



Universidad Católica Boliviana "San Pablo"

Facultad de Ingeniería

Departamento de Ingeniería de Sistemas

La Paz - Bolivia

PROYECTO DE GRADO PRESENTADO AL DEPARTAMENTO DE
INGENIERÍA DE SISTEMAS PARA LA OBTENCIÓN DEL GRADO
ACADÉMICO DE LICENCIATURA EN INGENIERÍA DE SISTEMAS

"SISTEMA DE INFORMACIÓN ORIENTADO A LA
RESERVA Y PAGO DE PARQUEOS PÚBLICOS EN LA
CIUDAD DE LA PAZ CON TECNOLOGÍA MÓVIL,
COMPUTACIÓN EN LA NUBE Y
GEOREFERENCIAMIENTO"

Realizado por:

José Renán Estenssoro Mercado

Tutor: Ing. Ernesto Omar Campohermoso Alcón

Relator: Dr. Miguel Ángel Villarroel Salgueiro

AÑO 2021

Contenido

1.	INTRODUCCIÓN	1
2.	MARCO REFERENCIAL	1
2.1	ANTECEDENTES.....	1
2.2	DESCRIPCIÓN DEL OBJETO DEL ESTUDIO	2
2.3	IDENTIFICACIÓN Y FORMULACIÓN DEL PROBLEMA DE INVESTIGACIÓN	3
2.3.1	PLANTEAMIENTO DEL PROBLEMA.....	3
2.3.2	FORMULACIÓN DEL PROBLEMA.....	4
2.4	OBJETIVOS	5
2.4.1	GENERAL.....	5
2.4.2	ESPECÍFICOS.....	5
2.5	JUSTIFICACIONES	5
2.5.1	JUSTIFICACIÓN TEÓRICA	5
2.5.2	JUSTIFICACIÓN PRÁCTICA.....	6
2.5.3	JUSTIFICACIÓN METODOLÓGICA.....	6
2.5.4	JUSTIFICACIÓN SOCIAL.....	6
2.6	LÍMITES Y ALCANCES	7
2.6.1	ALCANCES	7
2.6.2	LIMITES.....	7
2.7	ASPECTOS METODOLÓGICOS DE INVESTIGACIÓN	7
2.7.1	TIPO DE ESTUDIO	7
2.7.2	MÉTODO DE INVESTIGACIÓN	8
2.8	HERRAMIENTAS Y MODELOS	8
2.8.1	MODELOS	8
2.8.2	HERRAMIENTAS	8
2.9	RECOPILACIÓN METÓDICA DE DATOS	9
2.9.1	FUENTES PRIMARIAS.....	10
2.9.2	FUENTES SECUNDARIAS	10
2.10	ANÁLISIS PRELIMINAR DEL PROYECTO	11
2.11	ANÁLISIS SMART.....	13
2.12	ANÁLISIS PART	14
3	MARCO TEÓRICO	15
3.1	APLICACIÓN MÓVIL.....	15

3.2	ANDROID	16
3.3	SISTEMA DE COBRO KHIPU	19
3.4	FIREBASE	19
3.4.1	CARACTERÍSTICAS DE FIREBASE.....	19
3.4.2	SERVICIOS DE FIREBASE	20
3.5	CÓDIGOS QR	24
3.6	SCRUM	24
4	ANÁLISIS	26
4.1	USUARIOS.....	26
4.2	PLANIFICACIÓN.....	27
4.3	HISTORIAS DE USUARIOS	29
4.3.1	PRIMER SPRINT.....	29
4.3.1.1	REGISTRO DE USUARIOS, INICIO DE SESIÓN, VALIDACIÓN DE CERTIFICADOS	29
4.3.1.2	MENÚ PRINCIPAL.....	32
4.3.1.3	REGISTRO DE VEHÍCULOS.....	33
4.3.1.4	BÚSQUEDA, RESERVA Y PAGO ESTACIONAMIENTO DISPONIBLE	35
4.3.2	SEGUNDO SPRINT	39
4.3.2.1	INICIO Y FIN DE HORA	39
4.3.2.2	CARGA DE CRÉDITO	41
4.3.2.3	HISTORIAL DE CARGA Y FACTURACIÓN	44
4.3.2.4	LISTA DE RESERVAS Y GASTOS	45
4.3.3	TERCER SPRINT	46
4.3.3.1	PERFIL DE USUARIO, CAMBIAR CONTRASEÑA Y CERRAR SESIÓN.....	46
4.3.3.2	REPORTAR PROBLEMA.....	48
4.3.3.3	NOTIFICACIONES PUSH	49
4.3.4	CUARTO SPRINT	51
4.3.4.1	MENÚ PRINCIPAL (CONTROLADOR)	51
4.3.4.2	VERIFICAR RESERVA POR QR Y PLACAS (CONTROLADOR).....	52
4.3.4.3	RESERVA MANUAL (CONTROLADOR)	55
5	DISEÑO	56
5.1	CASOS DE USO	56
5.1.1	CASO DE USO 1 REGISTRO, LOGIN, Y AUTENTICACIÓN DE USUARIOS.	56
5.1.1.1	DESCRIPCIÓN DEL CASO DE USO	56
5.1.1.2	ACTORES.....	56
5.1.1.3	CASO DE USO	57
5.1.1.4	DIAGRAMA CASO DE USO	59

5.1.1.5	FLUJO DEL CASO DE USO	59
5.1.2	CASO DE USO 2: REGISTRO DE VEHÍCULOS	61
5.1.2.1	DESCRIPCIÓN DEL CASO DE USO	61
5.1.2.2	ACTORES.....	61
5.1.2.3	CASO DE USO.....	61
5.1.2.4	DIAGRAMA DE CASO DE USO.....	62
5.1.2.5	FLUJO DEL CASO DE USO.....	62
5.1.3	BÚSQUEDA, RESERVA Y PAGO DE ESTACIONAMIENTO DISPONIBLE	63
5.1.3.1	DESCRIPCIÓN DEL CASO DE USO	63
5.1.3.2	ACTORES.....	63
5.1.3.3	CASO DE USO.....	63
5.1.3.4	DIAGRAMA DE CASO DE USO.....	64
5.1.3.5	FLUJO DEL CASO DE USO.....	65
5.1.4	INICIO Y FIN DE HORA	66
5.1.4.1	DESCRIPCIÓN DEL CASO DE USO	66
5.1.4.2	ACTORES.....	66
5.1.4.3	CASO DE USO.....	66
5.1.4.4	DIAGRAMA DE CASO DE USO.....	67
5.1.4.5	FLUJO DEL CASO DE USO.....	67
5.1.5	CARGA DE CRÉDITO.....	68
5.1.5.1	DESCRIPCIÓN DEL CASO DE USO	68
5.1.5.2	ACTORES.....	68
5.1.5.3	CASO DE USO.....	69
5.1.5.4	DIAGRAMA DE CASO DE USO.....	69
5.1.5.5	FLUJO DEL CASO DE USO.....	70
5.1.6	FACTURACIÓN	71
5.1.6.1	DESCRIPCIÓN DEL CASO DE USO	71
5.1.6.2	ACTORES.....	71
5.1.6.3	CASO DE USO.....	71
5.1.6.4	DIAGRAMA DE CASO DE USO.....	72
5.1.6.5	FLUJO DEL CASO DE USO.....	72
5.2	DIAGRAMA DE COMUNICACIÓN	73
5.3	DIAGRAMAS DE SECUENCIA.....	74
6	ESTRUCTURA DEL PROTOTIPO.....	82
6.1	BASE DE DATOS EN TIEMPO REAL	82
6.1.1	CONFIGURACIÓN DE LA BASE DE DATOS AL PROYECTO.....	82

