FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

APP MÓVIL DE RENTA DE AUTOS ONLINE ENTRE PARTICULARES MEDIANTE GPS

AUTOR

Ronald Paúl Arias Sánchez

AÑO

2020

FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

APP MÓVIL DE RENTA DE AUTOS ONLINE ENTRE PARTICULARES MEDIANTE GPS

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Sistemas de Computación e Informática

Profesor Guía
MSc. Bernarda Cecibel Sandoval Romo

Autor
Ronald Paúl Arias Sánchez

Año
2020

DECLARACIÓN DEL PROFESOR GUÍA

Declaro haber dirigido el trabajo, App móvil de renta de autos online entre particulares mediante GPS, a través de reuniones periódicas con el estudiante Ronald Paúl Arias Sánchez, en el semestre 202010, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.

Bernarda Cecibel Sandoval Romo

Mestra En Ciencia Da Computacao

CI: 1709974453

DECLARACIÓN DEL PROFESOR CORRECTOR

Declaro haber revisado este trabajo, App móvil de renta de autos online entre particulares mediante GPS, del estudiante Ronald Paúl Arias Sánchez, en el semestre 202010, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación.

Paulo Roberto Guerra Terán

Máster Universitario en Software y Sistemas

CI: 1002856050

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes.

Ronald Paúl Arias Sánchez

CI: 1721207270

AGRADECIMIENTOS

A mis padres y su valioso esfuerzo.

RESUMEN

En este proyecto se realizó la implementación y desarrollo de una aplicación móvil iOS programada en Swift, utilizando la infraestructura en la nube de Google Cloud Platform, Firestore como motor de base de datos y Firebase para el manejo y autenticación de usuarios. Se utilizó el propio gestor web de la plataforma para su respectiva administración. En cuanto al objetivo de la aplicación móvil, se buscó el alquiler y renta de autos entre personas particulares. Para cumplir con la meta planteada, se permitió el registro de usuarios en la aplicación móvil, los cuales pueden ser propietarios de vehículos, quienes pueden ofrecer sus vehículos en el caso de querer generar ingresos económicos, o, personas que buscan rentar un vehículo. Los usuarios son capaces de realizar una búsqueda por rango de fechas y ubicación de su ciudad mediante GPS, cuando la búsqueda se ejecuta se despliega una lista de vehículos disponibles, teniendo la posibilidad de generar una reserva. La aplicación también brinda la opción de administrar reservas realizadas. Para el desarrollo del proyecto se utilizó un marco de trabajo ágil que permitió el desarrollo iterativo e incremental de la aplicación.

ABSTRACT

In this project, the implementation and development of an iOS mobile application programmed in Swift was carried out, using the Google Cloud Platform cloud infrastructure, Firestore as a database engine and Firebase for user management and authentication. The platform's own web manager was used for its respective administration. Regarding the objective of the mobile application, the rental and rental of cars between private individuals was sought. To meet the stated goal, the registration of users in the mobile application was allowed, which may be vehicle owners, who can offer their vehicles in the case of wanting to generate economic income, or, people looking to rent a vehicle. Users can perform a search by date range and location of their city by GPS, when the search is executed a list of available vehicles is displayed, having the possibility to generate a reservation. The application also provides the option to manage reservations made. For the development of the project an agile framework was used that allowed the iterative and incremental development of the application.

ÍNDICE

1. Capítulo I: Introducción 1
1.1 Alcance 3
1.2 Justificación 6
1.3 Objetivo General 6
1.4 Objetivos específicos 7
2. Capítulo II: Marco Teórico 7
2.1 Introducción 7
2.2 Dispositivos móviles 8
2.2.1 Tipos de dispositivos móviles 9
2.2.1.1 Tablet 9
2.2.1.2 Smartphone 10
2.2.2 Sistemas operativos móviles 11
2.2.2.1 Android 12
2.2.2.2 iOS 12
2.2.3 Aplicaciones móviles 12
2.2.3.1 Aplicaciones nativas 14
2.2.3.2 Aplicaciones web 15
2.2.3.3 Aplicaciones híbridas 16
2.3 Metodología de desarrollo 16
2.3.1 Scrum 18
2.3.1.1 Equipo Scrum (Scrum Team) 20
2.3.1.2 Eventos de Scrum 22
2.3.1.3 Artefactos de Scrum 23

2.4 Lenguaje de programación Swift 23
2.5 Herramientas de desarrollo 25
2.5.1 Cloud Firestore 25
2.5.2 Xcode 26
3. Capítulo III: Desarrollo e Implementación 28
3.1 Introducción.....28
3.2 Diagrama de solución del sistema 29
3.3 Aplicaciones existentes de renta de vehículos 31
3.3.1 Product Backlog 32
3.3.2 Historias de Usuario 38
3.3.2.1 Registro de usuarios en la aplicación..... 38
3.3.2.2 Inicio de sesión de usuarios en la aplicación 39
3.3.2.3 Cierre de sesión de usuarios en la aplicación 40
3.3.2.4 Búsqueda de vehículos..... 41
3.3.2.5 Lista de vehículos disponibles 42
3.3.2.6 Detalles del vehículo 43
3.3.2.7 Confirmación de datos del usuario 43
3.3.2.8 Detalle de la transacción final..... 44
3.3.2.9 Mensaje de confirmación de la reserva 45
3.3.2.10 Envío de mail al reservar 46
3.3.2.11 Administrar reservas del cliente 47
3.3.2.12 Cancelar una reserva..... 47
3.3.2.13 Modificar datos personales..... 48
3.3.2.14 Ingresar vehículo para rentar 49
3.3.2.15 Visualizar vehículos ingresados 50

3.3.2.16 Solicitar registro de pago de reserva 51
3.3.2.17 Reporte de dinero recaudado 52
3.3.2.18 Menú lateral 52
3.3.2.19 Validación de fechas de búsqueda 53
3.3.2.20 Borrar reservas no pagadas 54
3.3.3 Sprint Cero 55
3.3.3.1 Paleta de colores 55
3.3.3.2 Tipografía 56
3.3.3.3 Iconos 57
3.3.3.4 Icono principal de la app 58
3.3.3.5 Icono en dispositivo 58
3.3.4 Resultados del Sprint 1 59
3.3.6 Resultados del Sprint 2 63
3.3.7 Resultados del Sprint 3 66
3.3.8 Resultados del Sprint 4 70
4. Capítulo IV: Resultados 72
4.1 Introducción 72
4.2 Entorno aplicación móvil 73
4.2.1 Entorno aplicación móvil cliente 73
4.2.2 Entorno aplicación móvil propietario 74
4.3 Arquitectura interna de la aplicación (MVC) 74
4.4 Colecciones en Firestore 78
4.5 Caso de Prueba 80
5. Conclusiones y Recomendaciones 84
5.1 Conclusiones 84

5.2 Recomendaciones 85

Referencias 86

ÍNDICE DE FIGURAS

Figura 1. Evolución de dispositivos móviles. 10
Figura 2. Evolución de los sistemas operativos móviles. 11
Figura 3. Scrum. Adaptado de Scrum 19
Figura 4. Diagrama de solución y explicación. 29
Figura 5. Referencias de otras aplicaciones móviles 31
Figura 6. Referencias de otras aplicaciones móviles 32
Figura 7. Product Backlog. 33
Figura 8. Paleta de colores. 56
Figura 9. Tipografía Nunito Sans 57
Figura 10. Iconos 57
Figura 11. Icono de la aplicación. 58
Figura 12. Icono de la aplicación en el dispositivo 58
Figura 13. Pantalla de búsqueda en la aplicación 59
Figura 14. Colección de vehículos en Firestore. 59
Figura 15. Lista de vehículos disponibles 60
Figura 16. Registro de usuarios en la aplicación 60
Figura 17. Usuarios registrados en la Firebase 61
Figura 18. Colección de usuarios en Firestore 61
Figura 19. Login de usuarios en la aplicación. 62
Figura 20. Botón para cerrar sesión. 62
Figura 21. Reservas 63
Figura 22. Cancelación de una reserva 63
Figura 23. Ingreso de vehículos para rentar 64
Figura 24. Visualizar vehículos del propietario 65
Figura 25. Resumen de la reserva 66
Figura 26. Registro de usuarios en la aplicación 66
Figura 27. Función sendEmail 67
Figura 28. Confirmación de la reserva 67
Figura 29. Registro de usuarios en la aplicación 68
Figura 30. Menú lateral 68
Figura 31. Confirmación de datos del usuario 69

Figura 32. Actualización del perfil 69
Figura 33. Detalle del vehículo 70
Figura 34. Selección de fecha 70
Figura 35. Solicitar registro de pago 71
Figura 36. Reporte de dinero 71
Figura 37. Función bookingscleanup 72
Figura 38. Pantallas cliente 73
Figura 39. Pantallas propietario 74
Figura 40. Arquitectura interna de la aplicación 75
Figura 41. Paquete Model 75
Figura 42. Paquete Views 75
Figura 43. Storyboard 76
Figura 44. Paquete Controllers 77
Figura 45. Colección Bookings 78
Figura 46. Colección Brands 78
Figura 47. Colección Constants 79
Figura 48. Colección Payment Requests 79
Figura 49. Colección Users 80
Figura 50. Colección Vehicles 80
Figura 51. Caso de Prueba 83

1. Capítulo I: Introducción

Con la llegada de la era tecnológica a la vida cotidiana de las personas, es difícil no encontrar un negocio o una necesidad en la red. Más allá de páginas web informativas, ahora existen modelos de negocio que funcionan globalmente en internet y están al alcance de todos quienes tengan un dispositivo electrónico a la mano, como, por ejemplo, una computadora, un *smartphone*, un *smartwatch* y cualquier otro dispositivo con conectividad a internet. Actualmente la sociedad es capaz de realizar transacciones de todo tipo en apenas segundos, la mayoría usa sus teléfonos inteligentes para acceder a sitios web o aplicaciones móviles que cumplan con un objetivo en concreto.

En el siglo XXI, se da el boom de la tecnología, aparecen conceptos como el comercio electrónico, también llamado *eCommerce*. Las empresas se adaptaron a este nuevo canal para realizar ventas en Internet. Al principio, el comercio electrónico abrió las puertas a las ventas de bienes o servicios gracias a la *World Wide Web* (WWW) combinando servicios de pago en línea con las tarjetas de crédito al momento de efectuar las transacciones. (Turban, 2009).

El tiempo continuó su rumbo, las empresas adaptaban su modelo de negocio para estar presentes en la red y muchas de ellas empezaron a destacar, cada página web estaba disponible para funcionar en ordenadores de escritorio, sin embargo, el 9 de enero del 2007, Steve Jobs, con buzo negro, jeans azules y zapatillas blancas presentaba al mundo en el *Macworld* el nuevo dispositivo electrónico que combinaba tres cosas: las características musicales de un iPod, un teléfono celular y un comunicador de internet, todo en una sola pieza denominada iPhone. (Merchant, 2017).
Con la llegada del iPhone surgió la App Store, una tienda digital en donde los usuarios de este dispositivo tienen la posibilidad de descargar aplicaciones

móviles que cumplen con un objetivo en concreto. Las empresas empezaron a adaptar sus sitios web a diferentes tamaños de pantalla, en este caso, a la pantalla de un iPhone. Con el tiempo la competencia entre otras marcas hizo que surgieran nuevos dispositivos que compitieran con el nuevo invento de Apple y empezaron a crear *smartphones* de diferentes marcas, mucho más económicos y al alcance de muchos segmentos en la sociedad. El momento de aparición de los terminales inteligentes o *smartphones* se conoce como la tecnología de tercera generación. (Maicas, 2011).

Una vez que se ha ubicado en el contexto ideal de la tecnología que se vive hoy, si se desea referir al comercio electrónico, se lo puede hacer como *mCommerce*, debido a que el uso de aplicaciones móviles ha ido creciendo a un ritmo acelerado y que los *smartphones* han ido desarrollando sus tecnologías tanto en conectividad como en confianza por parte de sus usuarios al momento de transaccionar mediante estos dispositivos. Las plataformas de pago han ido creciendo, se han probado en países de primer mundo y actualmente están disponibles en varios dispositivos móviles como *Android Pay*, *Samsung Pay* o *Apple Pay* (Ashraf, 2017).

Como se ha visto y vivido en los últimos años, ya muchas personas han dejado de lado los ordenadores tradicionales y una gran parte de la población utiliza sus *smartphones* para acceder a sitios web o aplicaciones móviles que sean de su interés, los términos aplicaciones web y móviles son cada vez más escuchados. Cada día más usuarios utilizan sus dispositivos para acceder a redes sociales, encontrar información en internet, revisar sus cuentas bancarias, conseguir empleos e incluso generar fuentes de trabajo. Las personas prefieren utilizar aplicaciones móviles en lugar de ingresar a un sitio web debido a su facilidad de uso y adaptación de contenido, como, por ejemplo, la aplicación de Mercado Libre. (Turban, 2009).

Las necesidades que presenta la sociedad en el mundo actual se han ido incrementando a medida que pasa el tiempo y las empresas de tecnología han ido desarrollando aplicaciones de software que faciliten la vida de los usuarios, hay aplicaciones con mucho éxito en Ecuador, como, por ejemplo, Glovo para entregar comida a domicilio, Uber que busca choferes en tiempo real de acuerdo con la posición GPS, entre otras.

Los beneficiarios siempre serán los usuarios, los cuales pueden ofrecer servicios a otras personas, como es el caso de Airbnb que ofrece departamentos para rentar a turistas. La misma lógica de negocio puede ser aplicada en una aplicación que permite la renta de autos entre personas particulares ofreciendo ganancia para los usuarios que utilicen la aplicación y un porcentaje para los dueños de la App.

1.1 Alcance

Implementación de una aplicación móvil iOS programada en Swift con la utilización de Xcode y la API de Google Cloud Firebase que permita la renta de autos por parte de personas particulares. Para esto se debe permitir el registro de usuarios en la aplicación móvil, se realizará una búsqueda por rango de fechas y ubicación de la ciudad mediante GPS, se desplegará una lista de vehículos disponibles y se procederá a generar una reserva. También los propietarios de vehículos pueden acceder a la aplicación, pero ellos además visualizarán las reservas sobre sus autos y las ganancias totales generadas por las rentas, para convertirse en propietario se permite el ingreso de vehículos en la misma aplicación. Por otro lado, el propietario de la aplicación (desarrollador) utilizará el gestor web de Google Firebase para administrar las cuentas de usuario y la base de datos.

El servidor de la aplicación estará alojado en la nube en Google Cloud Firebase que permite el escalamiento de la aplicación en dependencia de los usuarios que tenga la aplicación, es decir, a mayores usuarios tiene un mayor costo, pero en una fase inicial con pocos usuarios la aplicación puede mantenerse de forma gratuita. Se utilizará la versión gratuita de Firebase.

La interfaz que manejará la aplicación móvil será amigable con el usuario, se tomará como referencia varias aplicaciones que hay en el mercado y se pensará siempre en la comodidad y experiencia de usuario.

La aplicación obtendrá las coordenadas del usuario mediante el GPS del dispositivo, luego determinará la ciudad de las coordenadas obtenidas y buscará en la base de datos los vehículos que coincidan con dicha ciudad y un rango de fechas previamente ingresado, inicialmente para este proyecto la aplicación tendrá un enfoque sólo en la ciudad de Quito.

La aplicación será únicamente para iOS, no para Android. La aplicación estará disponible localmente para la presentación, no se subirá al AppStore por el costo de la licencia. La aplicación no realiza ningún trámite legal en cuanto a la renta vehicular, solo se encarga de gestionar la búsqueda de vehículos y contactarlas con el propietario. La aplicación será compatible con iPhone y iPad con la versión iOS 13.

Los propietarios y sus vehículos serán ingresados directamente en la aplicación móvil, cualquier usuario registrado puede poner en renta sus vehículos, el instante en que un usuario registra su vehículo, este se convierte en propietario. Los propietarios podrán ingresar en la aplicación móvil con su email y contraseña, la interfaz de usuario para ellos será diferente ya que el menú tendrá

dos nuevas opciones, una para visualizar sus vehículos y otra para ver el total de las rentas.

Para desarrollar aplicaciones iOS se necesita una cuenta de desarrollador de Apple, la cual tiene un valor anual de 100 dólares, sin embargo, Apple permite desarrollar aplicaciones de forma gratuita con algunas limitaciones, entre las más importantes que afectan a este proyecto, no se permite subir la aplicación al App Store mientras no se tenga la licencia, no se permite realizar cobros online desde la aplicación y no se permite enviar notificaciones externas directamente a otro dispositivo. Con esto en cuenta, cuando se haga una reserva sobre un vehículo solamente se enviará un mail y no una notificación directa al dispositivo del propietario.

La aplicación no hace procesos de pago de ningún tipo, como se menciona anteriormente, se encarga de contactar al propietario del vehículo con el interesado. El pago sería registrado por el administrador en un tiempo máximo de 3 días antes de la entrega del vehículo. Esto se debe a que implementar pagos en la aplicación está limitado por Apple mientras no se tenga una licencia oficial de desarrollador.
Al ser una aplicación que trabaja directamente con los servidores de Google, es importante contar con una conexión a internet, de lo contrario la aplicación pierde su funcionalidad.

Para el desarrollo se utilizará un tablero Kanban en donde se pondrán las actividades que están a la cola, las que se están desarrollando, las que se están probando y las que han sido concluidas. También se usará Scrum que es un framework de trabajo que ha sido visualizado como uno de los mejores a la hora de realizar proyectos complejos ya que permite la entrega continua mediante incrementos funcionales en ciclos cortos de máximo un mes cada uno.

1.2 Justificación

En la actualidad, muchos modelos de negocio han sido trasladados al mundo de las aplicaciones móviles dando resultados positivos y ganancias no solo para los creadores de la aplicación, sino también para los usuarios que se han registrado y ofrecido sus servicios como parte funcional de la aplicación, como, por ejemplo, el caso de Airbnb, en la cual los usuarios registrados ponen a disposición casas o habitaciones para ser rentadas por otras personas registradas también en la aplicación.

Basado en ese modelo de negocio se ha pensado que también se pueden ofrecer otros servicios, como lo es la renta de vehículos, en la cual los usuarios pondrían a disposición sus vehículos y otras personas podrían alquilarlos.

Para los usuarios que necesitan un vehículo tendrán la oportunidad de buscar un vehículo en un rango de fechas, hacer una reservación, alquilarlo y devolverlo. Para los usuarios que deseen ganar dinero mediante el alquiler de sus vehículos tendrán la oportunidad de registrar su automotor, obtener solicitudes de alquiler de acuerdo con su calendario de disponibilidad, encontrar al conductor y firmar un contrato, recuperar su auto después de alquilarlo y revisar que todo esté en orden. De esa forma se logrará un modelo de negocio dentro de una aplicación móvil ofreciendo beneficios a quienes buscan un vehículo y a quienes buscan alquilar el suyo.

1.3 Objetivo General

Implementar una aplicación móvil en iOS con Swift y la API de Google que involucre el GPS para el alquiler de vehículos online por parte de personas particulares que busquen alquilar o poner en alquiler, con el fin de generar un modelo de negocio rentable.

1.4 Objetivos específicos

- Definir una arquitectura de software escalable y reconocida que se adapte a las necesidades de la aplicación.

- Diseñar e implementar un modelo de datos en la nube de Google Cloud Firebase que se adapte a las necesidades de crecimiento de la aplicación para que soporte todas las transacciones generadas por la aplicación móvil y el crecimiento de usuarios.

- Aplicar la arquitectura de software Modelo – Vista – Controlador, que aporte al rendimiento y mantenimiento de la aplicación móvil mediante buenas prácticas de programación como *clean code*, nombres en inglés, patrones de diseño de software y características de la programación orientada a objetos.

- Diseñar interfaces de usuario sencillas y amigables con el usuario final para ofrecer una buena experiencia en la aplicación.

