y teniendo en cuenta la arquitectura definida con el fin de cumplir con los objetivos del Producto Mínimo Viable.

9.5. Manuales

Con el fin de dar una guía en el manejo de las aplicaciones, cada una contará con un botón de preguntas frecuentes en donde se enconarán cortos videos demostrando las funciones principales.

10. RESULTADOS

Como producto final del proyecto de grado se desarrollaron dos aplicaciones siguiendo los lineamientos especificados en cada uno de los documentos entregados junto con este y cumpliendo los estándares propuestos por el equipo de trabajo.

En primer lugar, se obtuvo la aplicación denominada FlashPark la cual le permite a cualquier persona que tenga en su posesión un Smartphone registrarse, iniciar sesión, navegar a través del mapa, ver parqueaderos, realizar reserva y solicitar navegación hasta cada uno de estos.

Por otra parte, se desarrolló la aplicación destinada a los dueños de los parqueaderos sobre la cual estos pueden registrar toda la información pertinente a estos y así lograr controlar de una manera efectiva los diferentes clientes que se pueden presentar a través de la aplicación.

Cabe resaltar que cada una de estas aplicaciones pasó por un control de pruebas y calidad sobre cada una de sus funcionalidades con el fin de identificar los posibles errores que se presentaron a lo largo de todo el desarrollo y así, al terminar el desarrollo, lograr aceptar cada una de estas aplicaciones.

11. CONCLUSIONES

11.1. Análisis de impacto del proyecto

- Una vez finalizado el desarrollo se logró determinar que es viable realizar un emprendimiento dirigido a resolver la problemática descrita en este proyecto y haciendo

uso de las herramientas y metodologías que se definieron desde el inicio debido a que, al hacer un uso eficiente y correcto, el proyecto se ajusta a los costos presupuestados.

- La organización de los documentos fue de vital importancia desde la planeación del proyecto de grado y se logró realizar de una manera eficiente en Sistemas de Información colaborativos que permitieron el trabajo en grupo, sincrónico y asincrónico, como Microsoft Teams, Google Drive, Trello y Jitsi meet. En las aplicaciones de Microsoft Teams y Google Drive se tenía toda la información y documentación relevante al proyecto.
- Durante el desarrollo del proyecto se evidenció la necesidad de utilizar herramientas centralizadas que permitieran controlar el código y manejar sus versiones, debido a que se presentaron problemas como la incompatibilidad de dependencias y problemas de integración con el motor de bases de datos utilizado.
- Se logró evidenciar la importancia de contar con el apoyo de un director de proyecto y de brindar una revisión semanal a los compromisos de cumplimiento de metas. En la fase previa a iniciar el proyecto existieron serios problemas de comunicación los cuales fueron solucionados al tener una dirección adecuada.
- Para la aplicación del cliente, se tomó la decisión de no implementar los mecanismos de recaudo de dinero en el MVP debido a tres razones fundamentales: la primera, debido a problemas legales relacionados con la captación de dinero, la segunda, debido a que las plataformas de pago en línea exigían requisitos de personería jurídica con los que no se contaba y por último debido a la extensión de la implementación la cual no podría llevarse a cabo dada la ausencia del cuarto integrante del equipo de trabajo quien, por problemas personales no pudo participar durante éste semestre en el desarrollo del proyecto.
- Gracias a que se tuvo un desarrollo eficiente del documento sobre el Plan del Proyecto (PMP) cuando el integrante Carlos María abandonó el proyecto se pudo realizar una sencilla redistribución del trabajo que este integrante tenía asignado, a pesar de que generalmente estos problemas suelen tardar mucho tiempo en resolverse.

- A pesar de que se tuvo en cuenta la seguridad de las cuentas de los usuarios por medio de verificación de dos pasos, y autenticación de correo electrónico, se evidenció una falta a la hora de incluir la seguridad en el diseño de la aplicación.

11.2. Trabajo futuro

Los requerimientos que quedaron pendientes por desarrollar para completar con el objetivo final del proyecto serán enumerados a continuación:

- Tener en cuenta la seguridad de la aplicación en la corrección del diseño y realizar las debidas pruebas con el fin de reducir las amenazas en temas de seguridad.
- Realizar un plan de pruebas más robusto, agregando pruebas para medir los requerimientos no funcionales asegurando así la estabilidad de la aplicación.
- La tercera aplicación que consiste en el uso administrativo del producto para dar soporte a las otras aplicaciones implementadas.
- Permitir al usuario calificar su experiencia en el parqueadero, así como al dueño de este.
- Permitir al usuario ver el promedio de sus calificaciones.
- Permitir al usuario reportar cualquier inconveniente sobre el parqueadero.
- El sistema debe permitir códigos de descuento.
- El sistema debe permitir bloquear cuentas de usuarios y partners si estas no cumplen con los estándares definidos.
- El sistema debe contar con publicidad y convenios.
- A la hora de terminar con las aplicaciones en su totalidad es necesario reevaluar nuevamente un plan financiero con los costos reales de todo el desarrollo para asegurar la viabilidad del proyecto y tener un porcentaje de utilidad real.

12. REFERENCIAS

¿Qué es el lenguaje de programación Dart? (2021). Retrieved 8 November 2021, from https://inlab.fib.upc.edu/es/blog/que-es-el-lenguaje-de-programacion-dart

Maria Hernández, C., & Bernal Clavijo, R. (2016). Techmob Solutions: empresa de desarrollo de soluciones móviles para ubicación, manejo de información y reservas de establecimientos comerciales tales como parqueaderos. Obtenido de https://repository.javeriana.edu.co/handle/10554/38365

Qué es Firebase: funcionalidades, ventajas y conclusiones. (2021). Retrieved 8 November 2021, from https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/

Flutter vs React Native vs Xamarin for Cross Platform Development (2018). Retrieved 27 November 2021, from https://hackernoon.com/flutter-vs-react-native-vs-xamarin-for-cross-platform-development-5f92cfb178ff

React Native, Flutter, Xamarin: a comparison (2018). Retrieved 28 November 2021, from https://blog.novoda.com/react-native-flutter-xamarin-a-comparison/

Comparación entre 3 de los framework web más de moda en 2020: Flutter, React Native y Xamarin (2020). Retrieved 26 November 2021, from https://ciberninjas.com/comparacion-flutter-react-native-xamarin/

13. ANEXOS

ANEXO	**NOMBRE**
Anexo A	Techmob Solutions: empresa de desarrollo de soluciones móviles para ubicación, manejo de información y reservas de establecimientos comerciales tales como parqueaderos
Anexo B	Descripción de Diseño del Software - FlashPark (SDD)
Anexo C	Plan de Proyecto - FlashPark (SPMP)
Anexo D	Especificación de Requerimientos - FlashPark (SRS)
Anexo E	Plan de pruebas