6.1.2	ESTRUCTURA DE UNA BASE DE DATOS DE FIREBASE	85
6.1.3	DISEÑO DE LA BASE DE DATOS	86
6.1.4	PRUEBA DE CARGA A LA BASE DE DATOS	95
6.2	ARQUITECTURA DEL PROTOTIPO	99
6.2.1	MODELO VISTA CONTROLADOR.....	99
6.2.2	DIAGRAMA DE C4	100
6.3	ESTRUCTURA DE LA APLICACIÓN.....	102
6.3.1	DISEÑO DE INTERFACES GRAFICAS	102
6.3.1.1	FORMULARIOS.....	106
6.3.1.2	BOTONES.....	106
6.3.1.3	MENÚ PRINCIPAL.....	107
6.3.1.4	MAPA DE GOOGLE	107
6.3.2	FUNCIONES APLICACIÓN	109
6.3.2.1	DIAGRAMA DE CLASES	109
6.3.2.2	AUTENTICACIÓN DE USUARIOS.....	112
6.3.2.3	BASE DE DATOS.	114
6.3.2.4	CÓDIGO DEL SISTEMA	117
6.3.3	ARCHIVO BUILD.GRADLE.....	127
6.3.4	MANIFIESTO DE LA APLICACIÓN.....	128
6.4	ESTADÍSTICAS DE LA APLICACIÓN.....	130
7	CONCLUSIONES Y RECOMENDACIONES	133
7.1	CONCLUSIONES.....	133
7.2	RECOMENDACIONES	136
8	REFERENCIAS	138
8.1	BIBLIOGRAFÍA	138

Ilustración 1 Diagrama Ishikawa.....	Error! Bookmark not defined.
Ilustración 2 Arquitectura Android	Error! Bookmark not defined.
Ilustración 3 planificación 1	Error! Bookmark not defined.
Ilustración 4 planificación 2	Error! Bookmark not defined.
Ilustración 5 Registro de usuarios	Error! Bookmark not defined.
Ilustración 6 Iniciar sesión	Error! Bookmark not defined.
Ilustración 7 Menú principal.....	33

Ilustración 8 Registro de Vehículos	35
Ilustración 9 Mapa de estacionamientos	37
Ilustración 10 Reserva de parqueo.....	38
Ilustración 11 Confirmar y pagar	38
Ilustración 12 Inicio de hora.....	40
Ilustración 13 Fin de hora	41
Ilustración 14 Sistema de carga	43
Ilustración 15 sistema de carga Khipu.....	43
Ilustración 16 lista de recargas	44
Ilustración 17 Factura	44
Ilustración 18 Lista de reservas.....	46
Ilustración 19 Mi perfil.....	47
Ilustración 20 Reportar un problema	49
Ilustración 21 notificaciones Push	50
Ilustración 22 Menú Principal (controlador).....	52
Ilustración 23 Cámara QR	53
Ilustración 24 Confirmación de Reserva (Controlador)	54
Ilustración 25 Reserva por Placa (Controlador)	54
Ilustración 26 reserva manual	56
Ilustración 27 Agregar Base de datos	82
Ilustración 28 Regla de escritura y lectura falso.....	82
Ilustración 29 Solo usuarios autenticados pueden leer y escribir.....	83
Ilustración 30 Dueños de contenido	84
Ilustración 31 Todos Pueden leer, solo dueños de contenido escriben	84
Ilustración 32 Estructura Base de Datos	85
Ilustración 33 Anidación de datos.....	86
Ilustración 34 estructura de la base de datos.....	87
Ilustración 35 componentes usuarios	88
Ilustración 36 componentes vehículos	89
Ilustración 37 componentes recargas.....	90
Ilustración 38 componentes reserva.....	91
Ilustración 39 Componente Código estacionamiento	92
Ilustración 40 componente estacionamiento	93
Ilustración 41 componente disponibilidad.....	94

Ilustración 42 componente precio.....	94
Ilustración 43 Prueba de 100.....	96
Ilustración 44 http request	96
Ilustración 45 resultados http 1.....	97
Ilustración 46 resultados http 2.....	97
Ilustración 47 Prueba de 1000.....	98
Ilustración 48 Resultados negativos.....	99
Ilustración 49 Modelo Vista Controlador.....	100
Ilustración 50 Diagrama c4 nivel 1	100
Ilustración 51 Diagrama c4 nivel 2	101
Ilustración 52 Diagrama c4 nivel 3	102
Ilustración 53 Formularios	106
Ilustración 54 Botones	107
Ilustración 55 CardView.....	107
Ilustración 56 Mapa de Google.....	108
Ilustración 57 Pin	109
Ilustración 58 Diagrama de clases	111
Ilustración 59 Declarar Firebase.....	112
Ilustración 60 FirebaseAuth instance	112
Ilustración 61 Crear usuario	113
Ilustración 62 Iniciar Sesión Firebase.....	113
Ilustración 63 Verificar sesión.....	114
Ilustración 64 Reference.....	114
Ilustración 65 FirebaseDatabase	114
Ilustración 66 Lectura de datos Firebase	115
Ilustración 67 Map Firebase	116
Ilustración 68 Escribir datos Firebase	116
Ilustración 69 Obtener ubicación	117
Ilustración 70 setmarker	118
Ilustración 71 crear QR.....	119
Ilustración 72 ZXing	119
Ilustración 73 variable myscan	120
Ilustración 74 código inicial.....	120
Ilustración 75 Handle Result.....	120

Ilustración 76 Creación de instancias	121
Ilustración 77 Activar modo Sandbox	122
Ilustración 78 Configuración Básica	122
Ilustración 79 Obtener resultados.....	Error! Bookmark not defined.
Ilustración 80 Detener servicio	123
Ilustración 81 Imports web.....	124
Ilustración 82 imports	125
Ilustración 83 Firebase module	125
Ilustración 84 lectura datos.....	126
Ilustración 85 ngfor	126
Ilustración 86 Datos HTML	127
Ilustración 87 Android.....	127
Ilustración 88 Dependencias	128
Ilustración 89 Permisos	129
Ilustración 90 Especificaciones	129
Ilustración 91 Activitys	129
Ilustración 92 Meta data 1	130
Ilustración 93 meta data 2	130
Ilustración 94 Usuarios Activos	131
<i>Ilustración 95 interacción diaria</i>	131
Ilustración 96 Dispositivos	133
Caso de uso 1 Inicio de sesión, registro, verificación de credenciales	59
Caso de uso 2 Registro de vehiculos	62
Caso de uso 3 Búsqueda, reserva y pago de estacionamientos	64
Caso de uso 4 Inicio, fin de hora	67
Caso de uso 5 carga de crédito.....	69
Caso de uso 6 Facturación.....	72
Diagrama de secuencias 1 Iniciar sesión	75
Diagrama de secuencias 2 Registro de usuarios	76
Diagrama de secuencias 3 Registro de vehículos	77
Diagrama de secuencias 4 Búsqueda, reserva y pago de estacionamientos	78

Diagrama de secuencias 5 Inicio, fin de Hora	79
Diagrama de secuencias 6 Carga de crédito	80
Diagrama de secuencias 7 Facturación	81

1. Introducción

Con la globalización, los avances en la tecnología (especialmente en las tecnologías de la información y la comunicación que, sustentadas en el internet, están evolucionando de ser sistemas de aplicaciones de escritorio a sistemas más complejos) han modificado absolutamente todas las costumbres, comportamientos e interacciones de los seres humanos en prácticamente todas las sociedades del mundo. El uso de los celulares en los sistemas de información es ahora muy común, debido a que los teléfonos pueden ser fácilmente transportados y estar permanentemente conectados a Internet.