- Comprobar que las funcionalidades de la aplicación móvil se cumplan.

2. Capítulo II: Marco Teórico

2.1 Introducción

En este capítulo se hablará sobre los dispositivos móviles, los tipos de dispositivos que existen y han existido, sus sistemas operativos, como han evolucionado y como las aplicaciones móviles han ido tomando popularidad hasta la actualidad.

También se abarcará sobre la metodología de desarrollo aplicada en este proyecto, para ello se tomará como referencia la guía de Scrum para entender su propósito, visión, uso, teoría, sus valores, como se trabaja en un equipo, los eventos que ocurren y los artefactos.

Para conocer más a fondo el lenguaje de programación para desarrollar este proyecto se hablará sobre Swift, el lenguaje creado por Apple utilizado mundialmente, para ello se tomará como referencia la documentación oficial de la página web de *Apple Developer Documentation*.

Otro punto que se verá en este capítulo son las herramientas involucradas en el desarrollo del proyecto, las principales son Google Firestore que es una base de datos no relacional (NoSQL) en la nube y Xcode que es el entorno de desarrollo de Apple y actualmente está en la versión 11 que trabaja con Swift 5.

Finalmente se presentará un diagrama que cumpla con los objetivos de la solución y una explicación de por qué se ha decidido crear la arquitectura de esa manera.

2.2 Dispositivos móviles

Un dispositivo móvil es un aparato electrónico del tamaño aproximado de una mano, cuenta con capacidad de procesamiento, pueden contar con una conexión a red, poseen una memoria y han sido diseñados para cumplir con una función específica. (Alonso, 2011). En base a este concepto existen varios dispositivos móviles, como reproductores de audio, navegadores GPS, teléfonos móviles, *tablets*, entre otros. Este trabajo estará centrado en los teléfonos móviles y

tablets ya que son los más usados en la actualidad y ofrecen un sinnúmero de posibilidades para la implementación y utilización de aplicaciones móviles.

Los desarrollos recientes en tecnología móvil, incluida la incorporación del iPad, así como otros *tablets* y *smartphones*, han proporcionado nuevas herramientas significativas en la comunicación. La extensa disponibilidad de estas tecnologías portátiles, potentes y en red ha cambiado la forma en que trabajamos, aprendemos, pasamos nuestro tiempo libre e interactuamos socialmente. (McNaughton, 2013).

2.2.1 Tipos de dispositivos móviles

2.2.1.1 Tablet

Las *tablets* son la evolución de los PDA (Personal Digital Assistant). Un PDA fue construido originalmente como una agenda electrónica que cuente con un reconocimiento de escritura. Alrededor del 2009 un PDA podía realizar las mismas funciones que una computadora de escritorio, con la utilidad de ser portátil.

Posteriormente en el 2010, el 27 de enero fue presentada la primera generación del iPad diseñado por Apple. Se situó en la categoría de una computadora portátil y un teléfono inteligente. Aunque muchas personas creen que el iPad fue la primera Tablet creada, en realidad, este concepto de computadora portátil fue creado originalmente por Alan Kay allá en 1968 con el nombre de Dynabook. Se publicó un artículo en 1972 titulado "*A Personal Computer for Children of All Ages*" (Kay, 2000).

Las características de una *tablet* es que tienen pantalla táctil, conexión a una red, procesador, memoria, entre otras. Se diferencia de un teléfono inteligente por su tamaño que es superior.

Figura 1. Evolución de dispositivos móviles.

2.2.1.2 Smartphone

Un *Smartphone* o teléfono inteligente es un artefacto móvil inalámbrico que técnicamente se basa en el uso de ondas de radio, se caracteriza principalmente por ser portátil, no necesita estar en un lugar fijo y para conectarse a la red de telefonía no requiere ningún tipo de cableado.

Antes de la llegada del Smartphone existían los teléfonos móviles normales cuya principal función era la comunicación por voz, mensajería instantánea mediante *sms*, administración de contactos, juegos, cámara, música, entre otros. Con el paso de esta última década, los teléfonos fueron adquiriendo el concepto de teléfonos inteligentes, ya que fueron adaptando otras particularidades semejantes a las de un ordenador, con ello se puede decir que es una mezcla entre una Tablet y un teléfono móvil clásico.

Una de las características más importantes de estos dispositivos es que posibilita la instalación de aplicaciones adicionales, lo que conlleva al incremento de las funcionalidades presentes o mejorarlas. Generalmente estas aplicaciones suelen ser escritas y desplegadas por el fabricante o también por terceros. Otras de sus características están el uso de pantallas táctiles, sistemas operativos modernos, conexión a internet, cámaras digitales, entretenimiento, servicios multimedia, entre otras.

2.2.2 Sistemas operativos móviles

Un sistema operativo es el principal programa en ser ejecutado en un ordenador, es un conjunto de extensiones o rutinas de software que permiten la gestión y administración del hardware, hacen funcionar a un computador y proporcionan un entorno que permite ejecutar programas. (Naty, 2004).

El uso de un sistema operativo u otro determina las capacidades que puede ofrecer el dispositivo para interactuar con el usuario. Existe una gran diversidad de sistemas operativos orientados a los dispositivos móviles, hace unos años atrás los más populares eran Symbian, BlackBerry OS y Windows Mobile.

Figura 2. Evolución de los sistemas operativos móviles.

Adaptado de (Xcube Labs, 2011).

Con la llegada de los *smartphones* los sistemas operativos que destacaron y lo siguen haciendo hasta el día de hoy son iOS de Apple y Android de Google, de los cuales se hablará a continuación:

2.2.2.1 Android

Android es un sistema operativo de Google que está basado en Linux y fue creado inicialmente para funcionar solamente en dispositivos móviles, además, otros dispositivos como televisores o reproductores de música lo usan. El entorno de trabajo que ofrece Android para programar es Java y Kotlin, dos lenguajes de programación que pueden utilizarse para crear aplicaciones móviles de forma nativa.

2.2.2.2 iOS

iOS es el sistema operativo móvil desarrollado por Apple, fue creado originalmente para su uso en el iPhone, pero ahora también está presente en otros dispositivos de la misma compañía. iOS se deriva de Mac OS y ha sido simplificado para sacar un mejor rendimiento en sus dispositivos. Las tecnologías que usa este sistema operativo son el *Kernel* OS X, BSD sockets para la comunicación, Objective-C, C y C++. El entorno de trabajo que ofrece iOS para programar es Swift y Objective-C, dos lenguajes de programación que pueden utilizarse para crear aplicaciones móviles de forma nativa.

2.2.3 Aplicaciones móviles

Las aplicaciones móviles son programas de software que son instalados en los dispositivos móviles y son creados por los fabricantes o por desarrolladores externos. La función de una *app* es permitir al usuario efectuar varias tareas en

conjunto, estas pueden ser de entretenimiento, de comunicación, para uso personal, educativas, para acceder a un servicio, entre otras.

Generalmente las aplicaciones móviles se encuentran disponibles para los usuarios en las plataformas oficiales de distribución de las compañías propietarias de cada uno de los sistemas operativos, por ejemplo, las aplicaciones de iOS se encuentran disponibles en el App Store del dispositivo, las aplicaciones de Android se encuentran en el Play Store de Google.
En cuanto a costos, éstas pueden ser gratuitas o pagadas mediante un solo pago o suscripciones. Generalmente la mayoría de las que son gratis tienen publicidad, son versiones limitadas u ofrecen un servicio de pago, mientras que las de pago son completas y libres de publicidad. En promedio, alrededor del 20 al 30% del costo que tiene la aplicación en la tienda es destinada al distribuidor y el valor restante se entrega al desarrollador.

Algunas de las características de mayor importancia en las aplicaciones móviles son:

- Residen directamente en los dispositivos móviles de forma compilada, ofreciendo al usuario una serie de funcionalidades que satisfacen una necesidad.
- Proveen de un rápido y sencillo acceso a la información mediante características propias del dispositivo, por ejemplo, una aplicación bancaria que utiliza el reconocimiento facial para ser utilizada sin necesidad de acceder por la web.
- Satisfacción de una necesidad especifica del usuario.

En los últimos años los dispositivos móviles han sufrido una metamorfosis, convirtiéndose en una herramienta cotidiana para la gente alrededor del mundo,

de esa forma, el mercado de estos dispositivos ha crecido notablemente, especialmente en las plataformas de iOS y Android.

Para que una aplicación tenga un mayor alcance es preferible que se encuentre disponible en todas las plataformas.

Existen tres enfoques para desarrollar aplicaciones móviles: de forma nativa, multiplataforma web y multiplataforma hibrida.

2.2.3.1 Aplicaciones nativas

Las aplicaciones nativas son aquellas que se desarrollan para ejecutarse sobre una plataforma específica, es decir, una aplicación escrita para Android no utilizará el mismo código fuente para iOS, aunque cumplan la misma funcionalidad se deberá realizar el desarrollo de forma independiente para cada una de las plataformas.

La ventaja primordial de desarrollar aplicaciones nativas es que se puede aprovechar de mejor manera las capacidades del dispositivo que son propias de éste, como, por ejemplo, la cámara, el acelerómetro, el giroscopio, el GPS, entre otras cosas.

Otra ventaja sumamente importante es el rendimiento, al ser una aplicación nativa para esa plataforma no se necesita de intermediarios como capas web, ya que trabaja directamente sobre el dispositivo su ejecución es más rápida y consume menos recursos.

Una desventaja de las aplicaciones nativas es su costo de desarrollo, ya que es más elevado debido al trabajo para cada plataforma y el tiempo empleado, sin dejar de lado que se requiere la utilización de un lenguaje de programación distinto para cada plataforma, lo que implica mayores costos en actualización, distribución y mantenimiento.

Las aplicaciones nativas antes de ser publicadas en las tiendas oficiales de cada plataforma deben cumplir con una auditoria previa por parte del distribuidor y el desarrollador necesita una licencia para publicarla.

2.2.3.2 Aplicaciones web

Las aplicaciones web son desarrolladas para ejecutarse sobre un navegador en el dispositivo móvil. Se desarrollan utilizando HTML, CSS y Javascript, no necesitan de una auditoria por parte del distribuidor ya que no se instalan sobre el dispositivo, el único requerimiento es tener acceso a internet.

Generalmente los desarrolladores web crean una versión de una página web que se adapte a dispositivos móviles, conocido como maquetación *responsive*, los sitios se adaptan a cualquier tamaño de pantalla para que se vea bien sin importar la marca o características del dispositivo.

Entre las ventajas que se obtiene por el desarrollo de aplicaciones web es que cualquier mantenimiento o actualización se la hace directamente sobre el servidor donde está alojada la aplicación y el cambio es instantáneo y transparente para el usuario. Otra ventaja es que sin importar la plataforma en la que se ejecuta, no requiere adecuarse a ningún entorno operativo, solo se

necesita del navegador. Como desventaja se tiene que el rendimiento se ve disminuido comparado con una aplicación nativa, no puede acceder directamente a todos los recursos del dispositivo y no puede verse tan bien como una nativa.

2.2.3.3 Aplicaciones híbridas

Las aplicaciones híbridas utilizan parte de las aplicaciones web con las nativas, se utiliza HTML, CSS y Javascript y también es posible acceder a una buena parte de las capacidades presentes en los dispositivos como cámara, GPS, entre otras.
Al ser desarrolladas con tecnología web, se necesita un contenedor web dentro de la aplicación sobre el dispositivo, existe un intermediario que transforma los elementos web a elementos nativos.

Como ventajas se puede mencionar que existe la posibilidad de subir estas aplicaciones a la tienda oficial de la plataforma, la reutilización del mismo código para varias plataformas y el acceso a las características propias del dispositivo.

Como desventajas se tiene que, al ser una misma interfaz diseñada para las diferentes plataformas, su apariencia puede verse más pobre en comparación con una aplicación nativa, otra desventaja es que el rendimiento se ve afectado al tener una ejecución más lenta.

2.3 Metodología de desarrollo

Debido a la complejidad que puede presentar el desarrollo de software se han creado varias metodologías de desarrollo que inciden de diferente manera sobre

este proceso. Se tiene propuestas tradicionales centradas particularmente en el control de proceso, en establecer de forma rigurosa y ordenada las actividades involucradas, la producción de artefactos y las distintas herramientas que pueden utilizarse. Estas propuestas han tenido éxito en muchos casos, sin embargo, también han tenido fracasos en muchos otros.

Entre las metodologías de desarrollo tradicionales están el método de cascada, el de prototipos, espiral, incremental y el Desarrollo Rápido de Aplicaciones (RAD).

Por otro lado, están las propuestas con metodologías ágiles que se centran en otros ámbitos tales como el factor humano o el producto final de software. La filosofía presente en estas metodologías da mayor valor al individuo, generando mayor interacción y colaboración directa con el propietario y creando un desarrollo incremental de software que se basa en cortas iteraciones. Se recomienda utilizar esta metodología cuando los requisitos en un proyecto son muy cambiantes, cuando se tiene tiempos de desarrollo reducidos, pero se busca una alta calidad y cuando se tiene un contacto directo con el dueño del producto, es decir, el cliente.

Actualmente en el mercado de desarrollo de software, las metodologías ágiles se están imponiendo sobre las tradicionales, esto se debe al éxito que han tenido, lo que a su vez también ha generado algo de escepticismo para quienes aún no ven estas metodologías como una alternativa hacia las tradicionales.

Entre las metodologías de desarrollo ágiles están la programación extrema (XP), Scrum, Crystal, Kanban, *Lean Development* (LD), *Adaptative Software*

Development (ASD), *Feature Driven Development* (FDD), *Lean Software Development* (LSD) y el Proceso Unificado de Desarrollo de Software.

Las diferencias primordiales entre metodologías ágiles y tradicionales es que en las ágiles se está preparado para los cambios durante el proyecto, en las tradicionales existe cierta resistencia; en las ágiles el proceso es menos controlado a diferencia de las tradicionales que existe mucho control; en las ágiles no existe un contrato tradicional como en las tradicionales, de hecho, son contratos bastante flexibles; en las ágiles el cliente tiene un rol y es parte del equipo, en las tradicionales el cliente solo interviene en pocas ocasiones; en las ágiles se tiene equipos de trabajo con pocas personas, en las tradicionales los equipos pueden ser muy grandes y quizá estén distribuidos de acuerdo a una especialización; en las ágiles no es tan importante la documentación como en las tradicionales; en las ágiles se tiene constantes entregas hacia el cliente, en las tradicionales son pocas.

2.3.1 Scrum

Scrum es un marco de trabajo de desarrollo ágil creada por Ken Schwaber y Jeff Sutherland. Define un framework para gestionar proyectos y ha sido utilizado con éxito desde principios de los 90. Se recomienda su uso o adaptación en proyectos que puedan presentar un cambio rápido de requisitos.

Entre sus características está el desarrollo de software con iteraciones, conocidas como *sprints*, con una duración no mayor a 4 semanas, el resultado que se obtiene en cada sprint es un incremento que aporta valor para el cliente. Otra característica importante es el contacto directo con el cliente, con reuniones continuas a lo largo del proyecto.

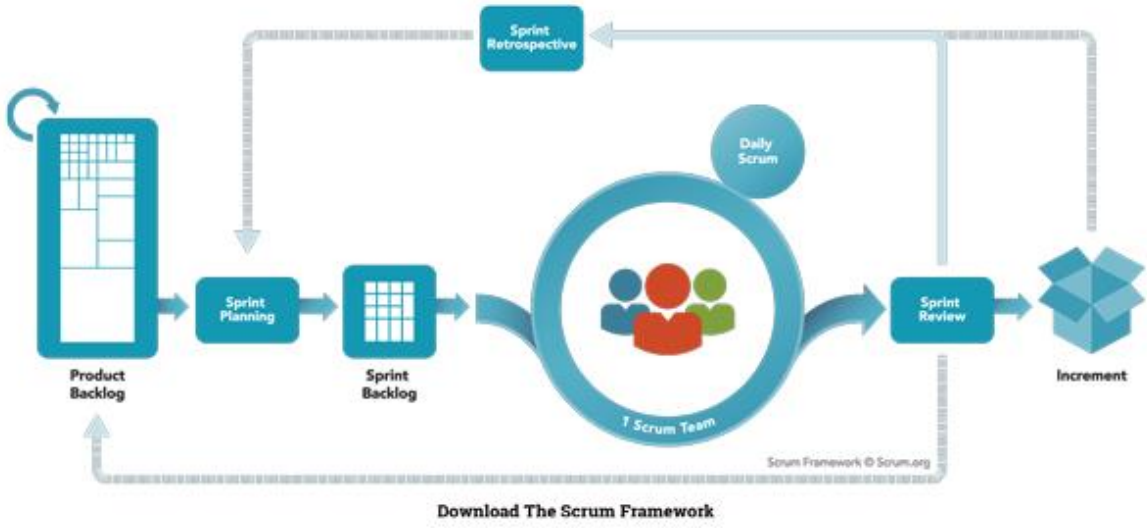


Figura 3. Scrum.

Adaptado de (Scrum, 2019).

Es importante mencionar que Scrum no es una metodología como tal, es un marco de trabajo con técnicas y procesos que pueden ser aplicados en diferentes proyectos independientemente si son de software o no. El framework Scrum consiste en la conformación de equipos Scrum, definición de roles, artefactos, eventos y reglas asociadas. El cumplimiento de cada uno de estos componentes es fundamental para la ejecución y éxito de Scrum.

Scrum se fundamenta en la teoría de control de proceso de forma empírica, de esa forma se ratifica que el conocimiento proviene de la experiencia obtenida en la práctica y la toma de decisiones sustentadas por el conocimiento. Scrum maneja un planteamiento iterativo e incremental que permite mejorar el control de riesgo y predictibilidad. Los tres pilares que sustentan la implementación de control son la transparencia, la inspección y la adaptación.

La transparencia se refiere a que los responsables del resultado aprecien y vean los aspectos significativos del proceso. La inspección se refiere a que

frecuentemente se deben inspeccionar los artefactos y poner atención en el progreso ante un objetivo para evitar alteraciones no deseadas, las inspecciones deben aportar valor y no interferir en el trabajo. La adaptación se refiere a que deben realizarse ajustes en caso de ver una desviación que afecte el resultado final.

2.3.1.1 Equipo Scrum (Scrum Team)

Como la guía de Scrum lo dice, el Equipo Scrum (*Scrum Team*) está formado por el Dueño del Producto (*Product Owner*), el Equipo de Desarrollo (*Development Team*) y el *Scrum Master*. (Sutherland y Schwaber, 2017).
Cada Equipo Scrum se caracteriza por ser auto organizado, es decir, ellos mismo son quienes se dirigen, son multifuncionales, poseen todas las competencias imprescindibles para ejercer su trabajo sin depender de personas externas. El *Scrum Team* realiza entregas de valor de forma iterativa e incremental con el fin de aumentar las oportunidades de conseguir retroalimentación, de esa forma se asegura una versión que aporte potencialmente funcionalidad al producto.

Dueño del Producto (*Product Owner*)

Para ejecutar el marco de trabajo Scrum es necesario contar con la participación directa del cliente dentro del equipo, se lo conoce como dueño del producto y es responsable de aumentar el valor del producto resultante.

Es importante recalcar que el dueño del producto no puede ser un comité, es una sola persona. Para que el dueño del producto pueda realizar bien su trabajo, todos deben respetar sus decisiones.

El dueño del producto tiene la responsabilidad de administrar la Lista del Producto.

Esta administración incluye manifestar de forma clara y concisa los elementos que constituyen la Lista del Producto, el ordenamiento de los elementos de acuerdo con la necesidad para alcanzar los objetivos de la forma más efectiva, la optimización del valor de trabajo efectuado por el equipo de desarrollo, garantizar que los elementos de la lista sean claros para todos y finalmente también comprobar que el equipo de desarrollo comprende cada uno de los elementos a un nivel adecuado.

Equipo de Desarrollo (*Development Team*)

El equipo de desarrollo está adaptado y constituido por los profesionales encargados de realizar el trabajo de proporcionar incrementos que potencialmente puedan ser puestos en producción al concluir cada sprint. Ante cada incremento se realiza una revisión del sprint.

Una de las características importantes que tienen los equipos de desarrollo es que son auto organizados, no puede intervenir nadie más que ellos, ni siquiera el Scrum Máster; son multifuncionales, poseen las competencias para llegar a un incremento. Scrum no considera títulos para el grupo del equipo de desarrollo, no existen sub-equipos. La responsabilidad la tiene todo el equipo de desarrollo en conjunto.

El tamaño en número de personas de un equipo de desarrollo comprende entre tres y nueve. Ni el rol de Dueño del Producto ni el *Scrum Master* se toman en

cuenta para la sumatoria, a menos que también contribuyan con la Lista de Pendientes en el equipo trabajando en desarrollo.

Scrum Master

El *Scrum Master* es el individuo responsable de apoyar y promover Scrum como lo indica la Guía de Scrum, esto con el fin de ayudar a entender a todos la teoría, la implementación correcta de las practicas, velar por el cumplimiento de las reglas y los valores de Scrum.

2.3.1.2 Eventos de Scrum

Sprint

El Sprint es el núcleo que le da sentido a Scrum, se toma como un periodo de tiempo de aproximadamente treinta días y en el cual su objetivo es crear un incremento utilizable y de valor del producto que puede ser puesto en producción. Ante la finalización de un Sprint se comienza uno nuevo. No se puede efectuar cambios que puedan perjudicar al Objetivo del Sprint. La única persona que puede cancelar un Sprint es el Dueño del Producto.

Los Sprints están compuestos por:

- Planificación del Sprint (*Sprint Planning*). En esta etapa se proyecta el trabajo a ejecutar en el Sprint.
- Scrums Diarios (*Daily Scrums*). Son reuniones de 15 minutos diarias con el equipo de desarrollo.
- Trabajo de Desarrollo.

- Revisión del Sprint (*Sprint Review*). Al concluir el Sprint se procede con la revisión del incremento, con el fin de obtener retroalimentación.
- Retrospectiva del Sprint (*Sprint Retrospective*). Se inspecciona a si mismo el Equipo Scrum con el fin de buscar mejoras.

2.3.1.3 Artefactos de Scrum

Lista de Producto (*Product Backlog*)

Es una lista estructurada, organizada y ordenada que contiene todos los elementos que se conocen que pueden ser necesarios para crear el producto. Es donde está el origen de requerimientos para realizar modificaciones en el producto. El responsable de esta lista es el Dueño del Producto.

Lista de Pendientes del Sprint (*Sprint Backlog*)

Son todos los elementos que conforman la Lista de Producto que han sido elegidos para el Sprint, con la adición de un plan que indica como se entregará el incremento del producto y como se cumplirá con el Objetivo del Sprint.

Incremento

Se entrega al final del Sprint, es el resultado de los elementos completados durante el Sprint, debe estar terminado, y aportar valor, est listo para ser utilizado.

2.4 Lenguaje de programación Swift

El lenguaje de programación de Apple utilizado para el desarrollo de software de *macOS, iOS, watchOS, tvOS* y más es Swift. Es un lenguaje de programación relativamente nuevo, fue lanzado oficialmente en el *Apple's Worldwide Developers Conference* en el 2014.

Anteriormente se utilizaba Objective-C para el desarrollo tecnológico en las plataformas Apple, sin embargo, Objective-C es un lenguaje antiguo, si bien esto no siempre es un problema, en este caso conduce a algunas dificultades. La sintaxis de Objective-C se solidificó antes del surgimiento de prominentes lenguajes de scripting en la década de 1990 que popularizaron una sintaxis más estilizada y elegante, como JavaScript, Python, Ruby, entre otros. Esto hace que Objective-C se sienta extraño para la mayoría de los desarrolladores cuando comienzan, por lo que su sintaxis puede ser un impedimento para la productividad del desarrollador. Además, como lenguaje antiguo, Objective-C no tiene muchos avances que los desarrolladores en lenguajes modernos disfrutan actualmente. (Mathias y Gallagher, 2016).

Swift es un lenguaje seguro, Objective-C no pretendía ser inseguro, pero las cosas han cambiado bastante desde que Objective-C se lanzó en la década de 1980. Por ejemplo, el compilador Swift tiene como objetivo minimizar el comportamiento indefinido, que está destinado a ahorrar tiempo al desarrollador en el código de depuración que falló durante el tiempo de ejecución de una aplicación. Otro objetivo de Swift es ser un reemplazo adecuado para la familia de lenguajes C (C, C ++ y Objective-C). Eso significa que Swift tiene que ser rápido. De hecho, el rendimiento de Swift es comparable a estos idiomas en la mayoría de los casos. Swift le brinda seguridad y rendimiento en una sintaxis limpia y moderna. El lenguaje es bastante expresivo; Los desarrolladores pueden escribir código que se siente natural. Esta característica hace que Swift sea un placer de escribir y fácil de leer, lo que lo hace ideal para colaborar en proyectos más grandes. (Mathias y Gallagher, 2016).

Por último, Apple quiere que Swift sea un lenguaje de programación de propósito general. Este deseo se reflejó en la decisión de abrir el código Swift en diciembre de 2015. El código abierto no solo invita a la participación del desarrollador para ayudar al progreso del idioma, sino que también facilita que los desarrolladores transfieran el lenguaje a otros sistemas más allá de macOS y iOS. (Mathias y Gallagher, 2016).

2.5 Herramientas de desarrollo

2.5.1 Cloud Firestore

El manejo de infraestructura muchas veces es un problema al momento de desplegar aplicaciones, almacenar datos, realizar analítica o la integración con otros sistemas. Grandes compañías como Amazon o Google ofrecen servicios en la nube que facilitan el manejo de infraestructura, y no solo eso, sino que también consagran seguridad y robustez en todas sus funcionalidades. Cloud Firestore es una base de datos no relacional NoSQL que reside en la nube de Google y se puede acceder directamente en Firebase o en Google Cloud Platform. Entre sus características principales es que es escalable y flexible, almacena y sincroniza los datos para el desarrollo, tanto en el lado del cliente como del servidor.

Gracias a su API, esta plataforma puede utilizarse para desarrollar programación en servidores y dispositivos móviles, ofrece una sincronización en tiempo real entre las aplicaciones cliente con la utilización de agentes de escucha que se encargan de verificar si hay algún cambio en la base de datos y anunciarlo a todos los dispositivos conectados y tiene una buena capacidad de respuesta a pesar de que existiera latencia en la red o fallas de conectividad.

Entre sus funciones clave destacan las siguientes:

- Flexibilidad, su modelo de datos admite estructuras flexibles y jerárquicas, utiliza colecciones y documentos para almacenar los datos.
- Consultas expresivas, las consultas pueden incluir filtros para obtener documentos específicos y con criterio definido. Utiliza indexación de forma predeterminada, lo que hace que las consultas tengan un gran rendimiento.
- Actualizaciones en tiempo real, utiliza *Realtime Database* para sincronizar los datos desde cualquier dispositivo conectado.
- Asistencia sin conexión, almacena datos en caché para que la aplicación pueda leer, escribir, escuchar y consultar datos sin importar que el dispositivo esté sin conexión.
- Escalable, ofrece la poderosa infraestructura de Google Cloud Platform, lo que conlleva a tener una replicación automática de datos en varias regiones, garantiza una coherencia sólida, realiza operaciones atómicas y ofrece asistencia sobre transacciones.
- Las *apps* que pueden acceder a Google Cloud Firebase son desde Android, iOS y Web. También se encuentra disponible para Node.js, Java, Go y Python.

2.5.2 Xcode

Xcode es el Entorno de Desarrollo Integrado (IDE) que ofrece Apple de manera oficial para los desarrolladores. Al momento de escribir este documento, Xcode se encuentra en la versión 11 que es compatible con Swift 5.

Xcode está centrado en la experiencia de desarrollo de Apple, se integra de forma nativa con los *frameworks* Cocoa y Cocoa Touch. Desde este IDE se

pueden crear aplicaciones para todos los dispositivos de la compañía. Entre algunas de sus características se pueden destacar las siguientes:

- Editor de código, admite la escritura de código con la utilización de un editor profesional, comprobación de sintaxis y burbujas de mensajes que muestren advertencias, errores o notificaciones relacionadas al desarrollo.

- Catálogo de activos, el editor del catálogo de activos en Xcode administra las imágenes de la aplicación, agrupando varias resoluciones del mismo activo. Al compilar, Xcode compila el catálogo de activos en el paquete más eficiente para la distribución final.

- Asistente de edición, el botón Asistente se encarga de dividir el editor en dos, creando una sección secundaria que muestra automáticamente los archivos de mayor utilidad para el desarrollador en función del código que está editando activamente.

- Asistente de versiones, permite la integración hacia controladores de versión como Git, permite visualizar una línea de tiempo de los cambios realizados.

- Creador de interfaces de usuario, permite diseñar y probar interfaces de usuario sin escribir una línea de código, se prototipa en minutos, luego permite conectar gráficamente la interfaz a la fuente dentro del editor Xcode.

- Documentación, permite buscar y encontrar fácilmente cualquier cosa en la Ayuda o en el visor de documentación y referencia de API.

- Simulador, Con el SDK de iOS, Xcode puede realizar la construcción, instalación, ejecución y depuración de aplicaciones Cocoa Touch en un simulador basado en Mac para un flujo de trabajo de desarrollo optimizado.

3. Capítulo III: Desarrollo e Implementación

3.1 Introducción

En este capítulo se hablará sobre el desarrollo e implementación de la aplicación móvil escrita en Swift y enlazada con Google Cloud Firebase. Para ello se analizará un diagrama de solución en donde se presentará la arquitectura entre la aplicación y la plataforma en la nube de Google Firebase.

Posteriormente se analizará la situación actual de este tipo de apps, revisando y analizando referentes con un modelo de negocio similar. Posteriormente se escribirán las historias de usuario que servirán para empezar con la implementación de los Sprints y el desarrollo iterativo e incremental, al final de cada ciclo se obtendrá una nueva funcionalidad que cumpla con los criterios de aceptación de cada historia de usuario.

En este capítulo también se especificarán temas de aspectos visuales y restricciones en el diseño gráfico de la aplicación.

3.2 Diagrama de solución del sistema

Como se puede apreciar en la Figura 4, tanto la aplicación móvil como la aplicación web se conectan a los servidores de Google Firebase mediante su API disponible para cada plataforma. Dentro de Firebase se tiene la base de datos Realtime Database Firestore y también se realiza el almacenamiento de la información.

Figura 4. Diagrama de solución y explicación. Aplicación móvil o web se conectan a los servidores de Firebase.

Adaptada de (Google, 2019).

Cuando un usuario ingresa a la aplicación móvil o web, se solicita el usuario y la contraseña que están registrados en los servidores de Google Firebase, que éste a su vez se encarga de validar la información y permitir el acceso a la información de cada usuario almacenado en la base de datos.

Firebase mediante su robusta infraestructura se encarga de procesar las transacciones, sincronizar datos, proveer de seguridad y mantener los datos actualizados en tiempo real para todos los usuarios.

Firebase ofrece una API para los desarrolladores que utilizan su infraestructura, compatible con Android, iOS, JavaScript, entre otras. En la plataforma de Google se tiene la documentación oficial en la cual explica paso a paso como conectar la aplicación con la nube, mientras este paso no haya sido realizado, la app no tiene ningún acceso a las funcionalidades ofrecidas. Para el caso de iOS se necesita del *framework* CocoaPods, similar a Maven en Java, con unas pocas líneas de código descarga las librerías necesarias para la integración con Google Firebase.

Una vez que se tiene acceso a las funcionalidades es posible utilizar la autenticación, la base de datos Firestore, el almacenamiento Storage y las funciones Cloud Functions.

La autenticación permite registrar y verificar que los usuarios ingresados tengan acceso a la información y a las funciones de la plataforma, el método de autenticación utilizado para esta aplicación es mediante el correo y la contraseña.

Firestore es la base de datos que se encuentra dentro de la plataforma Firebase y ofrece actualización de datos en tiempo real para todos los dispositivos conectados, con la misma API se puede realizar consultas, escribir datos, eliminarlos y actualizarlos. Para sincronizar datos en tiempo real se utiliza la propia plataforma en la nube y es transparente para el desarrollador la actualización de datos.

Storage permite almacenar archivos de los usuarios que utilizan la aplicación, para este proyecto se utiliza el servicio para guardar las imágenes de los vehículos y mostrarlas en la aplicación, la API permite que gracias a su sencillez

se pueda mostrar una imagen con muy pocas líneas de código. Finalmente, Cloud Functions permite al desarrollador ejecutar scripts automatizados, como por ejemplo envío de emails, *triggers*, entre otras.

3.3 Aplicaciones existentes de renta de vehículos

En el mercado actual las aplicaciones para renta de vehículos son varias, para el desarrollo de este proyecto se han tomado como referentes algunas de estas aplicaciones existentes. Algunas de las características en común entre todas las apps es la selección de un rango de fechas, una localización, manejo de usuarios, listas de vehículos y administración de reservas. En las figuras 2 y 3 se visualizan las aplicaciones que se han tomado como referencias para este proyecto.

Pantallas de inicio para realizar la búsqueda:

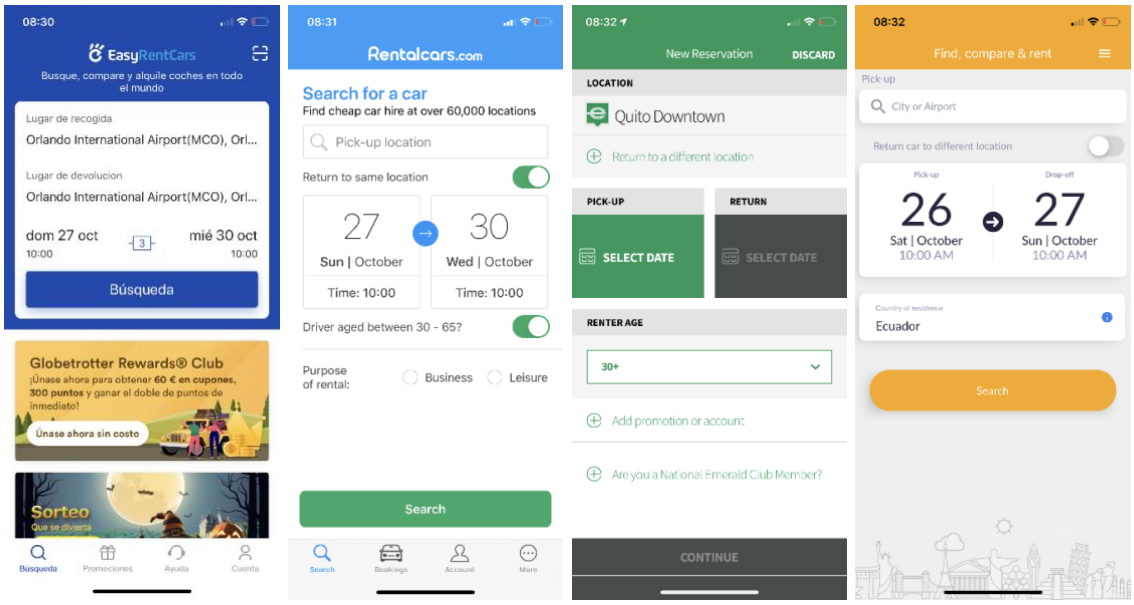


Figura 5. Referencias de otras aplicaciones móviles. Diseño de búsqueda de las aplicaciones EasyRentCars, Rentalcars, Enterprise y Rentcars.com.

Adaptada de (Apple, 2019)

Pantallas con lista de vehículos disponibles:

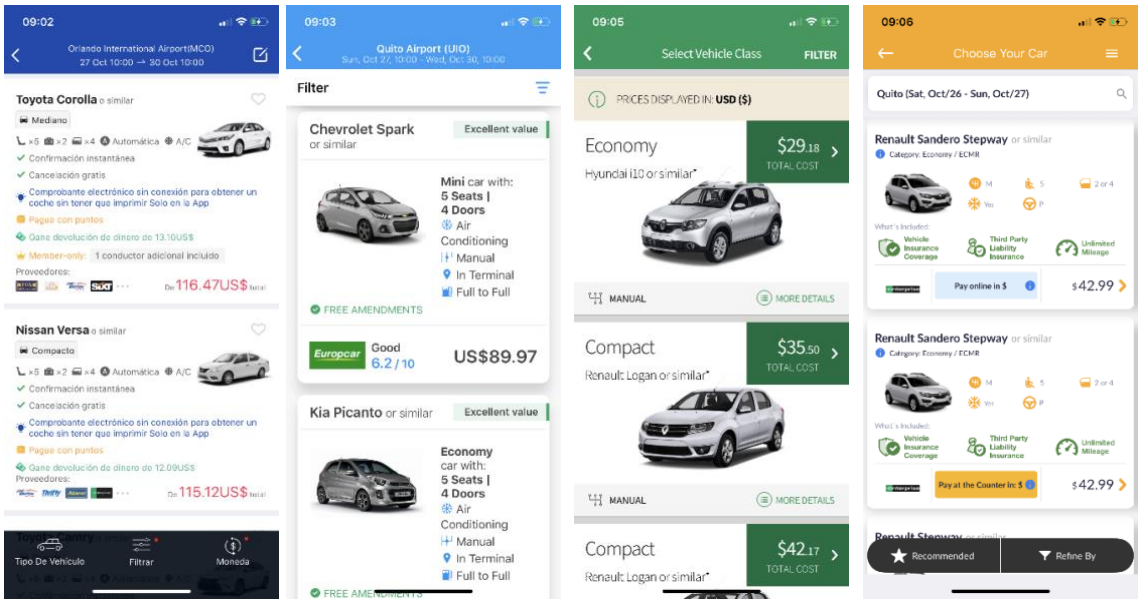


Figura 6. Referencias de otras aplicaciones móviles. Diseño de listas de vehículos de las aplicaciones EasyRentCars, Rentalcars, Enterprise y Rentcars.com.

Adaptada de (Apple, 2019).

La diferencia entre estas aplicaciones y la que se desarrolla en este proyecto es que en esta aplicación los usuarios particulares pueden subir sus vehículos a la plataforma, en las aplicaciones comerciales se realiza una búsqueda de vehículos a nivel empresarial en empresas como Avis, Budget, Rentacar, entre otras. Una vez que se han obtenido las referencias de las principales aplicaciones móviles en el mercado y la forma en cómo trabajan, se procede a realizar las historias de usuario para el desarrollo del proyecto.

3.3.1 Product Backlog

En la figura 7 se observa el Product Backlog con la lista de objetivos planteados por el cliente que representan su visión y expectativas.

Figura 7. Product Backlog. Requerimientos del Product Owner representadas en postits.

Para proceder con el orden de trabajo en el desarrollo de la aplicación móvil, es necesario identificar los *story points* (puntos de valoración técnicos) y los *value points* (puntos de valoración del cliente) para cada una de las historias de usuario, se ha utilizado la serie de Fibonacci, esto se puede apreciar en la Tabla 1 y en la Tabla 2 respectivamente.