Bolivia tiene actualmente poco más de 11 millones de habitantes. El número de celulares activos también es de alrededor de 11 millones y las conexiones a Internet son poco más de 10 millones. Esto quiere decir que, prácticamente, todos los habitantes del país tienen un celular que está conectado a Internet. (Urgente.bo, 2018)

La implementación de la tecnología en cualquier tipo de empresa es una obligación, actualmente muchas de las empresas que no utilizan aplicaciones móviles en sus negocios están en desventaja (La Razon, 2014). Al mismo tiempo, la mayoría de las empresas están atravesando, en la actualidad, un proceso de transformación digital.

2. Marco referencial

2.1 Antecedentes

“Aventón” es una aplicación diseñada por los estudiantes de la Universidad Católica Boliviana e implementada por la Alcaldía de la ciudad de La Paz que tiene como objetivo lograr que los mismos estudiantes se organicen para que varios lleguen a la universidad en un solo vehículo. Es decir, mediante esta iniciativa, un estudiante que se transporta en su coche a la universidad, en su trayecto irá recogiendo a varios/as compañeros/as logrando con ello reducir el consumo de gasolina (ya no son necesarios varios automóviles sino uno solo) y ahorrar en el pasaje, creando con ello una cultura de solidaridad y compañerismo. Adicionalmente, el dueño del vehículo logra una rebaja en el costo del estacionamiento.

Khipu es una plataforma de pago implementada por Red Enlace que permite realizar pagos por internet con una tarjeta de crédito o débito. Esta plataforma se integra fácilmente a cualquier tipo de sistema de comercios electrónicos. El dinero cobrado con el sistema Khipu se deposita de manera inmediata en una cuenta bancaria asociada a la Red Enlace sin ningún tipo de comisión.

Si bien existen varias iniciativas plausibles como las señaladas, aún no se ha diseñado una aplicación que permita gestionar al Municipio de La Paz los estacionamientos públicos de la ciudad. De momento, esta actividad se realiza mediante un sistema manual de cobros que se realizan con una maquina que genera las facturas de acuerdo a un tarifario (Bs. 5 la hora y Bs. 3 la media hora).

2.2 Descripción del objeto del estudio

El estacionamiento es una de las condiciones que puede mejorar o empeorar la calidad de vida en una ciudad. Conseguir parqueo en determinadas zonas y a ciertas horas puede resultar una tarea sumamente desagradable tanto por el tiempo que se pierde, la gasolina que se consume, el aire que se contamina como por el desgaste emocional en el que se ve envuelto el automovilista. Esto quiere decir que los ciudadanos que se transportan en vehículos, precisan de información sobre si existen estacionamientos disponibles en su destino para acceder a los mismos sin mayores contratiempos.

La demarcación de las áreas de estacionamiento en las calles y la determinación del sistema de cobro y las tarifas han generado una nueva fuente de ingresos para el municipio y, al mismo tiempo, han generado un nuevo problema: La mala calidad de las condiciones de trabajo de quienes se encargan del cobro y el control.

Para solucionar este problema de manera efectiva y a bajo costo, se propone desarrollar un sistema de información con tecnología móvil, cloud computing y mapas que, de manera inmediata, comunique al usuario la disponibilidad de un espacio para estacionamiento en la zona requerida y, al mismo tiempo, lo reserve una vez efectuado el pago.

2.3 Identificación y formulación del problema de investigación

2.3.1 Planteamiento del problema

En septiembre del 2017, la Alcaldía de la ciudad de La Paz aprobó el código de vía C003-0618 que señala que en 19 calles del centro de la ciudad se comenzará a cobrar por el estacionamiento de vehículos. Esta política ha sido ampliada a Obrajes y Calacoto. (Página Siete, 2018) Esta iniciativa pretende incrementar los ingresos del municipio, aunque su implementación es sumamente rústica y, por ello, ineficiente. Según estadísticas del municipio, esta actividad arroja pérdidas mensuales que alcanzan los Bs. 428.500.-, lo que significan bs. 5.142.000.- por año. Este déficit se debe a que la Alcaldía contrata 170 nuevos funcionarios. Cada uno de ellos gana 2.850 bolivianos. El gasto mensual en salarios por este concepto asciende a bs. 484.500 bolivianos; si a esto se le suma los gastos de 45 guardias municipales que ganan 3.200 bolivianos al mes, el costo total en personal es de bs. 628.500 y los ingresos reportados son de 200.000 bolivianos al mes (Agencia de noticias Fides, 2018). Es decir, el déficit mensual es de aproximadamente el 68%.

La dinámica implementada por la Alcaldía es simple. Para cada cuadra, se ha dispuesto un cobrador que debe vigilar y cobrar durante ocho horas continuas el estacionamiento de vehículos. Otro par de guardias municipales son los encargados de inmovilizar los vehículos que, o bien no han pagado, o bien se han excedido en el tiempo y lo que han cancelado ya no es suficiente. Este sistema provoca los siguientes problemas:

- 1.** En muchas ocasiones, estos funcionarios no cuentan con el cambio necesario lo que provoca demoras e incomodidad.
- 2.** Los usuarios no saben si encontrarán un lugar en la zona para estacionar su vehículo.
- 3.** El sistema arroja anualmente pérdidas por \$US 738.793.- dólares americanos.
- 4.** Cuando el usuario se excede del tiempo cancelado los guardias municipales proceden a colocar una tranca al automóvil lo que provoca demoras e incomodidad.
- 5.** Debido a la coyuntura actual, los funcionarios que cobran los estacionamientos pueden ser un foco de contagio o pueden contagiarse con el COVID – 19.

Debido al Sars-Cov-2 estamos obligados tratar de reducir el uso de dinero en efectivo y utilizar métodos de pago electrónicos.

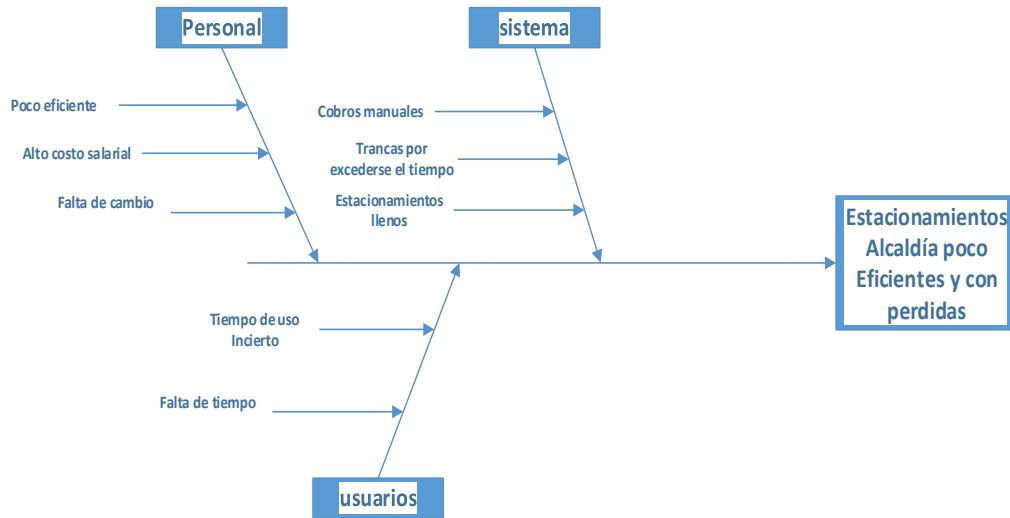


Ilustración 1 Diagrama Ishikawa

En el diagrama anterior se muestran los siguientes componentes:

1. El tener personal humano nos trae el problema de altos costos salariales (se debe promover un trabajo digno en las instituciones del Estado), existen momentos en los que estos no tienen cambio y en algunos casos el personal no está presente en la zona en los momentos necesarios.
2. El sistema implementado no es eficaz ya que casi todo el proceso se lo realiza de manera manual. En caso de exceso de tiempo, se ponen trancas. Para retirarlas se debe esperar al personal de control y ello toma su tiempo. De otro lado, es frecuente que en la búsqueda estacionamientos disponibles se den vueltas por la zona.
3. Los usuarios, por lo general, están apurados y no tienen tiempo para esperar a los funcionarios que deben cobrar y controlar el pago.