Tabla 1

Story Points

Valoración	Historia de usuario
13	Búsqueda de vehículos
8	Administrar reservas del cliente Envío de mail al reservar Lista de vehículos disponibles Borrar reservas no pagadas Menú lateral
5	Cancelar una reserva Validación de fechas de búsqueda Solicitar registro de pago de reserva Reporte de dinero recaudado Detalle de la transacción final Ingresar vehículo para rentar
3	Mensaje de confirmación de la reserva Confirmación de datos del usuario Detalles del vehículo Visualizar vehículos ingresados
2	Modificar datos personales Registro de usuarios en la aplicación Inicio de sesión de usuarios en la aplicación
1	Cierre de sesión de usuarios en la aplicación

Tabla 2

Value Points

Valoración	Historia de usuario
21	Lista de vehículos disponibles Búsqueda de vehículos
13	Registro de usuarios en la aplicación

	Visualizar vehículos ingresados Ingresar vehículo para rentar Administrar reservas del cliente Cancelar una reserva
8	Mensaje de confirmación de la reserva Envío de mail al reservar Detalle de la transacción final
5	Solicitar registro de pago de reserva Reporte de dinero recaudado Menú lateral Validación de fechas de búsqueda
3	Inicio de sesión de usuarios en la aplicación Cierre de sesión de usuarios en la aplicación Confirmación de datos del usuario Modificar datos personales
2	Borrar reservas no pagadas Detalles del vehículo

Una vez que se tiene claro los *story points* y los *value points* se enlista nuevamente todas las historias de usuario, pero ordenadas de mayor a menor con la valoración obtenida en los *value points* para saber que aporta más valor en el desarrollo para el cliente. Para este proyecto se trabaja con iteraciones de 2 semanas en las que se entregue varias historias de usuario por *sprint*.

Para seleccionar el orden de entrega en cada *sprint* se ha seleccionado historias de usuario que aporten valor *value points* y que equilibren el esfuerzo de desarrollo técnico *story points*, también se ha tomado en cuenta que cada día de trabajo tendrá alrededor de 5 a 6 horas, denominando este tiempo como *hora ideal*, toda esta información puede visualizarse en la Tabla 3 a continuación:

Tabla 3

Estimación de las historias de usuario

Nombre de la historia de usuario	Story	Value	Hora Ideal	Sprint
Búsqueda de vehículos	13	21	26	1
Lista de vehículos disponibles	8	21	16	1
Administrar reservas del cliente	8	13	16	2
Cancelar una reserva	5	13	10	2
Ingresar vehículo para rentar	5	13	10	2
Visualizar vehículos ingresados	3	13	6	2
Registro de usuarios en la aplicación	2	13	4	1
Envío de mail al reservar	8	8	16	3
Detalle de la transacción final	5	8	10	2
Mensaje de confirmación de la reserva	3	8	6	3
Menú lateral	8	5	16	3
Validación de fechas de búsqueda	5	5	10	4
Solicitar registro de pago de reserva	5	5	10	4
Reporte de dinero recaudado	5	5	10	4
Confirmación de datos del usuario	3	3	6	3
Modificar datos personales	2	3	4	3
Inicio de sesión de usuarios en la aplicación	2	3	4	1
Cierre de sesión en la aplicación	1	3	2	1
Borrar reservas no pagadas	8	2	16	4
Detalles del vehículo	3	2	6	3

Para resumir las historias de usuario que serán desarrolladas en los 4 *sprints* de 2 semanas de duración cada uno.

Se presentan las tablas de cada iteración en la Tabla 4, Tabla 5, Tabla 6 y Tabla 7 respectivamente.

Tabla 4

Sprint 1

Story Points	Value Points
26	61
Búsqueda de vehículos	
Lista de vehículos disponibles	
Registro de usuarios en la aplicación	
Inicio de sesión de usuarios en la aplicación	
Cierre de sesión de usuarios en la aplicación	

Tabla 5

Sprint 2

Story Points	Value Points
26	60
Administrar reservas del cliente	
Cancelar una reserva	
Ingresar vehículo para rentar	
Visualizar vehículos ingresados	
Detalle de la transacción final	

Tabla 6

Sprint 3

Story Points	Value Points
27	29
Envío de mail al reservar	
Mensaje de confirmación de la reserva	
Menú lateral	
Confirmación de datos del usuario	
Modificar datos personales	
Detalles del vehículo	

Tabla 7

Sprint 4

Story Points	Value Points
23	17
Validación de fechas de búsqueda Solicitar registro de pago de reserva Reporte de dinero recaudado Borrar reservas no pagadas	

3.3.2 Historias de Usuario

Para el desarrollo del proyecto se utilizan las siguientes historias de usuario:

Tabla 8

Historia de usuario de registro de usuarios en la aplicación

3.3.2.1 Registro de usuarios en la aplicación	
ID: TIT001	Usuario: cliente
Nombre de historia: Registro de usuarios en la aplicación	
Prioridad en negocio: 1	Riesgo en desarrollo: bajo
Estimación: 4 horas ideales	Sprint asignado: 1
Responsable: Equipo de desarrollo	
Descripción: Como cliente de la aplicación quiero registrarme en la aplicación para tener una cuenta de usuario.	
Criterios de aceptación: Para un registro inicial se debe colocar el usuario y la contraseña, posteriormente se debe ingresar el nombre, apellido y teléfono en una nueva vista. El usuario será el email y la contraseña, la cual debe estar protegida en la vista con puntos o asteriscos.	

En caso de que la contraseña o el usuario sean incorrectos debe mostrar un mensaje de error informando lo que sucede. De lo contrario, si todo va bien debe mostrarse un mensaje de confirmación de que el usuario se ha creado. Un usuario incorrecto es un mail sin formato y una contraseña debe poseer mínimo 6 caracteres.

El teclado que se muestra al presionar sobre el campo del correo debe ser un teclado para mail que contenga el @ sin cambiar a otro teclado.

En la interfaz de usuario es obligatorio que exista un campo para el correo y la contraseña. También debe haber 2 botones, uno principal que sea para continuar con el registro y otro para ir al login.

En la segunda vista de continuación del registro se ingresan los datos personales del usuario, su nombre, apellido y teléfono. En la parte inferior tiene un botón para completar el registro. Se debe validar que los campos no estén vacíos y la longitud del teléfono sea al menos de 10 dígitos.

Cuando se está creando un nuevo usuario debe aparecer un elemento visual como un *loader* para indicar al cliente que hay un proceso en *background*.

Tabla 9

Historia de usuario de inicio de sesión de usuarios en la aplicación

3.3.2.2 Inicio de sesión de usuarios en la aplicación	
ID: TIT002	Usuario: cliente
Nombre de historia: Inicio de sesión de usuarios en la aplicación	
Prioridad en negocio: 4	Riesgo en desarrollo: bajo
Estimación: 4 horas ideales	Sprint asignado: 1
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero ingresar a la aplicación con mi usuario y contraseña para acceder a las funcionalidades de la aplicación.	

Criterios de aceptación: Si el usuario ya ha sido previamente registrado puede iniciar sesión en la aplicación en cualquiera de sus dispositivos iOS. La pantalla de login tiene dos campos, el primero para el usuario que es el email y el segundo para la contraseña que será ocultada con puntos o asteriscos. Se debe validar que los campos no estén vacíos y que el usuario y la contraseña coincidan con lo registrado en la base de datos, de lo contrario se mostrará un mensaje de error y no se permitirá el inicio de sesión. El teclado que se muestra al presionar sobre el campo del correo debe ser un teclado para mail que contenga el carácter @. Cuando se está iniciando sesión debe aparecer un elemento visual como un loader para indicar al cliente que hay un proceso en background.

Tabla 10

Historia de usuario de cierre de sesión de usuarios en la aplicación

3.3.2.3 Cierre de sesión de usuarios en la aplicación	
ID: TIT003	Usuario: cliente
Nombre de historia: Cierre de sesión de usuarios en la aplicación	
Prioridad en negocio: 4	Riesgo en desarrollo: bajo
Estimación: 2 horas ideales	Sprint asignado: 1
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero cerrar sesión en el dispositivo para salir de mi cuenta en el dispositivo.	
Criterios de aceptación: Si el usuario ha ingresado en la aplicación puede cerrar sesión, para esto debe existir un botón que permita realizar un logout.	

Cuando un usuario no ha iniciado sesión en la aplicación puede realizar la búsqueda de vehículos, sin embargo, al momento de realizar una reserva el mismo proceso le solicitará que inicie sesión o se registre para proceder con la reserva.

Tabla 11

Historia de usuario de búsqueda de vehículos

3.3.2.4 Búsqueda de vehículos	
ID: TIT004	Usuario: cliente
Nombre de historia: Búsqueda de vehículos	
Prioridad en negocio: 1	Riesgo en desarrollo: bajo
Estimación: 26 horas ideales	Sprint asignado: 1
Responsable: Equipo de desarrollo	
Descripción: Como cliente de la aplicación quiero ingresar una fecha de inicio y una de fin para que me aparezcan los vehículos disponibles en una ubicación dada.	
Criterios de aceptación: La fecha de fin no puede ser inferior a la fecha de inicio. Al cargar la aplicación automáticamente debe ponerse la fecha del día actual. La localización es obtenida por la aplicación siempre y cuando se tenga acceso al GPS del celular, de lo contrario se deberá otorgar el permiso en los ajustes de privacidad del dispositivo. Deberá existir un pequeño botón para obtener las coordenadas y posteriormente la ciudad a la cual pertenecen esas coordenadas. Para las fechas de inicio y fin existirán dos botones respectivamente, el texto de cada botón será la fecha seleccionada. Para escoger la fecha se utilizará un calendario que aparecerá en el momento que se presione sobre el botón la fecha.	

Para obtener la ubicación automáticamente se puede usar un botón de ubicación que será presionado por el usuario y obtendrá la ciudad de donde se busca el vehículo.

En la interfaz de usuario es obligatorio que exista un campo para la ciudad, un pequeño botón estilo icono para obtener la ubicación automáticamente, un campo para la fecha de inicio y otro para la de fin y un botón para ejecutar la búsqueda.

Tabla 12

Historia de usuario de lista de vehículos disponibles

3.3.2.5 Lista de vehículos disponibles	
ID: TIT005	Usuario: cliente
Nombre de historia: Lista de vehículos disponibles	
Prioridad en negocio: 1	Riesgo en desarrollo: bajo
Estimación: 16 horas ideales	Sprint asignado: 1
Responsable: Equipo de desarrollo	
Descripción: Como cliente de la aplicación quiero ver una lista de vehículos disponibles en las fechas ingresadas para poder seleccionar uno.	
Criterios de aceptación: Se debe visualizar una lista con los autos que coincidan con la ciudad y rango de fechas previamente ingresadas. En cada uno de los elementos que conforman la lista se debe visualizar la marca y el modelo, el número de asientos, el número de puertas, si posee o no aire acondicionado, el tipo de transmisión del vehículo (manual o automático), una imagen miniatura referencial del vehículo, el precio diario y el precio total. Se muestran ordenados por precio de menor a mayor.	

Tabla 13

Historia de usuario de detalles del vehículo

3.3.2.6 Detalles del vehículo	
ID: TIT006	Usuario: cliente
Nombre de historia: Detalles del vehículo	
Prioridad en negocio: 4	Riesgo en desarrollo: bajo
Estimación: 6 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero ver todos los detalles adicionales del vehículo para poder obtener toda la información del auto que busco.	
Criterios de aceptación: En el detalle del vehículo se deben visualizar las características de forma más detallada, una imagen más grande del vehículo, el número de personas máximo, la transmisión, el número de puertas, si cuenta con aire acondicionado y el espacio para las maletas. Se mostrará una imagen del modelo del vehículo. No necesariamente es la foto real del vehículo, solo del modelo. También se visualizará la ciudad y el rango de fechas ingresadas en un inicio. En la parte inferior se visualizará el precio total a pagar y un botón para continuar con el proceso de reserva. Debe existir un botón para volver a la lista de vehículos. Al pulsar sobre el botón continuar se desplegarán en una nueva vista los datos del usuario en caso de querer actualizar, confirmar o incluso registrar.	

Tabla 14

Historia de usuario de confirmación de datos del usuario

3.3.2.7 Confirmación de datos del usuario	
ID: TIT007	Usuario: cliente

Nombre de historia: Confirmación de datos del usuario	
Prioridad en negocio: 4	Riesgo en desarrollo: bajo
Estimación: 6 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero confirmar mis datos personales para proceder con la reserva.	
Criterios de aceptación: Esta vista aparece en el proceso de reserva, en el caso de que el usuario este registrado y haya iniciado sesión aparecerán precargados sus datos personales (nombre, apellido y teléfono). Si el usuario está de acuerdo con esa información solo deberá presionar el botón de continuar para seguir con la reserva. Si el usuario no está de acuerdo con esos datos puede modificarlos en ese momento y actualizar sus datos personales, al momento de presionar el botón continuar los datos serán actualizados automáticamente. Si el usuario no se ha registrado en ese momento puede hacerlo, se ingresarán sus datos personales y en una nueva vista se ingresará el email y la contraseña, es decir, se realizará el registro como parte del proceso de reserva y luego se continuará la reserva sin tener que volver a empezar desde cero. En la parte inferior existirá un botón para continuar con el proceso de reserva.	

Tabla 15

Historia de usuario de detalle de la transacción final

3.3.2.8 Detalle de la transacción final	
ID: TIT008	Usuario: cliente

Nombre de historia: Detalle de la transacción final	
Prioridad en negocio: 3	Riesgo en desarrollo: bajo
Estimación: 10 horas ideales	Sprint asignado: 2
Responsable: Equipo de desarrollo	
Descripción: Como cliente de la aplicación quiero ver el resumen de toda la transacción para confirmar la reserva.	
Criterios de aceptación: Debe visualizarse el resumen del vehículo seleccionado con sus características principales, una mini imagen del vehículo seleccionado, las fechas de inicio y de fin y la ciudad. También debe visualizarse los datos personales del cliente. En la parte inferior se visualizará el precio final, un botón para confirmar la reserva y otro para volver al inicio. En el caso de que se pulse sobre el botón de Reservar aparecerá un mensaje indicando que la reserva se ha realizado con éxito y luego se volverá al inicio de la aplicación. En el caso de que se pulse sobre el botón de volver al inicio la reserva no se realizará y se volverá a la pantalla de inicio para ingresar nuevamente los datos. Previo a volver al inicio se confirmará con el usuario si en realidad quiere realizar esa acción.	

Tabla 16

Historia de usuario de confirmación de la reserva

3.3.2.9 Mensaje de confirmación de la reserva	
ID: TIT009	Usuario: cliente
Nombre de historia: Mensaje de confirmación de la reserva	
Prioridad en negocio: 3	Riesgo en desarrollo: bajo
Estimación: 6 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	

Descripción: Como cliente de la aplicación quiero comprobar que la reserva se hizo para comprobar que el proceso fue exitoso.
Criterios de aceptación: Cuando se presionó el botón de confirmar la reserva se debe crear un nuevo registro en la base de datos que tenga la información de la reserva (fechas y valores), con el vehículo y el usuario. Debe aparecer un mensaje al finalizar la reserva, puede ser un mensaje de error o uno indicando que todo salió bien. En *background* se ingresa una reserva en la base de datos y se envía un mail al propietario informando sobre la nueva reserva. Si todo sale bien se debe redirigir a la pantalla de inicio.

Tabla 17

Historia de usuario de envío de mail al reservar

3.3.2.10 Envío de mail al reservar	
ID: TIT010	**Usuario:** propietario
Nombre de historia: Envío de mail al reservar	
Prioridad en negocio: 2	Riesgo en desarrollo: bajo
Estimación: 16 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	
Descripción: Como propietario de un vehículo quiero recibir un correo electrónico para tener información sobre la reserva realizada.	
Criterios de aceptación: Cuando se realiza una reserva por parte de un cliente en la aplicación se debe notificar inmediatamente al propietario sobre la nueva reserva. El mail debe incluir el rango de fechas y el valor total de la reserva. El criterio de aceptación para esta historia es que funcione al menos con Gmail.	

Tabla 18

Historia de usuario de administrar reservas del cliente

3.3.2.11 Administrar reservas del cliente	
ID: TIT011	Usuario: cliente
Nombre de historia: Administrar reservas del cliente	
Prioridad en negocio: 2	Riesgo en desarrollo: bajo
Estimación: 16 horas ideales	Sprint asignado: 2
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero visualizar las reservas de vehículos que he realizado para administrar cada una de ellas.	
Criterios de aceptación: En la lista deben aparecer las reservas que se han hecho en la aplicación. En cada uno de los elementos que conforman la lista se debe visualizar la marca y el modelo, el número de asientos, el número de puertas, si posee o no aire acondicionado, el tipo de transmisión del vehículo (manual o automático), una imagen miniatura del vehículo, la fecha de inicio y el precio total. La lista debe estar ordenada desde las más antiguas hasta las más recientes. Al pulsar sobre un elemento de la lista se obtendrá un detalle de la reserva, y dentro de cada una se podrá realizar una cancelación.	

Tabla 19

Historia de usuario de cancelar una reserva

3.3.2.12 Cancelar una reserva	
ID: TIT012	Usuario: cliente
Nombre de historia: Cancelar una reserva	
Prioridad en negocio: 2	Riesgo en desarrollo: bajo

Estimación: 10 horas ideales	Sprint asignado: 2
Responsable: Equipo de desarrollo	
Descripción: Como cliente de la aplicación quiero tener la posibilidad de poder cancelar una reserva hecha previamente para dejar el carro libre para otra persona.	
Criterios de aceptación: La cancelación de reserva se hace desde el detalle de la lista de reservas hechas por el cliente. El cliente ingresa en una reserva y se visualiza el botón de cancelar reserva. Se puede cancelar una reserva con 3 días anteriores a la fecha de reserva o que aún no ha sido pagada. Cuando se hace clic sobre ese botón aparece un mensaje de confirmación preguntando al usuario si en realidad quiere cancelar la reserva. Cuando se cancela la reserva se elimina el registro de la base de datos y se regresa a la vista anterior de lista de reservas.	

Tabla 20

Historia de usuario de modificar datos personales

3.3.2.13 Modificar datos personales	
ID: TIT013	Usuario: cliente
Nombre de historia: Modificar datos personales	
Prioridad en negocio: 4	Riesgo en desarrollo: bajo
Estimación: 4 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero poder modificar mis datos personales para tener mi información actualizada.	

Criterios de aceptación: Para permitir actualizar los datos personales es indispensable que el cliente haya iniciado sesión en la aplicación. Debe existir una vista con 3 campos para el nombre, apellido y teléfono respectivamente. También un botón que diga Actualizar para proceder con la actualización de datos del usuario. Se debe validar que los campos no estén vacíos, y el teléfono tenga al menos 10 dígitos.

Tabla 21

Historia de usuario de ingresar vehículo para rentar

3.3.2.14 Ingresar vehículo para rentar	
ID: TIT014	Usuario: cliente
Nombre de historia: Ingresar vehículo para rentar	
Prioridad en negocio: 2	Riesgo en desarrollo: bajo
Estimación: 10 horas ideales	Sprint asignado: 2
Responsable: Equipo de desarrollo	
Descripción: Como cliente de la aplicación quiero ingresar un vehículo para que esté disponible para rentar.	
Criterios de aceptación: Cualquier usuario registrado en la aplicación tiene la posibilidad de convertirse en propietario, para ello debe dirigirse a la opción de “Lista tus vehículos”. Para iniciar con el proceso el usuario deberá seleccionar una marca y un modelo de una lista que se desplegará, posteriormente aparecerá un formulario en el cual se ingresará el valor diario, el tipo de transmisión (manual o automático), el número de puertas y de pasajeros.	

Cuando se ingresa el precio del vehículo por parte del dueño, se calcula en un porcentaje de 20% de comisión para el dueño de la aplicación (desarrollador), y se colocará automáticamente en el precio final visible para el cliente el precio calculado del vehículo.
Se debe validar que el precio ingresado sea mayor a cero.

En la parte inferior existirá un botón para Registrar el vehículo.

Una vez que se presiona el botón de registro se crea un nuevo vehículo en la base de datos, y en el caso de que el cliente haya ingresado un vehículo por primera vez, se convertirá en propietario.

Tabla 22

Historia de usuario de visualizar vehículos ingresados

3.3.2.15 Visualizar vehículos ingresados	
ID: TIT015	**Usuario:** propietario
Nombre de historia: Visualizar vehículos ingresados	
Prioridad en negocio: 2	Riesgo en desarrollo: bajo
Estimación: 6 horas ideales	Sprint asignado: 2
Responsable: Equipo de desarrollo	
Descripción: Como usuario propietario quiero ver una lista de mis vehículos y conocer el estado de estos para conocer si están reservados o no.	
Criterios de aceptación: Si el usuario que ha iniciado sesión en la aplicación tiene el rol de propietario debe aparecer una opción que permita visualizar sus vehículos. Cuando se tiene vehículos registrados deben aparecer en una lista. En cada uno de los elementos que conforman la lista se debe visualizar la marca y el modelo, el número de asientos, el número de puertas, si posee o no aire acondicionado,	

el tipo de transmisión del vehículo (manual o automático), una imagen miniatura del vehículo, y el precio diario.

Cuando se hace clic sobre uno de ellos aparecen los nombres de las personas que han reservado y si esta pagado o no.

Dentro de la lista de nombres se puede realizar una búsqueda por nombres o apellidos o filtrar por los que ya han sido pagados o no. Si se pulsa sobre uno de los nombres, aparece una ventana con la información personal del cliente (email y teléfono) en el caso de que ya haya pagado, de lo contrario aparece la información personal del dueño de la aplicación.

El propietario puede eliminar un vehículo siempre y cuando no tenga reservas sobre uno de ellos.

Tabla 23

Historia de usuario de solicitar registro de pago de reserva

3.3.2.16 Solicitar registro de pago de reserva	
ID: TIT016	Usuario: cliente
Nombre de historia: Solicitar registro de pago de reserva	
Prioridad en negocio: 3	Riesgo en desarrollo: bajo
Estimación: 10 horas ideales	Sprint asignado: 4
Responsable: Equipo de desarrollo	
Descripción: Como cliente quiero solicitar el registro el pago de una reserva para constatar el pago con el desarrollador.	
Criterios de aceptación: Dentro del detalle de cada reserva el cliente puede solicitar el registro de pago de una reserva mediante un botón, el cual le confirmará si está seguro de hacerlo.	

En caso de solicitar el registro del pago, no se permitirá solicitar de nuevo. Si ya se ha solicitado si es posible anular la solicitud. Solo se permitirá solicitar en las reservas que aún no están pagadas. En caso de que haya un error, se puede anular el pago para que quede nuevamente como no pagada.

Tabla 24

Historia de usuario de reporte de dinero recaudado

3.3.2.17 Reporte de dinero recaudado	
ID: TIT017	**Usuario:** propietario
Nombre de historia: Reporte de dinero recaudado	
Prioridad en negocio: 3	Riesgo en desarrollo: bajo
Estimación: 10 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	
Descripción: Como usuario propietario de un vehículo quiero ver el total de dinero recaudado en un lapso para conocer las ganancias.	
Criterios de aceptación: Cuando el usuario es propietario debe visualizarse un botón que indique el dinero recaudado. Cuando se hace clic sobre el botón aparece una nueva pantalla en la cual se despliegan los valores recaudados hasta el momento y por recaudar.	

Tabla 25

Historia de usuario de menú lateral

3.3.2.18 Menú lateral	
ID: TIT018	**Usuario:** cliente y propietario
Nombre de historia: Menú lateral	
Prioridad en negocio: 3	Riesgo en desarrollo: bajo
Estimación: 16 horas ideales	Sprint asignado: 3
Responsable: Equipo de desarrollo	

Descripción: Como usuarios de la aplicación queremos un menú lateral para seleccionar diferentes opciones.

Criterios de aceptación:

En el dispositivo debe aparecer un menú lateral que contenga las siguientes opciones de acuerdo con el rol.

Como cliente:

- Inicio
- Reservas
- Perfil
- Logout
- Lista tus vehículos

Como propietario:

- Inicio
- Reservas
- Perfil
- Logout
- Lista tus vehículos
- Mis vehículos
- Balance

Como usuario que no ha iniciado sesión:

- Inicio

Tabla 26

Historia de usuario de validación de fechas de búsqueda

3.3.2.19 Validación de fechas de búsqueda	
ID: TIT019	**Usuario:** propietario
Nombre de historia: Validación de fechas de búsqueda	
Prioridad en negocio: 3	Riesgo en desarrollo: bajo
Estimación: 10 horas ideales	Sprint asignado: 4

Responsable: Equipo de desarrollo
Descripción: Como propietario de vehículos quiero que el rango de fechas seleccionado para la búsqueda no sea superior a 3 meses para tener un mejor control sobre las reservas.
Criterios de aceptación: En vista de que los vehículos puedan darse de baja durante un periodo de tiempo cercano, se solicita que las fechas para realizar una reserva no sean mayores a 3 meses a partir de la fecha actual, es decir, si la fecha actual es 1 de enero, se podrá realizar una reserva con fecha de inicio hasta antes del 1 de marzo. Otro criterio de validación de fechas es que no se puede seleccionar una fecha anterior a la actual.

Tabla 27

Historia de usuario de borrar reservas no pagadas

3.3.2.20 Borrar reservas no pagadas	
ID: TIT020	**Usuario:** propietario
Nombre de historia: Borrar reservas no pagadas	
Prioridad en negocio: 4	Riesgo en desarrollo: bajo
Estimación: 16 horas ideales	Sprint asignado: 4
Responsable: Equipo de desarrollo	
Descripción: Como usuario propietario de un vehículo quiero que se eliminen las reservas no pagadas antes de los 3 días de efectuarse la reserva para que los vehículos reservados queden como disponibles de nuevo.	
Criterios de aceptación: Todos los días de forma automática tienen que revisarse las reservas y comprobar si dentro del rango de los 3 días posteriores, incluyendo al actual hay alguna reserva sin pagar, si es así debe eliminarse, dejando libre al vehículo. Este procedimiento debe ser automático y transparente para todos los usuarios.	

3.3.3 Sprint Cero

Antes de iniciar con los Sprints, es importante tener en cuenta la arquitectura de diseño de software que se utilizará dentro de la aplicación. Este proyecto debe adaptarse al crecimiento, ser fácilmente escalable y mantenible, para ello se utilizará el patrón de diseño Modelo – Vista – Controlador (MVC).

En el Modelo se crearán las clases o estructuras que interactúan directamente con los datos almacenados en la base de datos Firestore en la nube de Google.

En el Controlador se tendrán todas las clases que se encargan de la lógica de negocio.

En la Vista se incluirán los storyboards con las interfaces de usuario, tomando en cuenta las restricciones de diseño que deben emplearse.

Para el desarrollo de este proyecto se busca que el diseño de interfaces sea uniforme, con una apariencia limpia, y con una tipografía legible que no canse a los ojos del usuario.

3.3.3.1 Paleta de colores

Según Lara, el color verde es tranquilizante, equilibrado y funcional, funciona como color neutro ante los extremos del rojo y azul. Busca promover en la aplicación tranquilidad al conducir y funcionalidad al buscar. (Lara, 2011).

Adicional se ha incluido el color gris para los textos y blanco para dar una apariencia tranquila y limpia.

En la figura 8 se visualiza la paleta de colores permitidos en el diseño de interfaces de usuario de la aplicación con su respectivo código hexadecimal.

Figura 8. Paleta de colores.

3.3.3.2 Tipografía

Nunito Sans es una fuente gratuita de Google diseñada por Vernon Adams y Jacques Le Bailly. Es una superfamilia tipográfica sans serif bien equilibrada que se complementa con la interfaz.

El uso de una sola familia tipográfica es recomendado como una buena práctica de diseño.

En la figura 9 se visualiza la tipografía permitida en el diseño de interfaces de usuario de la aplicación con sus respectivos tamaños de fuente.

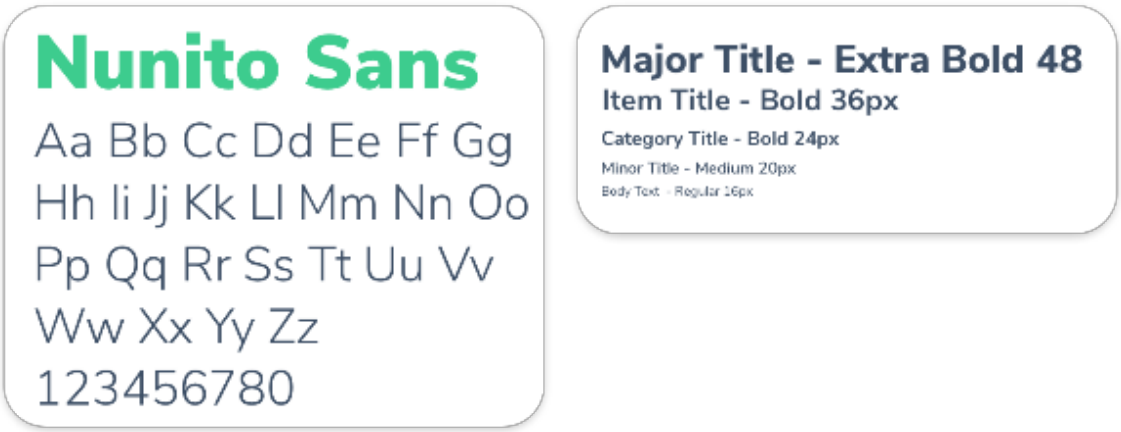


Figura 9. Tipografía Nunito Sans.

Adaptado de (Google Fonts, 2019).

3.3.3.3 Iconos

Para los iconos internos de la aplicación se utilizará SF Symbols, que proporciona un conjunto de más de 1,500 símbolos consistentes y altamente configurables que puede usar en la aplicación.

En la figura 10 Se visualiza la familia de iconos permitida en el diseño de interfaces de usuario de la aplicación.

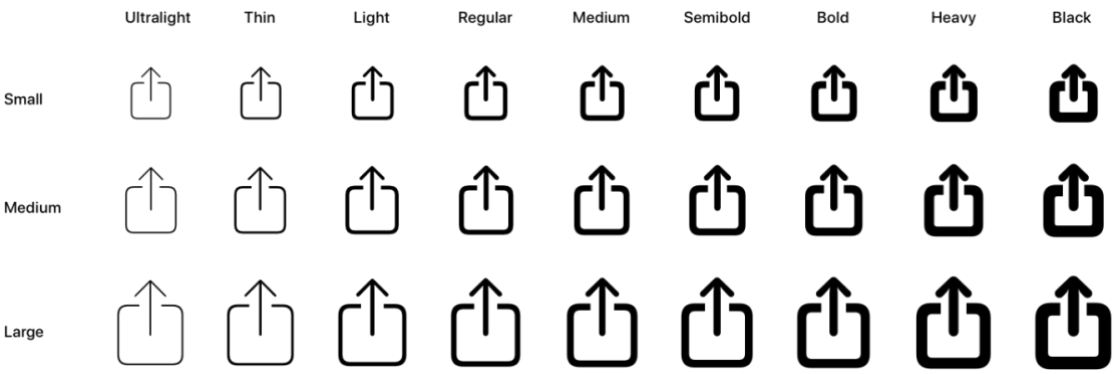


Figura 10. Iconos.

Adaptado de (Apple, 2019).

3.3.3.4 Icono principal de la app

El icono para la aplicación es el siguiente:

Figura 11. Icono de la aplicación.

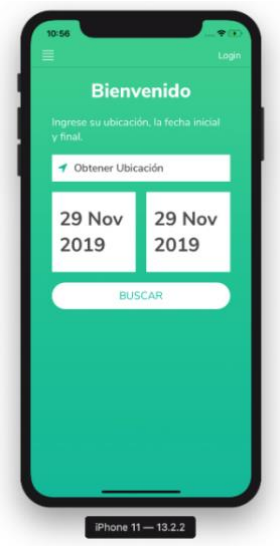
3.3.3.5 Icono en dispositivo

La aplicación se verá de la siguiente forma en el dispositivo:

Figura 12. Icono de la aplicación en el dispositivo

3.3.4 Resultados del Sprint 1

TIT004 - Búsqueda de vehículos



La búsqueda de vehículos se caracteriza por tener una localización, una fecha de inicio y una fecha de fin para la reserva de un vehículo, como se aprecia en la figura 13.

Solo se obtienen los vehículos disponibles para las fechas solicitadas en la pantalla de búsqueda.

Figura 13. Pantalla de búsqueda en la aplicación. Interfaz para obtener la ubicación y seleccionar el rango de fechas.

La funcionalidad consiste en ir a la colección de vehículos en Firestore, figura 14, buscar dentro de las reservas de cada vehículo que no existan reservas en las mismas fechas de búsqueda y consultar los vehículos disponibles.

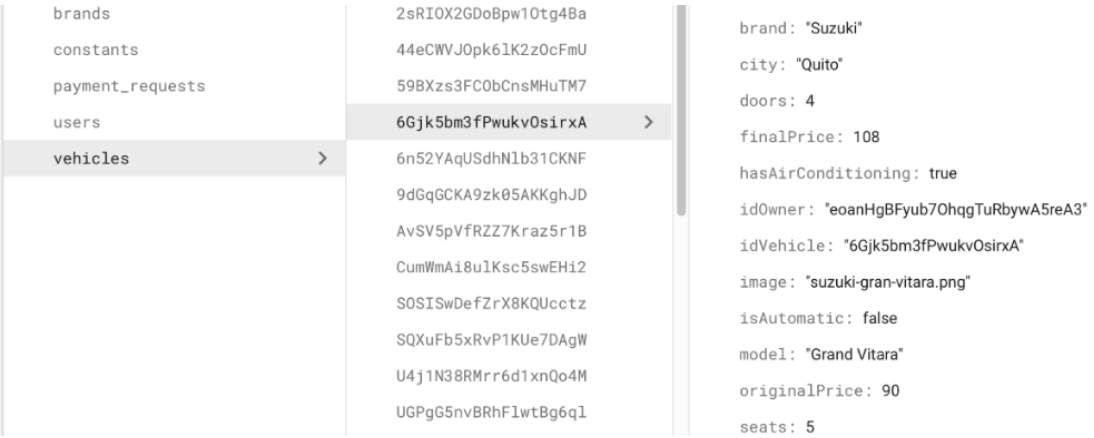
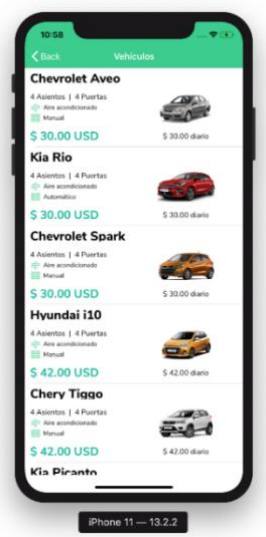


Figura 14. Colección de vehículos en Firestore. Datos almacenados de vehículos.

TIT005 - Lista de vehículos disponibles

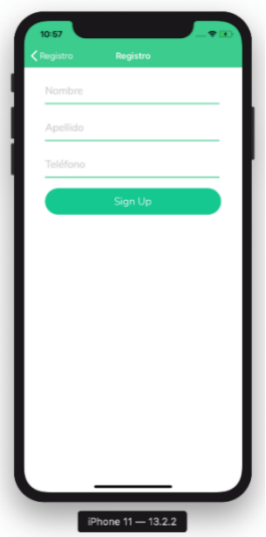


Después de que ha obtenido los vehículos disponibles dadas las fechas y la localización, se despliegan en una lista los resultados de disponibilidad, como en la figura 15.

Cada fila tiene la marca y modelo del vehículo, una breve descripción de sus características, el precio diario debajo de la imagen y el precio total dado el rango de fechas. La lista esta ordenada de menor a mayor precio.

Figura 15. Lista de vehículos disponibles. Lista obtenida de vehículos disponibles.

TIT001 - Registro de usuarios en la aplicación



La funcionalidad de registro de usuarios en la aplicación, como se ve en la figura 16, consiste en que el usuario ingrese sus datos personales (nombre, apellido y teléfono) y su email y contraseña que servirán para iniciar sesión una vez que se ha creado el usuario en la base de datos.

Firebase provee de una API al desarrollador que permite el registro de usuarios con email y contraseña, otorgando un nivel de autenticación para acceder a los datos almacenados.

Figura 16. Registro de usuarios en la aplicación. Interfaz para registrar usuarios en la aplicación.

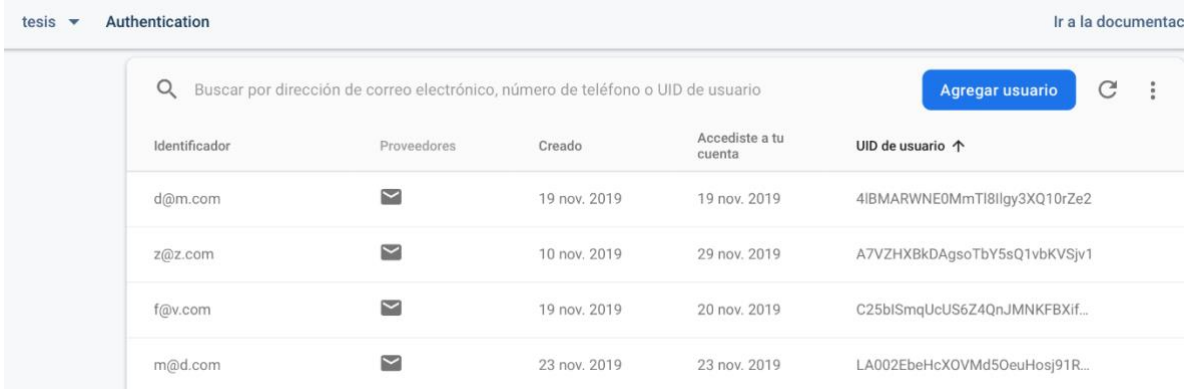


Figura 17. Usuarios registrados en la Firebase.

En la figura 17 se visualiza los usuarios registrados en Firebase que tienen acceso a los datos, con su respectivo UID. (User ID).

Al mismo tiempo que se registran los usuarios en Firebase, también se agregan en una colección llamada *users* en donde se tiene más detalles de cada usuario, su nombre, apellido, mail, si es propietario, su número telefónico y su identificador de usuario.

En la figura 18 Se visualiza la colección users con los registros de cada usuario almacenado en la base de datos.

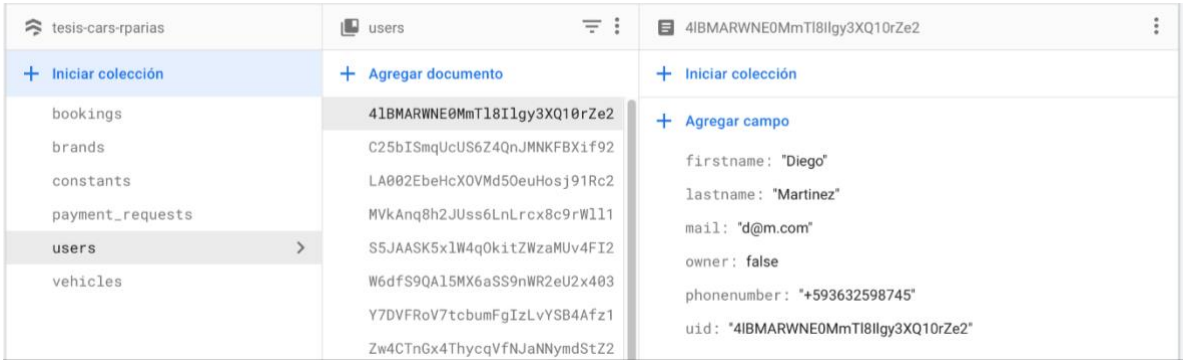
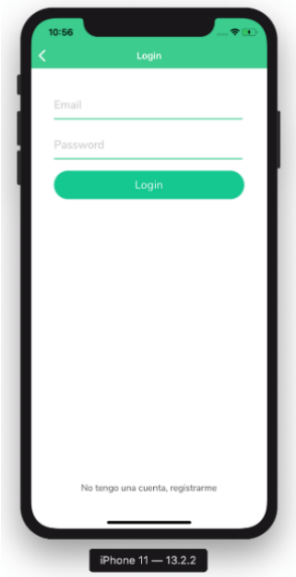


Figura 18. Colección de usuarios en Firestore. Colección users con los registros de cada usuario.