2.3.2 Formulación del problema

¿Cómo se puede gestionar los parqueos del municipio con una aplicación móvil, implementada en la nube, con base de datos en tiempo real, para que los usuarios

puedan buscar estacionamiento de forma georreferenciada, realizar reservas y pagar de forma segura?

2.4 Objetivos

2.4.1 General

Desarrollar una aplicación móvil que permita a los usuarios buscar estacionamientos, realizar reservas y pagar por las mismas utilizando instrumentos electrónicos de pago, para simplificar el proceso y coadyuvar a la reducción del uso de efectivo.

2.4.2 Específicos

- Desarrollar una aplicación móvil para que los usuarios puedan ver los estacionamientos disponibles, reservarlos y pagarlos, de acuerdo a lo establecido en el tarifario.
- Desarrollar una aplicación móvil para que los funcionarios de la alcaldía puedan verificar que todos los usuarios estacionados hayan pagado lo justo.
- Integrar las aplicaciones móviles a una base de datos en tiempo real para que los datos en el sistema estén en perfecta sincronía.
- Integrar el sistema a una pasarela de pagos para poder realizarlos sin la necesidad de manejar efectivo
- Diseñar un método de contingencia en caso de la perdida de la conexión a internet

2.5 Justificaciones

2.5.1 Justificación teórica

El desarrollo de la aplicación móvil pondrá en práctica todos los conocimientos aprendidos durante la carrera de ingeniería de sistemas en la Universidad Católica Boliviana. Se pondrá en práctica conceptos en áreas como la programación tanto en “Back-end” como “Front-end”, base de datos, desarrollo, aplicación de nuevas tecnologías como desarrollo de aplicaciones móviles, integración de sistemas, entre

otros. También se pondrán en práctica conceptos teóricos aprendidos como modelado en UML, la aplicación de metodologías de desarrollo como Scrum.

2.5.2 Justificación Práctica

La aplicación móvil permitirá que los conductores de vehículos encuentren rápidamente un estacionamiento y paguen por su uso. Al mismo tiempo, permitirá al municipio controlar el uso de los mismos y mejorar el sistema de cobro, lo que incrementará sustancialmente sus ingresos.

2.5.3 Justificación metodológica

Debido al corto tiempo para realizar el proyecto se debe elegir una metodología de desarrollo ágil. Scrum es una metodología de desarrollo ágil que encaja a la perfección con el desarrollo de este proyecto.

Scrum permite crear objetivos a cumplir en ciertos períodos de tiempo. Además, facilita tener reuniones semanales las cuales se implementarán durante la realización de este proyecto.

2.5.4 Justificación Social

El problema del estacionamiento es cada día más agudo en la ciudad. Circulan más vehículos de lo que la urbe tolera y esto ocasiona grandes embotellamientos, alta contaminación y falta de lugares de estacionamiento. Además, el sistema de cobro por el estacionamiento que ha implementado el municipio es arcaico y sumamente incómodo. El sistema funciona con cobradores desplazados en cada cuadra que deben controlar el pago. Son varios los casos en los que los usuarios no los encuentran y no logran pagar lo que les ocasiona multas. También existe la posibilidad de que los cobradores pierdan el dinero, no cuadren sus rendiciones de cuentas o bien no cuenten con el cambio necesario. A ello se suma las condiciones de trabajo. Los cobradores y los guardias municipales trabajan en condiciones paupérrimas: Varias horas a la intemperie, sin espacio para un descanso y mucho menos acceso a sanitarios.

2.6 Límites y alcances

2.6.1 Alcances

- El sistema realizará cobros online
- El sistema generará códigos QR para gestionar la reserva de estacionamiento
- El sistema tendrá un lector de códigos QR para gestionar las reservas
- El sistema generará una factura en PDF
- El sistema estará conectado a Google Maps
- El sistema mostrará los parqueos disponibles en el mapa
- El sistema tendrá un buscador de parqueo cercano disponible por la ubicación compartida
- El sistema generará informes empresariales para el municipio
- En el sistema se podrá verificar los parqueos con el número de placa
- El sistema pondrá multas a los usuarios que excedan el tiempo reservado y pagado
- El sistema generará una alerta cuando el tiempo reservado se está terminando con notificaciones Push
- El sistema te permitirá ampliar el tiempo del parqueo si este disponible
- El sistema soportara desconexiones de internet en el Back-end

2.6.2 Limites

- El sistema funcionara sólo en la ciudad de La Paz
- Sólo se llegará al diseño de prototipo, no habrá implementación
- El sistema no “Khipu” se implementará únicamente en el Sandbox de prueba proporcionado por la empresa
- La aplicación móvil se ejecutará únicamente en el sistema operativo Android

2.7 Aspectos metodológicos de investigación

2.7.1 Tipo de estudio

Dadas las características de la aplicación a desarrollar, se desarrollará un estudio descriptivo ya que se utilizarán técnicas específicas de recolección de datos, además se entrará a fondo sobre los procesos utilizados por la alcaldía. También se realizará un análisis de la arquitectura. Toda la información recolectada y analizada será medible y concreta.

2.7.2 Método de investigación

El proyecto de investigación se basará en el funcionamiento y los procesos de gestión de los estacionamientos de vehículos públicos que la alcaldía de la ciudad de La Paz implementa. Se desarrollará un nuevo método de pago y se analizarán nuevas formas de mejorar estos procesos de modo que se pueda cumplir con los nuevos protocolos de distanciamiento. Por estos motivos la investigación se realizará con el método inductivo.

2.8 Herramientas y modelos

2.8.1 Modelos

El sistema de información va a ser implementado con la metodología Scrum. Para desarrollar este proyecto se utilizará una metodología ágil y de fácil implementación

Scrum encaja perfectamente en las necesidades del proyecto al ser un marco de trabajo donde se aplican buenas prácticas. Con esta metodología se realizan entregas parciales de los trabajos finales lo que permite ser un marco ideal para proyectos donde los plazos son cortos.

2.8.2 Herramientas

Las herramientas a utilizar son las siguientes:

- Android Studio

Android Studio es IDE (Entorno de desarrollo integrado por sus siglas en inglés) para el desarrollo de aplicaciones con el sistema operativo Android. Este IDE fue desarrollado por la empresa Google.

- Sistema operativo Android

Android versión Jelly Bean (4.1-4.3.1) será la versión mínima del sistema de operativo móvil donde se implementará la aplicación. Google es la empresa propietaria de Android.

- Sistema de cobros Khipu

Khipu es un sistema de cobros por internet implementado por la empresa Red Enlace. Este sistema mediante métodos “Post” y “Gets” permite conectar cualquier aplicación móvil a un sistema de cobro seguro y fácil de usar. Asimismo, realiza depósitos a cualquier cuenta.