TIT002 - Inicio de sesión de usuarios en la aplicación

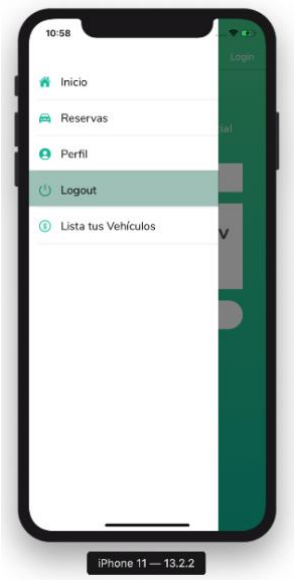


Una vez que los usuarios han sido registrados tienen la posibilidad de iniciar sesión en la aplicación como se ve en la figura 19, para ello se tiene una vista con dos campos: email y contraseña. Desde la misma ventana se puede ir a la sección de registro en caso de que el usuario no esté registrado en la app

Con la misma API de Firebase se hace la autenticación para verificar que el usuario y contraseña coincidan con los registrados en la base de datos.

Figura 19. Login de usuarios en la aplicación. Interfaz para iniciar sesión en la aplicación.

TIT003 - Cierre de sesión de usuarios en la aplicación



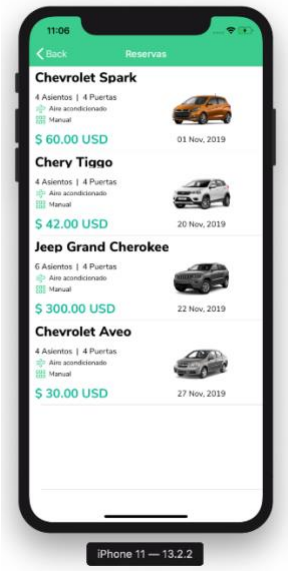
Para cerrar la sesión de usuario, se verifica que el usuario actual haya ingresado al sistema, luego se utiliza la API de Firebase para realizar un *logout* y salir.

En la figura 20 se visualiza el menú lateral con las opciones, entre ellas, la de cerrar sesión.

Figura 20. Botón para cerrar sesión. Menú lateral con la opción de cerrar sesión.

3.3.6 Resultados del Sprint 2

TIT011 - Administrar reservas del cliente



Como cliente se puede visualizar las reservas que se ha realizado desde que creó una cuenta en la aplicación, se puede visualizar las reservas en la figura 21.

Se presenta una lista de vehículos sobre los cuales se ha realizado una reserva, al momento de seleccionar alguna de las reservas se ingresa en una nueva pantalla que contiene los detalles de la reserva y también ofrece la posibilidad de cancelarla.

Figura 21. Reservas. Lista de reservas sobre los vehículos que ha realizado un cliente.

TIT012 - Cancelar una reserva

En el detalle de cada reserva sobre un vehículo, el cliente tiene la posibilidad de cancelar una reserva. En la parte superior derecha existe un botón para cancelar una reserva, como en la figura 22.

El momento que el cliente hace clic sobre el icono del basurero aparece un mensaje de confirmación preguntando si desea cancelar la reserva.

Figura 22. Cancelación de una reserva. Interfaz para cancelar una reserva por parte del cliente.

TIT014 - Ingresar vehículo para rentar

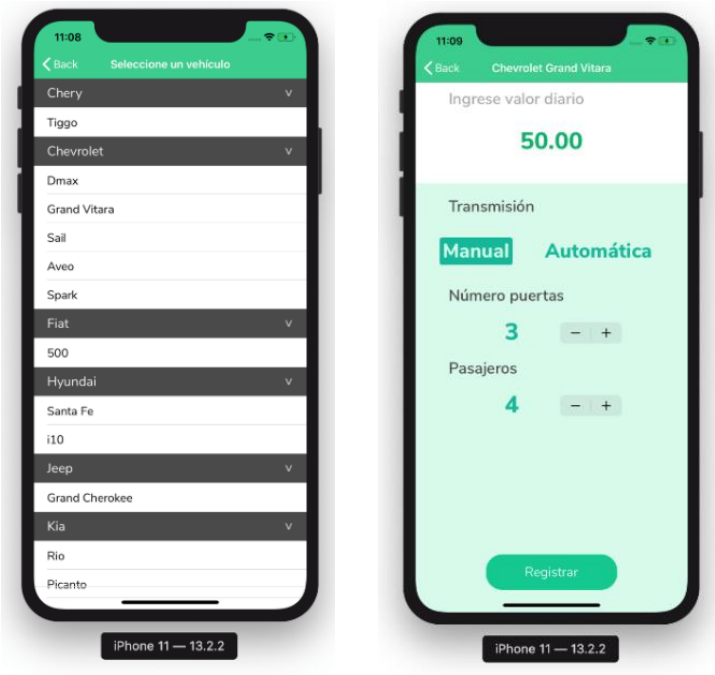


Figura 23. Ingreso de vehículos para rentar. Interfaz para registrar un nuevo vehículo por parte de los usuarios.

En la figura 23 se visualiza la interfaz para registrar un nuevo vehículo por parte de los usuarios. Como usuario de la aplicación se puede tener dos roles: cliente y propietario.

El cliente es aquel que busca un vehículo para rentar y no dispone de vehículos para rentar a otros usuarios. El propietario es el usuario que tiene vehículos para rentar a disposición de los clientes.

Un usuario cliente puede convertirse en propietario en el momento que tenga al menos un vehículo registrado y esté disponible para otros usuarios. Si el usuario elimina todos sus vehículos vuelve a convertirse en usuario cliente.

Para ingresar un nuevo vehículo se debe seleccionar el modelo de acuerdo con las marcas visualizadas, el precio diario, el tipo de transmisión, el número de puertas y la capacidad de pasajeros del vehículo.

TIT015 - Visualizar vehículos ingresados

Como propietario se puede acceder a cada uno de los vehículos registrados para conocer las reservas sobre ellos, o para eliminar el vehículo. Si se ingresa sobre un vehículo, aparece una lista de los clientes que han pagado y los que aún no han pagado, como se observa en la figura 24.

Se debe tener en cuenta que no se puede eliminar un vehículo sobre el cual hay reservas activas.
Para eliminar un vehículo se desplaza el dedo de derecha a izquierda.

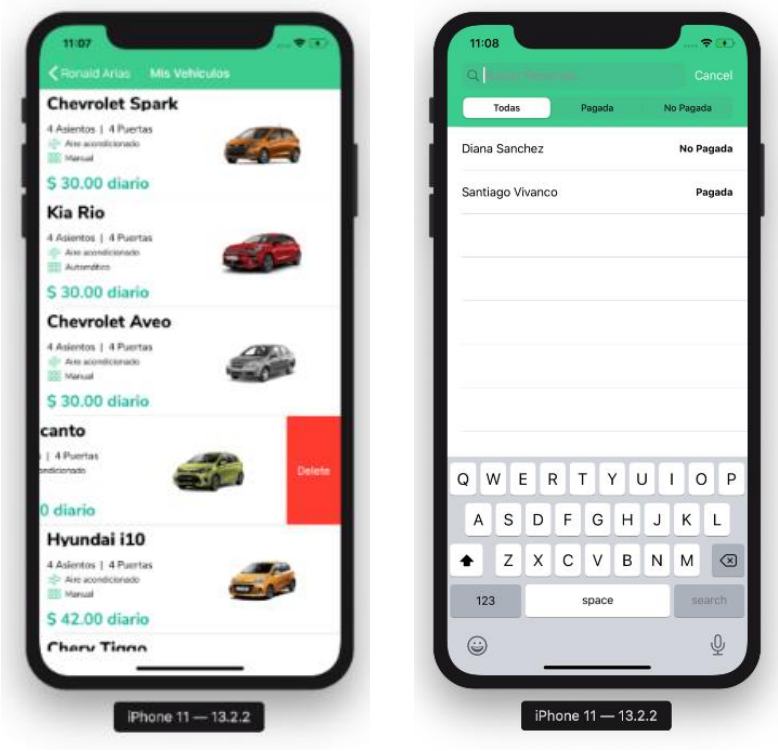
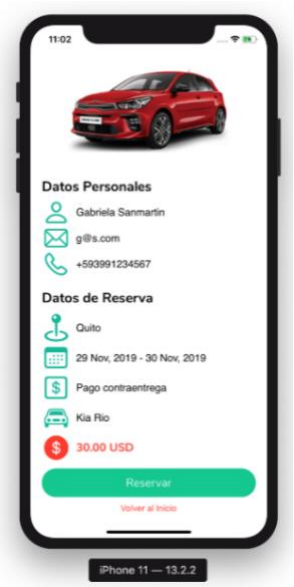


Figura 24. Visualizar vehículos del propietario. Interfaz para administrar los vehículos del propietario.

TIT008 - Detalle de la transacción final



Después de confirmar los datos personales del usuario, se muestra una pantalla de confirmación final, en la cual consta la información del cliente, la ciudad, las fechas y valor total de la reserva, como se observa en la figura 25.

También hay dos botones, uno para confirmar la reserva y otro para cancelar y volver al inicio.

Figura 25. Resumen de la reserva. Datos que serán generados para la reserva.

3.3.7 Resultados del Sprint 3

TIT010 - Envío de mail al reservar

para mí

Detalle de la Reserva

Email: rocko1192@gmail.com
Desde: Fri Nov 29 2019 05:00:00 GMT+0000 (UTC)
Hasta: Sat Nov 30 2019 05:00:00 GMT+0000 (UTC)
Valor USD: 30

Figura 26. Registro de usuarios en la aplicación. Se visualiza la interfaz para registrar usuarios en la aplicación.

Cuando se crea un nuevo documento dentro de la coleccion *bookings* existe un *trigger* (figura 27) que se dispara automáticamente y envía un correo electrónico al mail registrado, como se ve en la figura 26, indicando que se ha generado una reserva.

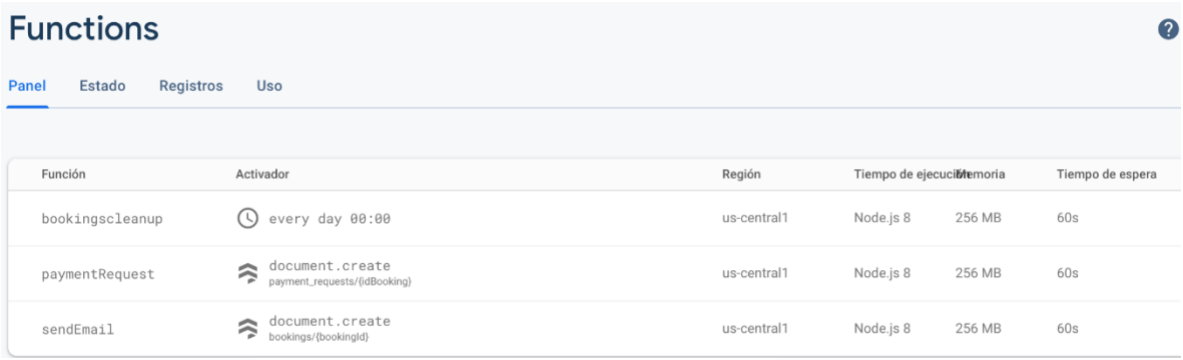


Figura 27. Función sendEmail. Se visualiza la función de enviar correo automáticamente al generar una nueva reserva.

TIT009 - Confirmación de la reserva

Cuando se ha presionado el botón de Confirmar aparece un mensaje de confirmación indicado que la reserva se ha realizado correctamente, como se observa en la figura 28.

En el momento que se realiza una reserva se crea un nuevo documento en la colección de reservas (*bookings*), como se observa en la figura 29.

Figura 28. Confirmación de la reserva. Mensaje de confirmación de la reserva.

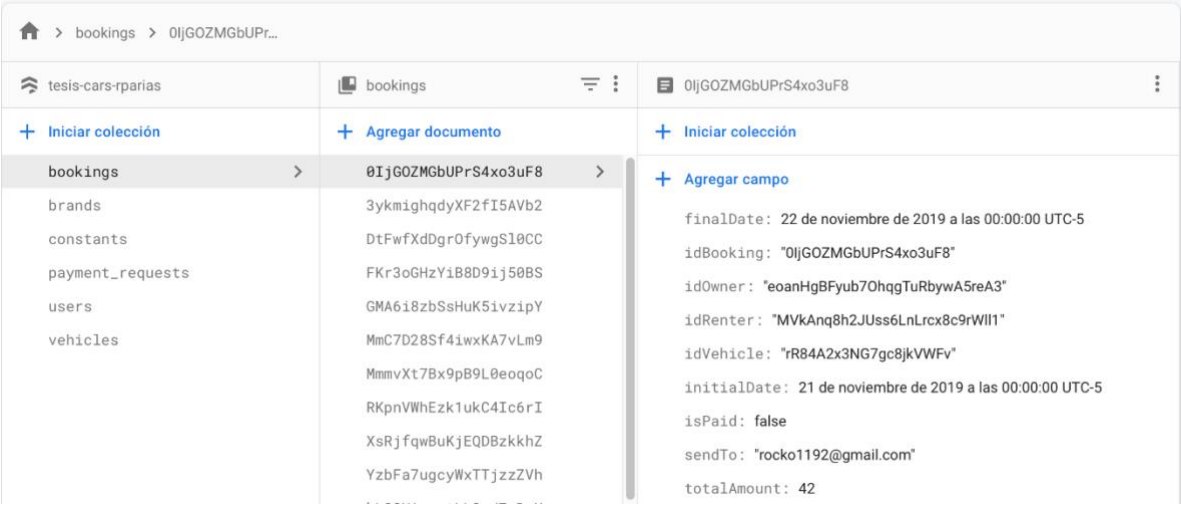


Figura 29. Registro de usuarios en la aplicación. Se visualiza la interfaz para registrar usuarios en la aplicación.

TIT018 - Menú lateral

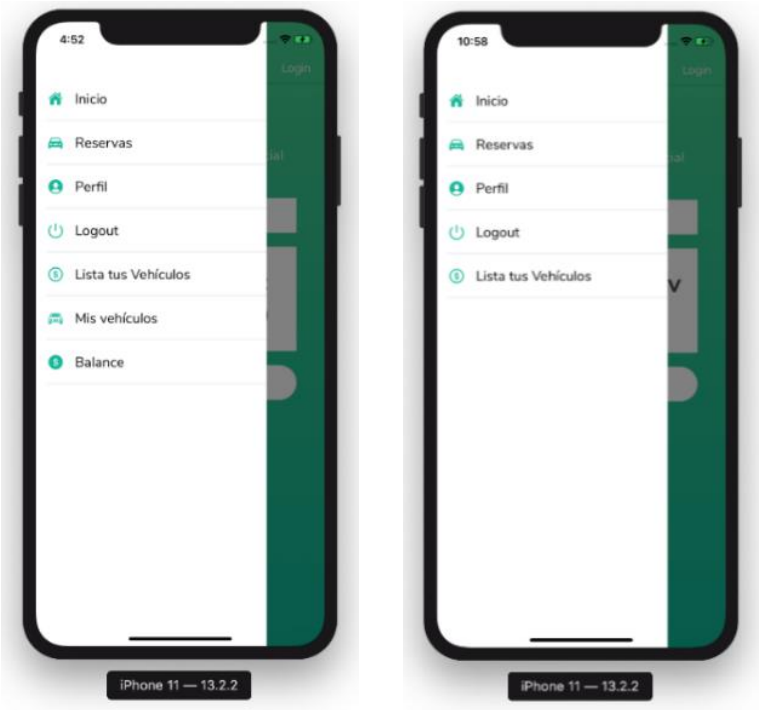
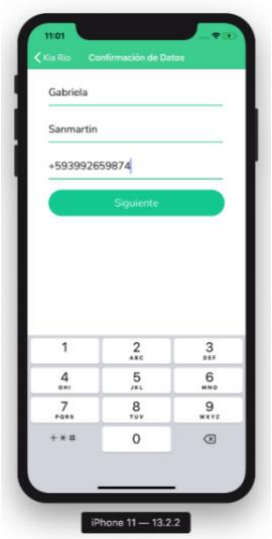


Figura 30. Menú lateral. Propietarios menú izquierda, Clientes menú derecha.

En dependencia del tipo de usuario (cliente o propietario) se visualiza el menú, como se observa en la figura 30.

TIT007 - Confirmación de datos del usuario

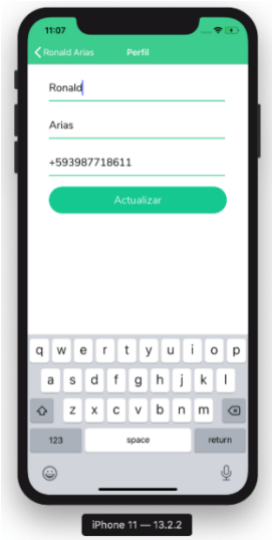


Para continuar con el proceso de reserva se debe verificar que los datos del usuario estén correctos. En el caso de que el usuario ya haya iniciado sesión con anterioridad se desplegaran sus datos automáticamente, teniendo la opción de modificar si lo desea, como se observa en la figura 31.

En el caso de que el usuario no haya iniciado sesión, el proceso de reserva lo conducirá a la pantalla de registro y el proceso continuará.

Figura 31. Confirmación de datos del usuario. Datos personales del usuario para confirmarlos.

TIT013 - Modificar datos personales



Como usuario de la aplicación existe la funcionalidad de actualizar los datos personales, para ello se debe acceder a la sección del perfil y aparece una ventana con los datos personales del cliente, como en la figura 32. Aparecen 3 campos para modificar el nombre, el apellido y el teléfono.

Desde el menú lateral, Figura 30, se ingresa a Perfil.

Figura 32. Actualización del perfil. Se visualiza la interfaz para actualizar los datos personales del usuario en la aplicación.

TIT006 - Detalles del vehículo

De la lista de vehículos anterior se puede seleccionar uno de los elementos y obtener a mayor detalle los datos del vehículo y una mejor visualización antes de continuar en el proceso, como se ve en la figura 33.

En la vista se presenta la imagen más grande del vehículo seleccionado, la capacidad de personas, el tipo de transmisión, el número de puertas, si cuenta con aire acondicionado, la capacidad de maletas, la ciudad, las fechas ingresadas, el total por la reserva y un botón para continuar.

Figura 33. Detalle del vehículo. Detalle de los datos del vehículo seleccionado para continuar con el proceso.

3.3.8 Resultados del Sprint 4

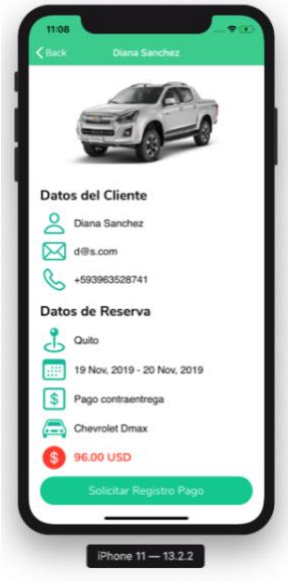
TIT019 - Validación de fechas de búsqueda

Para seleccionar una fecha aparece un calendario, como se observa en la figura 34.

En vista de que los vehículos puedan darse de baja durante un periodo de tiempo cercano, se solicita que las fechas para realizar una reserva no sean anteriores a la actual, ni mayores a 3 meses a partir de la fecha actual, es decir, si la fecha actual es 1 de enero, se podrá realizar una reserva con fecha de inicio hasta antes del 1 de marzo.

Figura 34. Selección de fecha. Interfaz de calendario para seleccionar una fecha.

TIT016 - Solicitar registro de pago de reserva



Como usuario cliente se puede solicitar al desarrollador o dueño de la aplicación el registro de pago cuando el cliente ha pagado por la reserva.

Para realizar esta acción se debe ingresar en los vehículos del propietario, seleccionar el vehículo, seleccionar a la persona que ha hecho la reserva, y presionar el botón para solicitar el registro de pago, como se ve en la figura 35.

Es importante tener en cuenta que solo se visualiza el botón con reservas que aún no han sido pagadas.

Figura 35. Solicitar registro de pago. Interfaz para solicitar el registro de pago de una reserva que aún no ha sido pagada.

TIT017 - Reporte de dinero recaudado

Como usuario propietario se puede obtener un reporte del dinero recaudado a nivel general sobre los vehículos que dispone, como se observa en la figura 36.

Las reservas que han sido pagadas aparecen en ganancias recaudadas, las reservas que aún faltan de pagar aparecen en ganancias por recaudar.

Figura 36. Reporte de dinero. Interfaz para visualizar el reporte de ganancias por la renta de vehículos para el propietario.

TIT020 - Borrar reservas no pagadas

Todos los días de forma automática tienen que revisarse las reservas y comprobar si dentro del rango de los 3 días posteriores, incluyendo al actual hay alguna reserva sin pagar, si es así debe eliminarse, dejando libre al vehículo.

La función se llama *bookingscleanup* y se visualiza su creación en la figura 37.

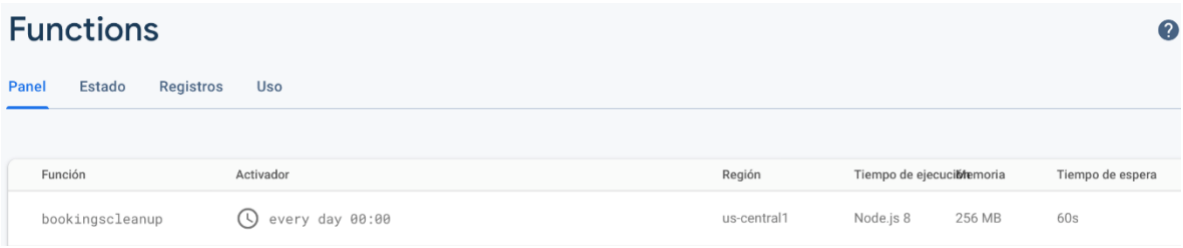


Figura 37. Función *bookingscleanup*. Función de eliminar reservas automáticamente al no cumplir con los pagos en las fechas establecidas.

4. Capítulo IV: Resultados

4.1 Introducción

En presente capítulo se visualizan los resultados obtenidos tanto en la aplicación móvil, como en el desarrollo de la aplicación a nivel de arquitectura y en la base de datos en Firestore.

Primero se visualizará la aplicación en diferentes dispositivos, el entorno para los usuarios cliente y el entorno para los usuarios propietarios. Posteriormente se visualizará como quedó la distribución de paquetes con el patrón MVC para la construcción de la aplicación móvil. Finalmente se visualizará las colecciones que se crearon en Firestore para el almacenamiento de datos.

4.2 Entorno aplicación móvil

En dependencia del usuario que ha iniciado sesión se tienen ciertas características disponibles, a continuación, se presentan los diferentes ambientes para los dos tipos de usuario: cliente (figura 38) y propietario (figura 39).

El principal cambio que se da es en el menú lateral, un cliente no tiene acceso a los vehículos ingresados como propietario ni al balance general de ganancias.

4.2.1 Entorno aplicación móvil cliente

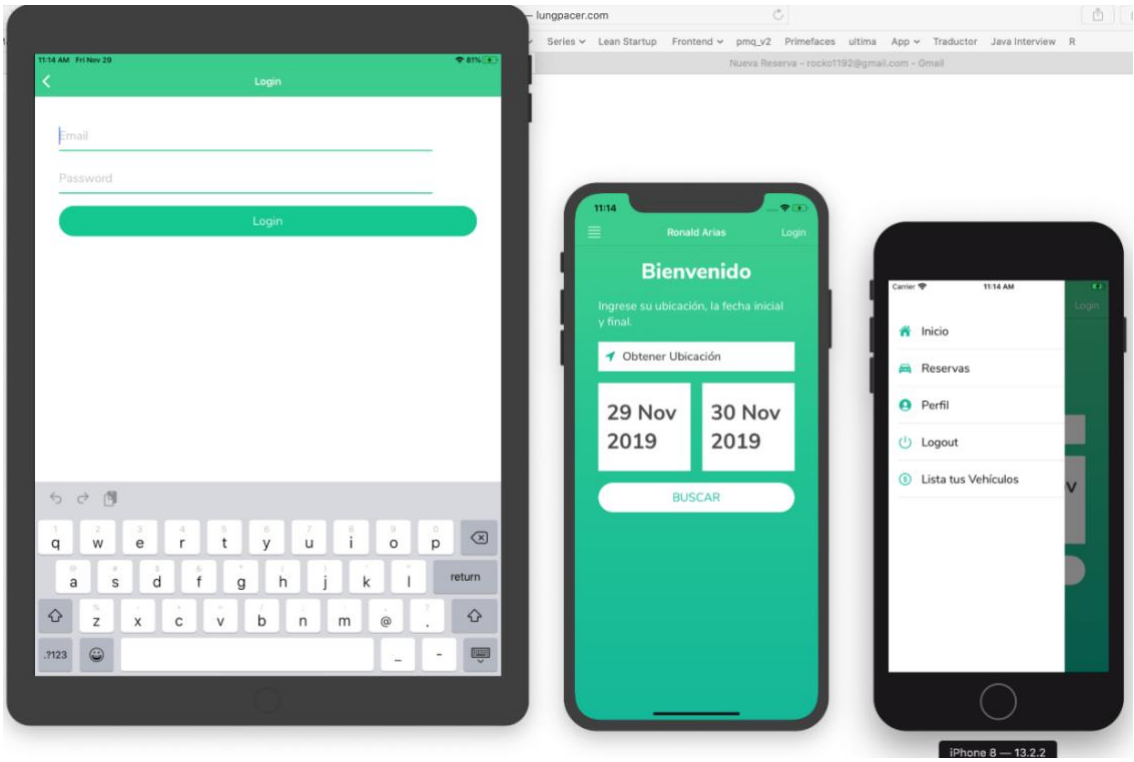


Figura 38. Pantallas cliente. Se visualizan algunas de las pantallas a las que el usuario cliente tiene acceso.

4.2.2 Entorno aplicación móvil propietario

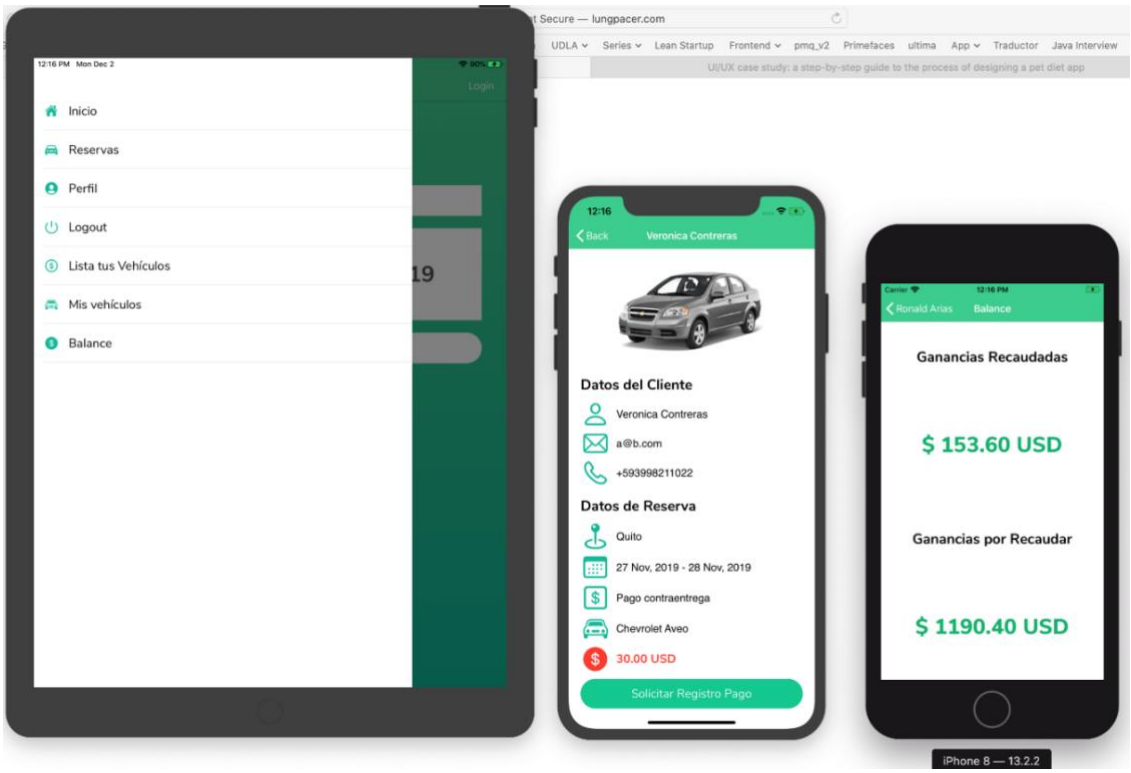


Figura 39. Pantallas propietario. Se visualizan algunas de las pantallas a las que el usuario propietario tiene acceso.

4.3 Arquitectura interna de la aplicación (MVC)

Para el desarrollo de la aplicación se ha utilizado Xcode como Entorno de Desarrollo Integrado. Se ha distribuido las clases, estructuras y storyboards en diferentes paquetes, de tal forma que se cumpla con el patrón Modelo – Vista – Controlador, como se ve en la figura 40.

En el modelo (figura 41) se tienen todas las estructuras acordes a las colecciones creadas en Firestore, en los controladores (figura 44) todas las clases que manejan la lógica y en la vista (figura 42) los storyboards con las interfaces de usuario (figura 43).

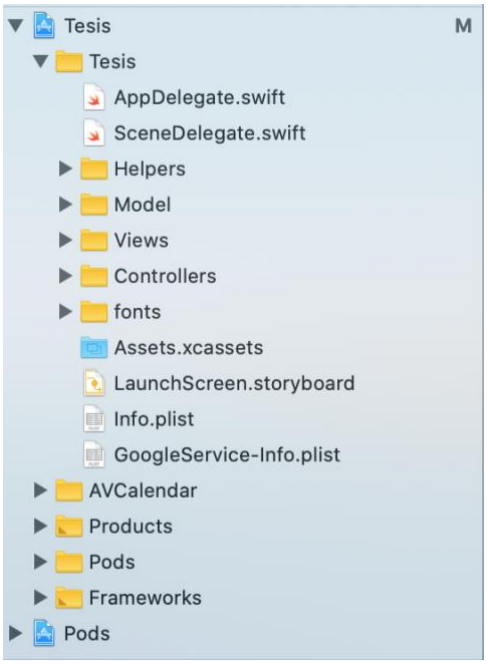


Figura 40. Arquitectura interna de la aplicación. Se visualizan los paquetes creados para cumplir con el patrón MVC.

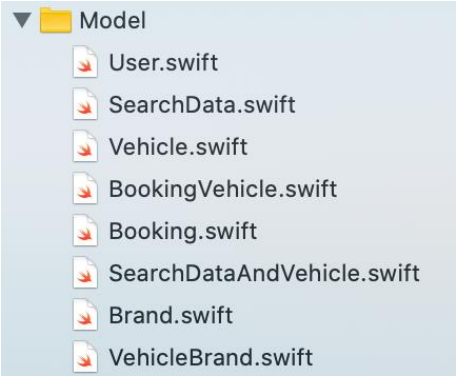


Figura 41. Paquete Model. Se visualizan las estructuras creadas dentro del modelo, acorde a las colecciones en la base de datos de Firestore.

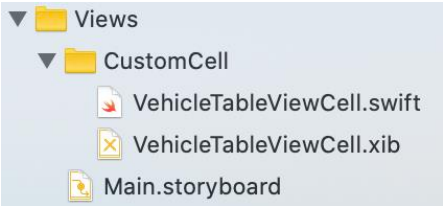


Figura 42. Paquete Views. Se visualizan las clases de interfaz de usuario creadas dentro de vista.

Al seleccionar el archivo *Main.storyboard* se tiene acceso a las interfaces de usuario y la distribución ordenada de las pantallas cuando se interactúa con ellas.

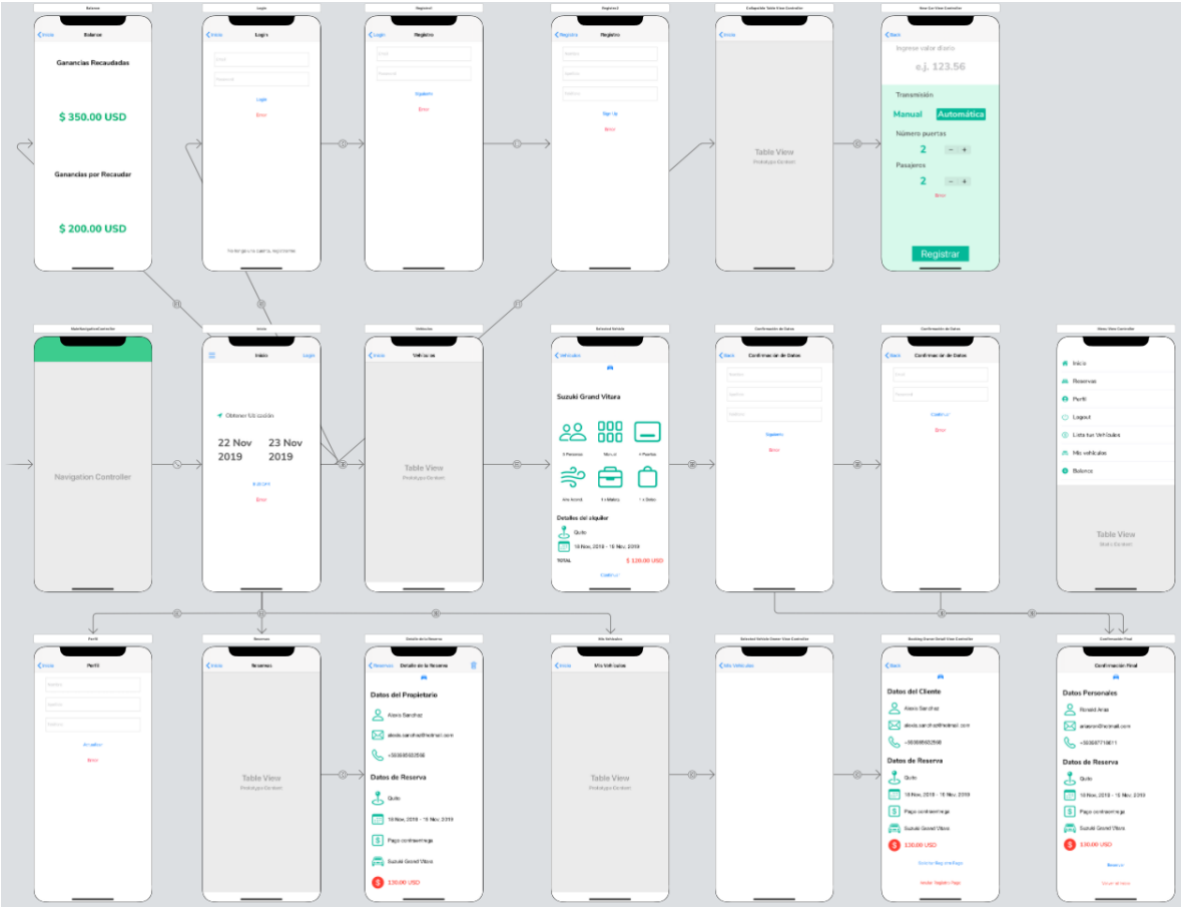


Figura 43. Storyboard. Se visualizan las interfaces de usuario creadas y el orden que siguen en la interacción del usuario.

Finalmente se tiene el paquete de *Controllers* en donde están las clases controladoras con la lógica de negocio.

Figura 44. Paquete Controllers. Se visualizan las clases controladoras que manejan la lógica de negocio.

4.4 Colecciones en Firestore

Finalmente, en Firestore se obtuvieron 6 colecciones con las cuales se desarrolló la aplicación y se almacenan los datos.

Bookings: colección para almacenar los datos de las reservas, en la figura 45 se visualizan los documentos con sus respectivos atributos.

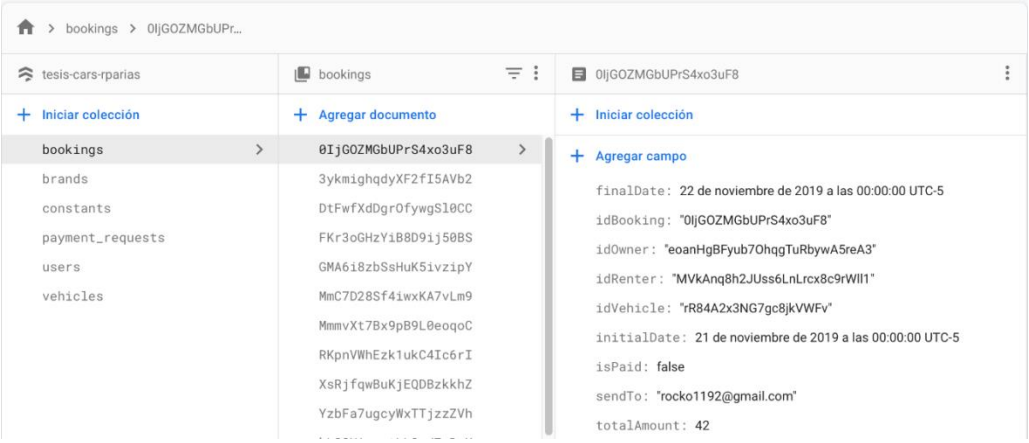


Figura 45. Colección Bookings. Documentos en la colección.

Brands: colección para almacenar los datos de las marcas y modelos de vehículos, en la figura 46 se visualizan los documentos con sus respectivos atributos.

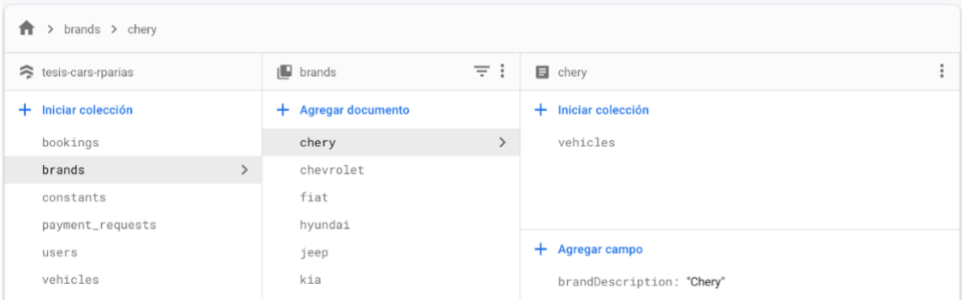


Figura 46. Colección Brands. Documentos en la colección.

Constants: colección para almacenar parámetros generales como información del desarrollador o porcentajes de ganancia en caso de cambiarlos, en la figura 47 se visualizan los documentos con sus respectivos atributos.