- Servidor Firebase

Firebase es un sistema multiplataforma que ofrece varios servicios para aplicaciones tanto móviles como webs. Firebase Server permite conectar de manera simple y fácil aplicaciones Android, IOS y web a sus servicios de base de datos Messaging y Cloud Storage, entre otros.

- Base de datos No SQL de Firebase

La Base de datos en tiempo real de Firebase ofrece un servicio de almacenamiento de datos en tiempo real.

- Lenguaje de programación Java

El lenguaje de programación será Java que es el lenguaje más utilizado para la programación de aplicaciones móviles en Android

- Windows 7

Para ejecutar programas como Android Studio entre otros se utilizará el sistema operativo de Microsoft Windows 7

- Enterprise Architect

Enterprise Architect es un software para el diseño de diagramas UML que permite realizar la documentación de la aplicación

- Microsoft Office Word

Microsoft Office Word es la herramienta por excelencia para la redacción de trabajos.

2.9 Recopilación metódica de datos

Para poder cumplir de manera exitosa con los objetivos establecidos, se realizará un análisis riguroso del proceso de negocio del sistema de estacionamiento que implementa el municipio paceño. Para tal efecto, se procesará toda la información sobre los estacionamientos, tales como gastos, personal, ganancias, etc. También se analizará un nuevo método de pago

2.9.1 Fuentes primarias

La información se recolectará a través de la metodología de observación directa, analizando la estructura y funcionamiento del sistema de estacionamientos del GAMPL. Actualmente no existe un sistema de pagos similar al que se piensa emplear, por tal motivo todo el código fuente a utilizar será de elaboración propia.

Por otra parte, serán necesarios otros datos más específicos, como la ubicación de los estacionamientos, su numeración y las tarifas vigentes, entre otros. Esta información se podrá recolectar con ayuda de los de los cobradores, así como de otros funcionarios del GAMPL.

Sin embargo, debido a la Pandemia del Coronavirus no se podrán realizar entrevistas o reuniones. Por ello, algunos datos administrativos y del sistema se recolectarán mediante correo electrónico o las páginas web de las instituciones del Estado.

2.9.2 Fuentes secundarias

Durante el desarrollo de la aplicación será necesario contar con toda la documentación elaborada por Khipu, Firebase y Google Developers.

La documentación elaborada por Khipu es probablemente la más importante. Khipu será el método de pago del sistema. La API de Khipu integrará la aplicación a su sistema de manera que los usuarios puedan realizar pagos con tarjetas de crédito o débito funcionales en Bolivia.

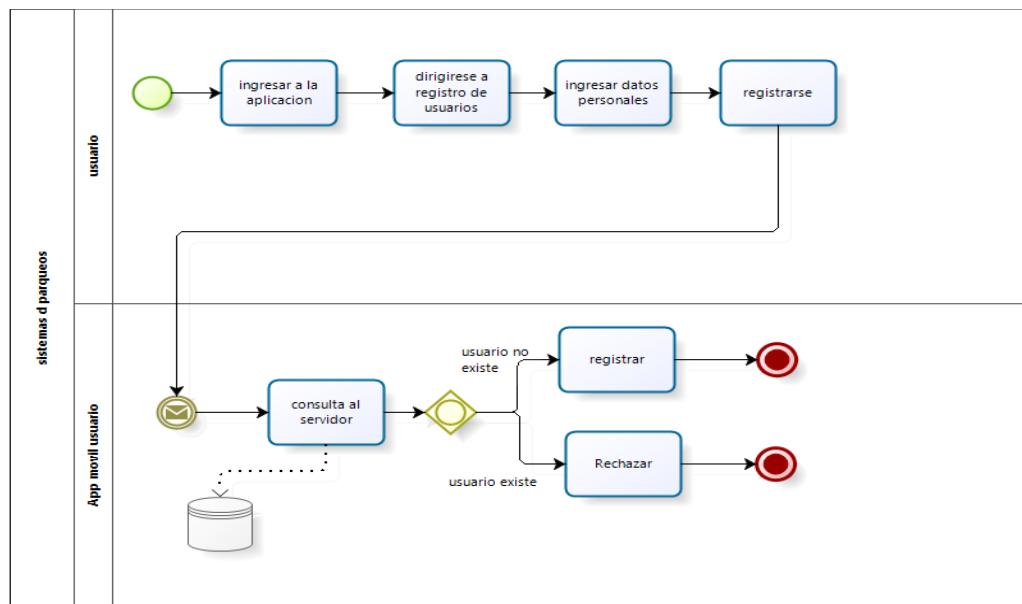
Firebase es la plataforma base de nuestro proyecto. Las API's de esta aplicación van a permitir realizar múltiples funciones como almacenamiento de datos, notificaciones en segundo plano, almacenamiento de documentos y enviar facturas a los correos electrónico, etc.

Las API's de Google Developers permitirán integrar los servicios de Google entre ellos. En el sistema se va realizar la integración de Google Maps con el sistema de Firebase. Esto permitirá tener los datos conectados, lo que también posibilitará que se puedan tener mejores estadísticas de uso en las consolas de los sistemas.

2.10 Análisis preliminar del proyecto

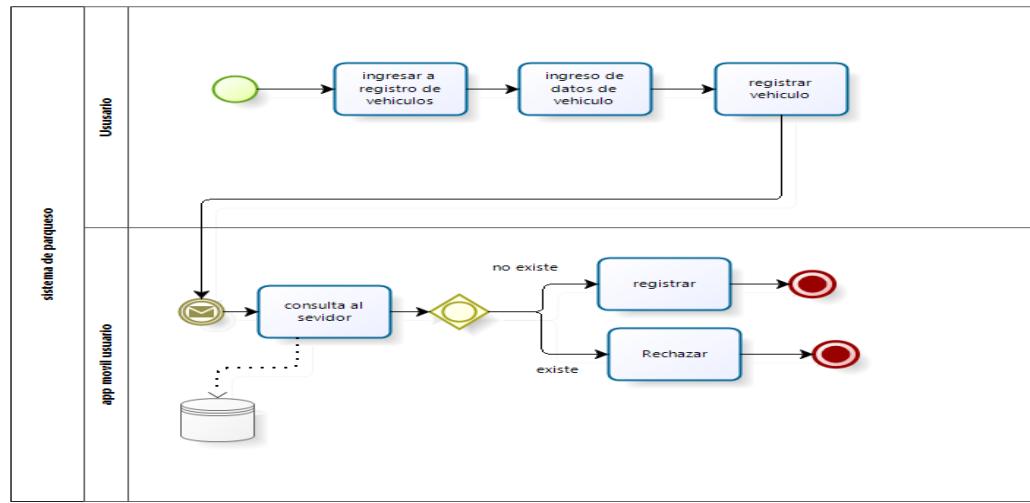
Para el análisis preliminar se realizaron varios diagramas en Bizagi. Para analizar los procesos que tanto los usuarios como los administradores tienen que realizar, se muestran en los siguientes diagramas los cuatro procesos principales que se requieren para el uso de la aplicación.

El primer proceso a analizar se refiere al registro de usuarios. Es necesario que todos los usuarios estén registrados en el sistema para su correcta identificación y para realizar la correspondiente facturación.



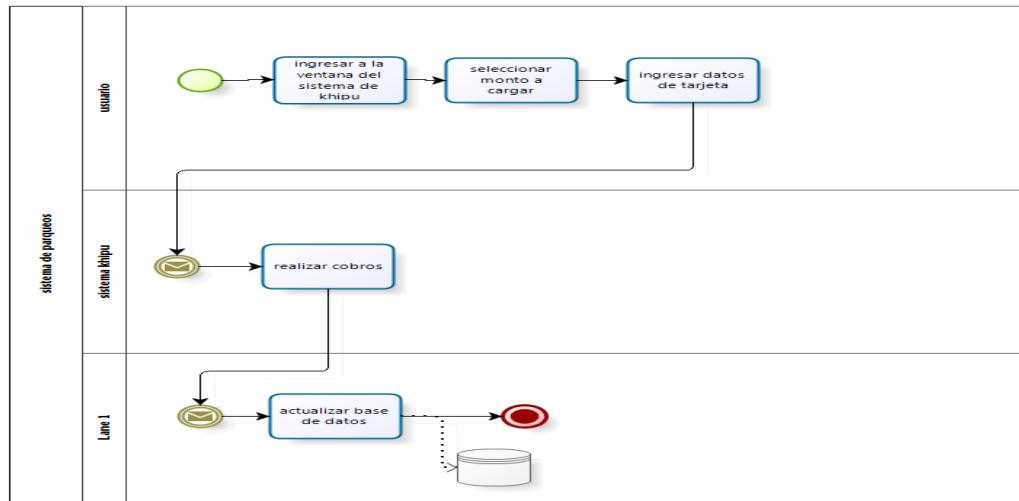
Procesos del sistema 1 Registro de usuarios

El segundo diagrama nos muestra el proceso de registro de vehículos. Se debe considerar que los usuarios pueden ser propietarios de uno o más vehículos. Por ello, es necesario que los usuarios registren todos sus automóviles. Esto también servirá para que los controladores verifiquen si el vehículo que reservó es el que está estacionado



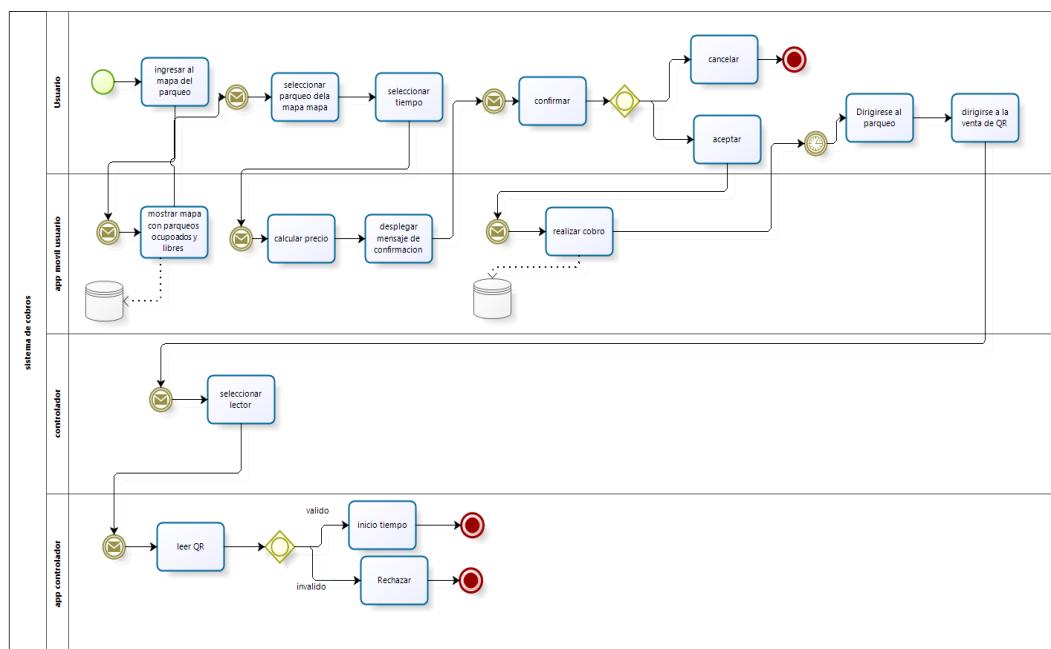
Procesos del sistema 2 Registro de vehículos

Para que los usuarios tengan una mejor experiencia y no tengan que ingresar los datos de su tarjeta repetidamente, la aplicación tendrá la opción de carga de crédito. Para ello, cada usuario puede seleccionar un monto a cargar en el sistema ingresando sus datos.



Procesos del sistema 3 Carga crédito

El proceso más importante del sistema es la parte de reservas. Es aquí donde el usuario podrá realizar la reserva de parqueos disponibles eligiendo el estacionamiento de su conveniencia en el mapa.



Procesos del sistema 4 Reserva de estacionamiento

2.11 Análisis Smart

El análisis SMART nos permite establecer un análisis del proyecto con objetivos medibles y alcanzables, esto nos va a ayudar a que el proyecto sea exitoso.

Especifico

El proyecto es bien claro y específico. Se pretende desarrollar una aplicación en la plataforma Android para realizar la reserva y el pago de los estacionamientos públicos que la alcaldía de la ciudad de La Paz ofrece. Los objetivos y alcances ya fueron establecidos en el presente documento.

Medible

A través del cronograma de implementación establecido, se podrá verificar el cumplimiento exitoso de las diversas etapas. Paralelamente, la metodología Scrum nos permitirá tener métricas sobre los avances y la calendarización.

Al mismo tiempo, el prototipado de la aplicación permitirá evaluar y testear su eficacia y eficiencia.

Por último, se utilizará el estudio de casos y las historias de usuario para una evaluación cualitativa.

Alcanzable

Todos los objetivos del presente proyecto son alcanzables. Es necesario señalar que no se necesita realizar inversiones de dinero para obtener la tecnología necesaria. Se utilizará, además, en su mayoría herramientas de software libre.

Real

Todas las herramientas a implementarse existen y son muy populares para los desarrolladores. Por ejemplo, Firebase es una plataforma de Google que da soporte de base de datos, almacenamiento y mensajería, entre otros servicios, que es utilizada por varias empresas y negocios exitosos de todo el mundo como “The New York Times”, “Trivago” y “The Economist” (Firebase, 2020).

Tiempo

Con la ayuda de Scrum, se podrá realizar el proyecto en los tiempos establecidos para obtener un prototipo funcional. El proyecto tiene una dimensión mediana por lo que se prevé poder terminar con la etapa de desarrollo a tiempo.

2.12 Análisis Part

El análisis PART permite realizar un análisis en la parte técnica del proyecto y permitirá realizar un análisis más específico de estos aspectos.

Procesos

Los procesos de este proyecto se definirán con las metodologías de UML (casos de uso y las historias de usuarios) y Mockups. Sin embargo, durante la realización del

proyecto estos procesos pueden de modificarse según se vayan identificando nuevas necesidades. La metodología Scrum permite realizar cambios según la necesidad.

Actores

En el presente proyecto se identificaron los siguientes actores:

- Usuarios:

Los Usuarios son los principales actores del proyecto. Son conductores y/o propietarios de vehículos que precisan usar los estacionamientos de la alcaldía.

- Controladores:

Los controladores son funcionarios de la alcaldía encargados de controlar y supervisar el correcto uso de los estacionamientos y su pago.

Recursos

Los recursos tanto de Hardware como de Software están establecidos en el presente documento. Las herramientas de Software van a ser libres o gratuitas. Todo el proyecto será administrado por la metodología Scrum

Tecnología

Se implementará una variedad y distintos tipos de tecnología. La mayor parte serán de Google. La aplicación se va a desarrollar en Java con Android Studio. La base de datos a implementar, será en tiempo real. Además, se tendrá almacenamiento en la nube. Se podrán lanzar notificaciones “Push”. Todos estos servicios los provee Firebase.