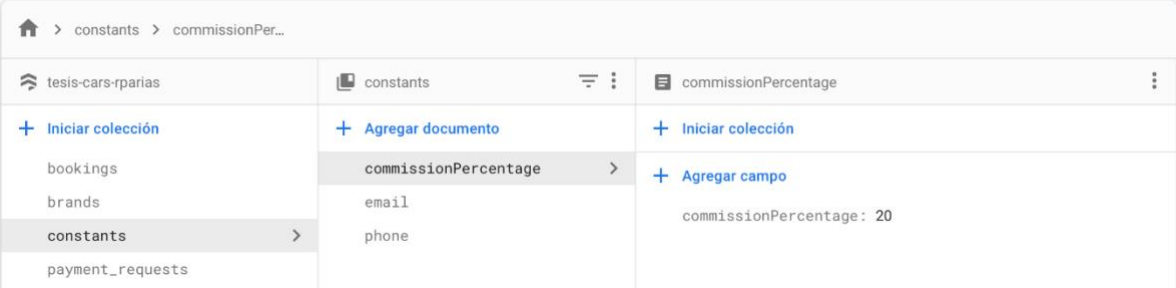


Figura 47. Colección Constants. Documentos en la colección.

Payment Requests: colección para almacenar las solicitudes de pago por parte de los propietarios hacia el dueño de la aplicación, en la figura 48 se visualizan los documentos con sus respectivos atributos.

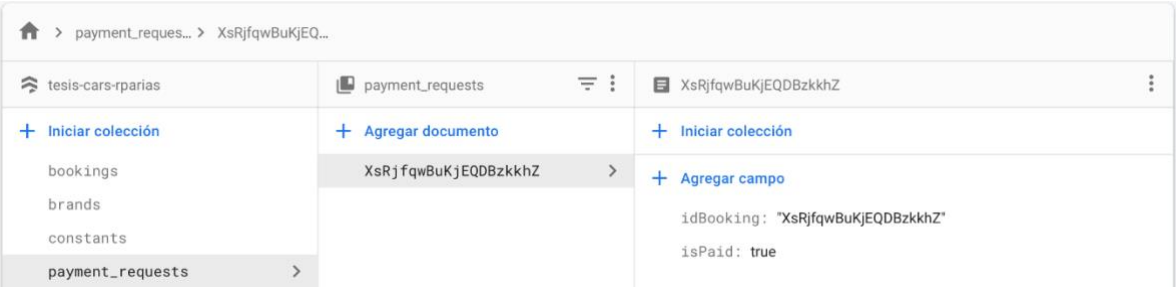


Figura 48. Colección Payment Requests. Documentos en la colección.

Users: colección para almacenar los datos de los usuarios, en la figura 49 se visualizan los documentos en la colección con sus respectivos atributos.

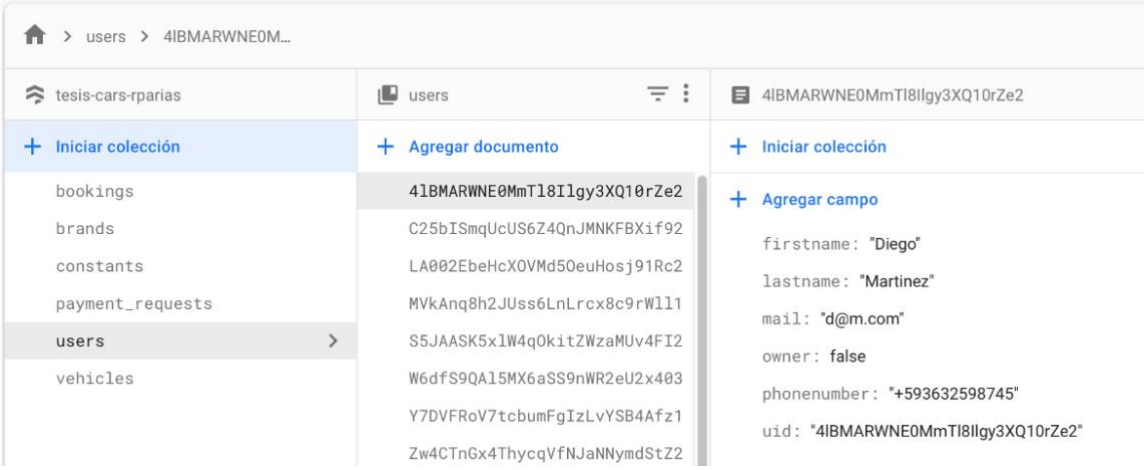


Figura 49. Colección Users. Documentos en la colección.

Vehicles: colección para almacenar los datos de los vehículos ingresados, en la figura 50 se visualizan los documentos con sus respectivos atributos

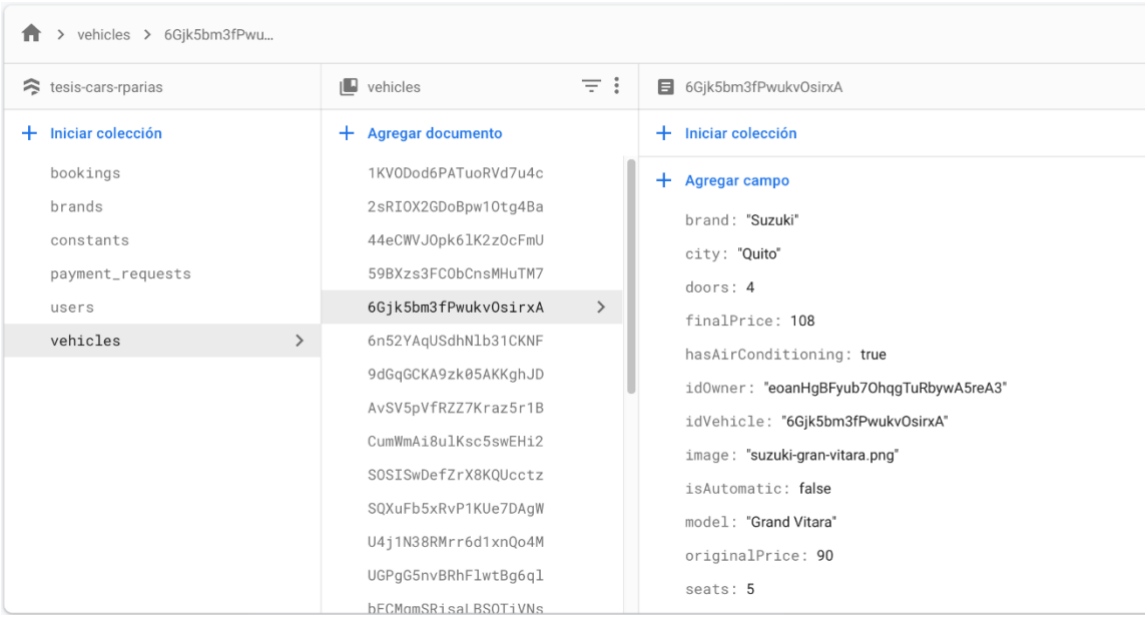


Figura 50. Colección Vehicles. Documentos en la colección.

4.5 Caso de Prueba

Para cumplir con el objetivo principal de la aplicación que es la renta de vehículos, se realizó la validación y verificación del cumplimiento de los criterios de aceptación en cada historia de usuario y, a continuación, en la figura 51 se muestra cual es el proceso para rentar un vehículo, suponiendo que el usuario ya ha ingresado en la aplicación con su usuario y contraseña.

Tabla 28

Caso de prueba

Nombre del caso: Renta vehículo		
Descripción del caso de prueba: Se desea rentar un vehículo desde la pantalla de inicio hasta la confirmación final.		
Pasos y condiciones de ejecución:		
Paso	**Resultado esperado**	**Resultado obtenido**
1) Se ingresa la ubicación, la fecha de inicio y la de fin en la pantalla principal.	Ubicación, fecha de inicio y fecha de fin ingresadas, al pulsar sobre buscar no debe aparecer ningún mensaje de error, posterior debe aparecer la lista de vehículos disponibles para las fechas ingresadas.	OK.
2) Se selecciona un vehículo de la lista.	Aparece una lista con los vehículos disponibles, ordenadas por precio de menor a mayor, cada vehículo debe mostrar su nombre, precio diario y una imagen correspondiente.	OK

	Se hace clic sobre un vehículo y debe continuar a la pantalla de sus detalles.	
3) Se verifica la información del vehículo seleccionado.	Se despliega la información detallada del vehículo seleccionado, con el número de pasajeros permitidos, transmisión, puertas, adicional también la ciudad seleccionada y el precio por día.	OK
4) Se comprueban los datos personales del usuario.	En la pantalla aparecen tres campos, con el nombre, apellido y teléfono respectivamente, el usuario puede editar sus datos y actualizarlos, todos los campos deben estar llenos.	OK
5) Se verifica los datos de la reserva.	Se visualiza un resumen de la reserva, que contempla una imagen del vehículo en la parte superior, debajo los datos personales del cliente, la ciudad, las fechas seleccionadas, la forma de pago y un botón para confirmar la reserva y otro para volver al inicio en caso de no querer reservar ese vehículo.	OK
6) Se realiza una reserva.	Al pulsar sobre el botón reservar, se visualiza un	OK

	mensaje de confirmación que indica que la reserva se ha generado con éxito.

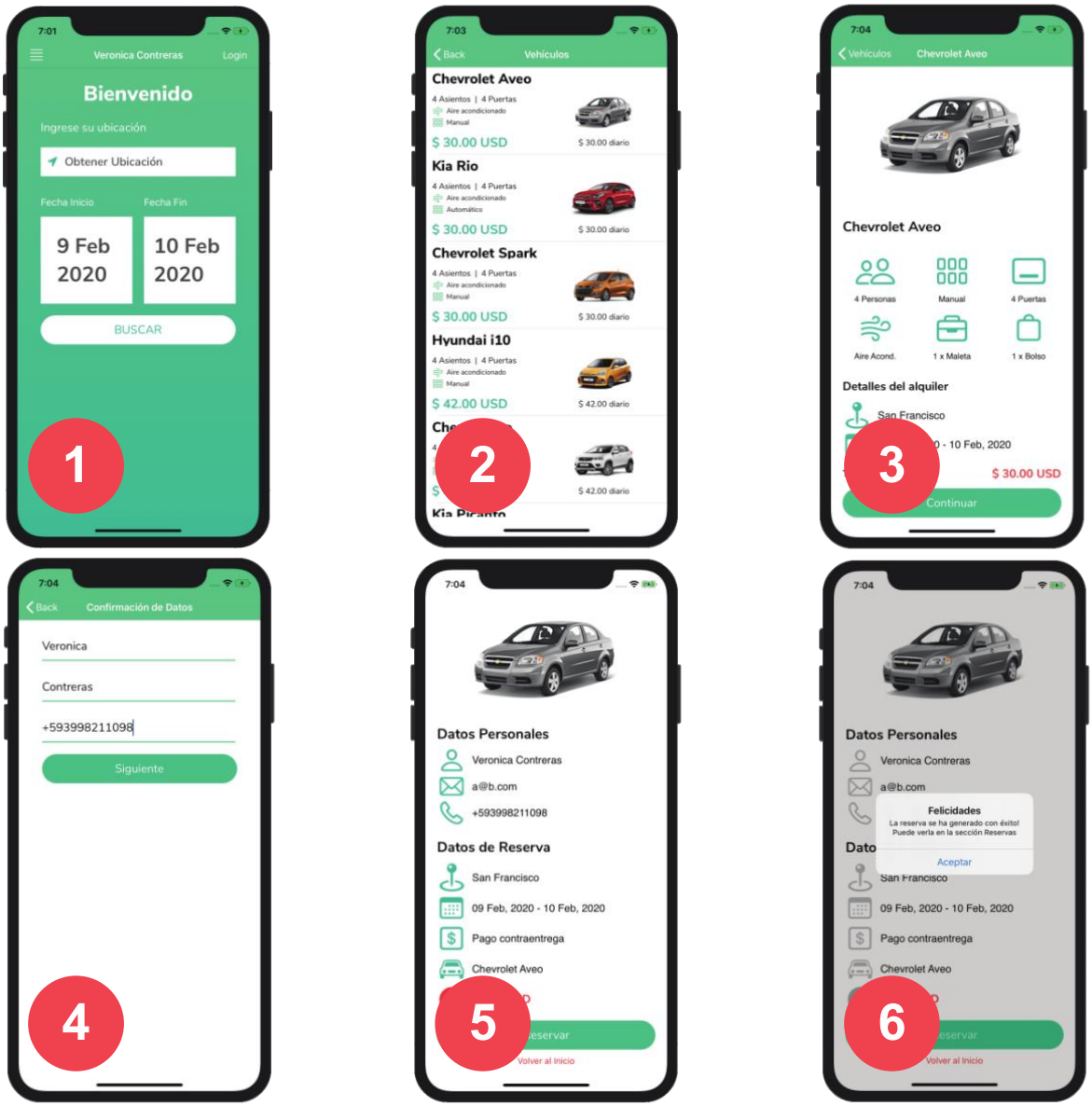


Figura 51. Caso de Prueba. Paso a paso para reservar un vehículo

5. Conclusiones y Recomendaciones

5.1 Conclusiones

Se definió una arquitectura de software escalable y reconocida que se adapte a las exigencias y requisitos de la aplicación, mediante el patrón MVC (Modelo – Vista – Controlador).

Se diseñó e implementó un modelo de datos en la nube de Google Cloud Firebase que se adapte a las necesidades de crecimiento de la aplicación para que soporte todas las transacciones generadas por la aplicación móvil y el crecimiento de usuarios. Se ha utilizado el plan Spark que soporta hasta 1GB de almacenamiento, que es el gratuito, en el caso de que se necesite de más recursos se recomienda migrar a un plan pagado.

Se aplicó el patrón de diseño Adapter, Facade y State. Con el uso de estos patrones de diseño de software se aportó al rendimiento y mantenimiento de la aplicación, además del uso de buenas prácticas de programación como *clean code,* comentarios, nombres en inglés y características de la programación orientada a objetos.

Se diseñó interfaces de usuario sencillas y amigables con el usuario final para ofrecer una buena experiencia de usabilidad en la aplicación.

Mediante el cumplimiento de los criterios de aceptación se comprobó que las funcionalidades de la aplicación móvil se efectúen a lo solicitado. Se ha obtenido un producto que cumple con cada una de las historias de usuario propuestas en un inicio y que a futuro podrían ser ampliadas a nuevas funcionalidades.

5.2 Recomendaciones

Se recomienda utilizar un patrón de arquitectura de software conocido como MVC. Al tener un patrón de diseño Modelo – Vista – Controlador es mucho más fácil de entender como está construida la aplicación, y en el caso de que un desarrollador externo quiera colaborar en este proyecto no será difícil involucrarse y entender cómo realizar cambios en las interfaces de usuario, controladores o en el modelo de datos.

Se recomienda el uso de frameworks de desarrollo de software, ya que estos han sido previamente probados, con resultados de acuerdo con las necesidades y con documentación acerca de cómo aplicarlos. Para otros desarrolladores será más sencillo entender y el mantenimiento será más simple de realizar.

Gracias al uso de la plataforma de Google se tiene una aplicación que es fácilmente escalable, ofrece todas las funcionalidades de servidor, cuenta con una base de datos no relacional y se realizan consultas con mayor optimización y acceso a la información, sin importar que la cantidad de usuarios crezca. Se recomienda leer la documentación oficial de Google para el manejo de la API de Google Firebase.

En el caso de querer lanzar la aplicación al mercado del App Store se recomienda utilizar un plan pagado en Google Firebase, de esa forma se asegura que el crecimiento y transaccionalidad no tengan inconvenientes, el plan gratuito funciona bien para escenarios de corto alcance. También es importante tener en cuenta que se debe pagar por la licencia de desarrollador de Apple, de lo contrario la aplicación no podrá ser puesta en el mercado.

Se recomienda siempre implementar un diseño de interfaces amigables con el usuario, no sirve de nada tener un software que sea difícil de usar o que no se use.

Referencias

Alonso, A., Artime, I., Rodríguez, M. y Baniello, R. (2011). Dispositivos móviles. EPSIG Ing. Telecomunicación Universidad de Oviedo.

Apple. (2019). *App Store*. Recuperado el 10 de noviembre de 2019 de https://www.apple.com/ios/app-store/

Apple. (2019). *SF Symbols*. Recuperado el 11 de diciembre de 2019 de https://developer.apple.com/design/human-interface-guidelines/sf-symbols/overview/

Apple. (2019). *Xcode 11*. Recuperado el 15 noviembre de 2019 de https://developer.apple.com/xcode/

Ashraf, A., Thongpapanl, N., Menguc, B. y Northey, G. (2017). *The role of m-commerce readiness in emerging and developed markets*. Journal of International Marketing, *25*(2), 25-51.

Báez, M., Borrego, Á., Cordero, J., Cruz, L., González, M., Hernández, F. y Torralbo, P. (2012). Introducción a Android. EME Madrid, España, 121.

Canós, J. y Letelier, M. (2012). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia.

Correa, M. (2014). Composición del sistema operativo móvil iOS de Apple y el hardware y software que lo utilizan. Universidad Tecnológica del Perú.

Delía, L., Galdamez, N., Thomas, P. y Pesado, P. (2013). Un análisis experimental de tipo de aplicaciones para dispositivos móviles. Congreso Argentino de Ciencias de la Computación (CACIC) (Vol. 18).

Easy Tour International Ltd. (2019). *EasyRentCars – Rental cars (2.5.4)*. Recuperado el 12 de diciembre de 2019 de https://apps.apple.com/es/app/easyrentcars-alquiler-coches/id1227605924

Enterprise Holdings, Inc. (2019). *Enterprise Rent-A-Car (3.3)*. Recuperado el 12 de diciembre de 2019 de https://apps.apple.com/es/app/enterprise-rent-a-car/id1020641417

Google Fonts. (2019). *Nunito Sans*. Recuperado el 11 de diciembre de 2019 de https://fonts.google.com/specimen/Nunito+Sans

Google. (2019). *Firebase*. Recuperado el 11 de diciembre de 2019 de https://firebase.google.com/

Kay, A. (2000). Dynabooks: *Past, present, and future*. The University of Chicago Press Journals.

Lara, O. (2011). Influencia del color en las preferencias de los consumidores. Revista Observatorio Calasanz, *2*(4), 228-246.

Maicas, J. y Mata, P. (s.f.). ¿Nuevo escenario competitivo en la industria de la telefonía móvil?: Compitiendo a través del iPhone. UCJC Business and Society Review, 3(31).

Maida, E. y Pacienzia, J. (2015). Metodologías de desarrollo de software. Universidad Católica Argentina.

Mathias, M. y Gallagher, J. (2016). *Swift Programming: The Big Nerd Ranch Guide*. Pearson Technology Group.

McNaughton, D. y Light, J. (2013). *The iPad and mobile technology revolution: Benefits and challenges for individuals who require augmentative and alternative communication*.

Merchant, B. (2017). *The one device: The secret history of the iPhone*. Hachette UK.

Naty, L. (2004). Sistemas operativos. Instituto Tecnológico Superior de Acayucan.

Rentcars Ltda. (2019). *Rentcars.com (2.2.3)*. Recuperado el 12 de diciembre de 2019 de https://apps.apple.com/es/app/rentcars-com-alquiler-de-auto/id918167519

Sutherland, J. y Schwaber, K. (2017). La guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego. Scrum. 268.

TravelJigsaw Ltd. (2019). *Rentalcars.com (3.56.0)*. Recuperado el 12 de diciembre de 2019 de https://apps.apple.com/es/app/rentalcars-com-alquiler-coches/id570156001

Turban, E., Lee, J., King, D., Liang, T. y Turban, D. (2009). *Electronic commerce 2010*. Prentice Hall Press.

Xcube Labs. (2011). *Timeline: Mobile Operating Systems*. Recuperado el 2 de febrero de 2020 de

https://visual.ly/community/infographic/computers/timeline-mobile-operating-systems