También se tendrá una pasarela de pagos llamada Khipu.

3 Marco teórico

3.1 Aplicación móvil

Los dispositivos móviles como las tabletas y los teléfonos inteligentes ejecutan aplicaciones como los ordenadores tanto portátiles como de escritorio. Estos están diseñados para realizar las mismas funciones que se tiene en los ordenadores, pero estas están comprimidas para que puedan estar al alcance de nuestras manos. (CGFAprende libre, 2017).

3.2 Android

Android es una iniciativa de Google. Es denominado “Proyecto de código abierto Android” o AOSP (por sus siglas en inglés). Android es una plataforma “Open Source”, que permite que sus usuarios o desarrolladores puedan leer, escribir, editar o modificar el código del sistema operativo como les guste. El AOSP tiene como objetivo principal crear estándares de compatibilidad con todos los creadores de dispositivos, para que todos puedan tener este sistema operativo personalizado y que los accesorios creados por las diferentes marcas y empresas puedan tener una compatibilidad y una experiencia de uso estándar (Android , 2018).

Arquitectura de Android

El sistema operativo de Android tiene 4 niveles y 5 capas como podemos observar en la imagen que se muestra más adelante. Todas las capas utilizan servicios a los niveles inferiores y ofrecen servicios a los niveles superiores. Los Niveles se los cuenta de abajo hacia arriba siendo el Kernel Linux el primer nivel y Aplicación el ultimo (FRANCO, 2012).

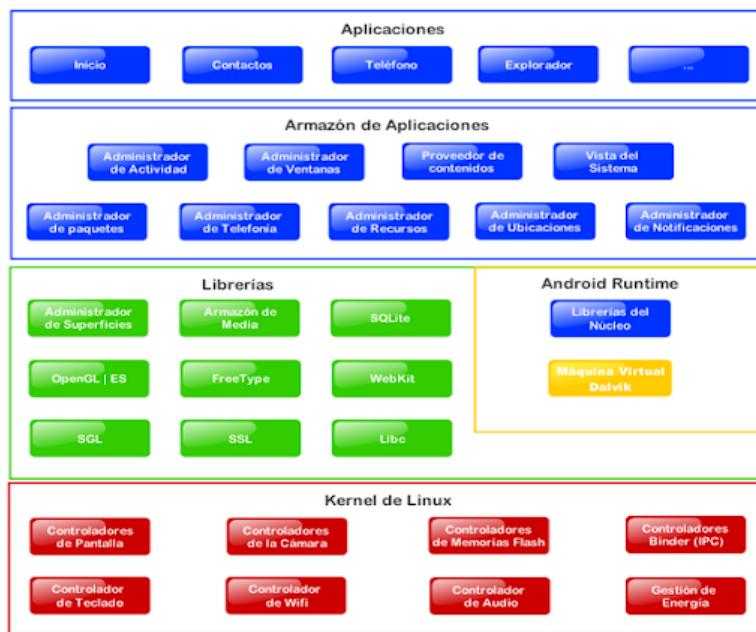


Ilustración 2 Arquitectura Android

Nota: Error! Reference source not found., imagen tomada de una publicación realizada por la universidad Carlos III de Madrid (Universidad Carlos III Madrid, 2018).

- **Aplicaciones:** Esta capa es la parte superior y visible. En esta sección es por donde los usuarios finales interactúan con el teléfono. Podemos visualizar todas las aplicaciones finales que el usuario puede usar. Las aplicaciones pueden ser las que vienen por defecto con el sistema de Android o como las aplicaciones desarrolladas por terceros o incluso por el mismo usuario. Todas las aplicaciones de esta capa utilizan los servicios API y librerías de los niveles anteriores. Estas aplicaciones pueden ser contactos, calendarios, Facebook o cualquier otra (Joshua J. Drake, 2014).
- **Framework o Armazón de aplicaciones:** Esta capa representa todas las herramientas utilizadas por los desarrolladores. Para crear, diseñar e implementar una aplicación en esta plataforma es necesario utilizar el conjunto de API, Frameworks y servicios que este nivel proporciona (Joshua J. Drake, 2014).

Las API más importantes en este nivel son:

Activity manager: Gestiona el ciclo de vida de todas las aplicaciones en este sistema operativo (Universidad Carlos III Madrid, 2018).

Window manager: Permite realizar la gestión entre las ventanas de las diferentes aplicaciones que tiene el dispositivo móvil. Para poder realizar estas gestiones es necesario que pueda utilizar los servicios de Surface Manager que está en el siguiente nivel (Universidad Carlos III Madrid, 2018).

Telephone manager: Incluye todas las API vinculadas con las funcionalidades del teléfono móvil como llamadas telefónicas, mensajes, etc (Joshua J. Drake, 2014).

Content provider: Permite que las aplicaciones que están instaladas en el teléfono puedan compartir la información entre ellas. Por ejemplo, gracias a Content Provider, la famosa aplicación de mensajería “WhatsApp” puede obtener toda la información que la aplicación de contactos tiene (Universidad Carlos III Madrid, 2018).

View System: Provee todos los elementos necesarios para que los desarrolladores puedan construir y diseñar las Interfaces de usuario. Aquí se proporciona todos los objetos necesarios para que los usuarios puedan interactuar con el teléfono móvil, como los botones, lista, textos, tamaño de ventanas, colores etc. (Universidad Carlos III Madrid, 2018).

Location Manager: Provee los servicios de ubicación y posicionamiento (Universidad Carlos III Madrid, 2018).

Notification Manager: Permite a las aplicaciones el manejo de notificaciones, las notificaciones son generalmente utilizadas por las aplicaciones para informar al usuario eventos o ocurrencias de manera inmediata para que el mismo pueda tener una mejor interacción con la aplicación (Universidad Carlos III Madrid, 2018).

XMPP Service: Permite realizar el intercambio de mensajes basado en XML (Universidad Carlos III Madrid, 2018).

- **Librerías:** Esta capa corresponde a las librerías utilizadas por Android. Estas librerías fueron escritas utilizando C/C++ y proporcionan a Android la mayor parte de sus capacidades más características junto a su núcleo basados en GNU/Linux. Estas librerías prácticamente son el corazón del sistema operativo Android.

Las librerías más conocidas o usadas son:

Librería libc: Aquí están todas las funciones y estándares del lenguaje C (Universidad Carlos III Madrid, 2018).

Librería Surface Manager: Es el gestor de los diferentes elementos de navegación de pantalla, también gestiona las ventanas de todas las aplicaciones abiertas (Universidad Carlos III Madrid, 2018).

Open GL/SL y SGL: GL/SL contiene todas las librerías gráficas que los dispositivos móviles utilizan. Además, permite gestionar gráficas en 3D. Si el hardware lo permite, puede crear gráficos en 3D con el mismo dispositivo móvil. SGL proporciona todas las gráficas en 2D. La mayoría de las aplicaciones que tienen los dispositivos móviles utilizan las gráficas en 2D, lo que hace que esta librería sea una de las más usadas por las aplicaciones.

Una de las características que sobresale de Android es la capacidad de poder combinar gráficas en 2D y 3D en una misma aplicación (Universidad Carlos III Madrid, 2018).

Librería Media Libraries: Proporciona todo el contenido necesario para los archivos multimedia como música, videos e imágenes (Universidad Carlos III Madrid, 2018).

FreeType: Permite manejar todo tipo de tipografías y fuentes deseadas (Universidad Carlos III Madrid, 2018).

Librería SSL: Permite el manejo del mismo protocolo, para que los usuarios puedan tener navegaciones más seguras por las aplicaciones que sirven de navegadores Web (Universidad Carlos III Madrid, 2018).

Librería SQLite: Permite crear, administrar y gestionar bases de datos relaciones SQL.

Librería WebKit: Proporciona un motor para las aplicaciones que sirven de navegadores web. (Universidad Carlos III Madrid, 2018).

- **Tiempo de ejecución:** Al mismo nivel de las librerías de Android se encuentra esta capa conocida como “entorno de ejecución”, está formada por lo que se conocen como las “Core Libraries”, que son librerías basadas en Java y otros tipos de ejecuciones. (Universidad Carlos III Madrid, 2018).
- **Núcleo Linux:** El sistema operativo tiene Kernel o núcleo Linux. Esta capa te permite tener una abstracción para el hardware. En esta capa se tienen todos los drivers necesarios para que exista una comunicación eficaz entre el hardware y el software. Siempre que un fabricante crea un dispositivo de hardware nuevo, lo primero que hace es crear los drivers necesarios para que pueda ser utilizado por los dispositivos Android (Universidad Carlos III Madrid, 2018).

3.3 Sistema de cobro Khipu

Khipu es un sistema de cobros en línea desarrollado en Chile. Fue implementado en Bolivia por Red Enlace el año 2013. Su principal función es integrar los sistemas de comercio electrónico a sus respectivas cuentas bancarias de una manera rápida y sencilla (Khipu, 2015).

Khipu es la primera herramienta de pagos electrónicos implementada en Bolivia, la misión de Khipu es facilitar los procesos de cobro y pago (Khipu, 2015).

Para poder cobrar con Khipu se tiene que estar afiliado a Red Enlace. Para lograr esta afiliación es necesario llenar un formulario de suscripción y seguir metódicamente los requerimientos que la empresa solicita (Khipu, 2015).

3.4 Firebase

Firebase es una plataforma para aplicaciones móviles y web desarrollada por Google desde el 2014. Su objetivo principal es facilitar la creación e integración de dispositivos móviles IOS y Android como web. Esta plataforma tiene varias funciones para que cualquier desarrollador pueda combinar y adaptar sus funciones. (IEBSchool, 2016)

3.4.1 Características de Firebase

Firebase es una plataforma muy completa, tiene 6 características principales: Analítica, desarrollo, poder de crecimientos, monetización, rapidez y agilidad. (IEBSchool, 2016)

Analítica

Firebase tiene un panel de control gratuito, intuitivo y fácil de usar. Con este panel, Firebase permite tener el máximo control de rendimiento de las aplicaciones tanto móviles como web.

Los datos que Firebase facilita al usuario para la toma de decisiones son datos reales. (IEBSchool, 2016)

Desarrollo

Firebase permite crear mejores aplicaciones móviles y web. Firebase ayuda a reducir el tiempo de desarrollo y de optimización mediante varias funciones en las que se destacan: Detección de errores, testeo de aplicaciones, supervisión de rendimiento, entre otras funciones (IEBSchool, 2016)

Monetización

Mediante la extensión “AdMob”, Firebase permite vender publicidad en sus aplicaciones. Esto ayuda a la aplicación a generar dinero según la cantidad de usuarios que la utilizan regularmente (IEBSchool, 2016).

Rapidez

Implementar Firebase puede ser fácil y rápido gracias a su API que es sencilla de entender y muy intuitiva. Gracias a esta API, Firebase se concentra en los problemas del cliente en vez ocuparse en la construcción de infraestructuras muy complejas (IEBSchool, 2016).

Agilidad

Firebase ofrece aplicaciones multiplataforma con API integradas a SDK individuales para IOS, Java, JavaScript. De esta manera se pueden desarrollar aplicaciones multiplataforma desde la misma plataforma (IEBSchool, 2016).

3.4.2 Servicios de Firebase

Real Time Database

Es una base de datos NO SQL alojada en la nube que permite almacenar y sincronizar todos los datos de los usuarios en tiempo real (Firebase, 2020).

Con la base de datos en tiempo real de Firebase, los usuarios accedan a sus datos desde cualquier dispositivo. Esto ayuda a trabajar en conjunto (Firebase, 2020).

Firebase permite tener una conexión en la nube sin tener que estar conectado a servidores. También permite ejecutar funciones en la nube ejecutando “Trigger” usando las funciones en Cloud Functions. Cuando los usuarios no tengan una conexión estable, o simplemente no tengan conexión a Internet Firebase, utiliza el cache del dispositivo para almacenar los datos y cuando el dispositivo se vuelve a conectar, los datos son sincronizados automáticamente en la nube. La base de datos en tiempo real de Firebase también se puede integrar al sistema de autenticación de Firebase y esto ofrece un modelo de seguridad declarativo, para permitir el acceso, según la identidad del usuario (Firebase, 2020).

Cloud FireStore

La base de datos en tiempo real de Firebase no es el único servicio de almacenamiento de datos que Firebase proporciona. Firebase también ofrece el servicio de FireStore es una base de datos NO SQL. Esta base de datos permite almacenar, sincronizar y consultar los datos de las aplicaciones a un nivel de escala.

FireStore está conformada por colecciones y documentos para estructurar datos con facilidad. Permite crear jerarquías para que los datos sean almacenados y relacionados y para que se pueda recuperar los datos mediante consultas expresivas de manera simple. Igual que la base de datos en tiempo real, FireStore permite acceder a los datos almacenados desde dispositivos móviles o web sin la necesidad de un servidor propio. FireStore permite ejecutar código en Backend alojado en la nube con Cloud Functions, sin la necesidad de tener código en la aplicación móvil (Firebase, 2020).

De la misma manera que en la base de datos en tiempo real, permite almacenar datos en el cache del teléfono sin la necesidad de una conexión a Internet, para después sincronizarla una vez se tenga una conexión estable (Firebase, 2020).

Con la tecnología de infraestructura de FireStore, se puede también diseñar una aplicación de gran escala para los negocios sin manejar y administrar servidores (Firebase, 2020).

Cloud Functions

Como se mencionó anteriormente, Cloud Functions permite crear código en la nube sin la necesidad de manejar y administrar servidores. Cloud Functions funciona con funciones escritas en JavaScript y estas funcionan en un entorno node.js seguro. Estas funciones solo se ejecutan cuando se emite un evento específico bajo observación.

Cloud Functions no necesita de mucho mantenimiento. Para implementarla, es necesario un solo comando, después Cloud Functions realiza los ajustes de escala de los recursos de procesamiento automáticamente para que coincidan con los patrones de uso dependiendo del comportamiento de la aplicación (Firebase, 2020).

Cloud Functions protege el código de la aplicación de una mejor manera, ya que en muchas ocasiones el código puede ser modificado por el cliente. Cloud Functions está completamente aislada del cliente así que las funciones usadas en la lógica del negocio son completamente seguras (Firebase, 2020).

Kit AA

El Kit de AA o Kit de Artificial Intelligence permite incorporar a la aplicación características potentes de aprendizaje automático para cualquier dispositivo, sea Android o IOS. Una de las mejores características de este Kit es que no es necesario ser un experto en inteligencia artificial para poder implementarlo (Firebase, 2020).

Este Kit viene con API's ya listas para usar. Estas API's son: Reconocimiento de texto, rostros, escanear de códigos de barra y QR y reconocimiento de imágenes, entre otras funciones. El Kit AA permite tener la API en el mismo dispositivo móvil o implementarla en la nube. Estas API's son muy intuitivas y fáciles de usar. La API para los dispositivos móviles funciona más rápido que la API en la nube y puede usarse sin una conexión a internet. La API en la nube usa de mejor manera la potencia de la tecnología de aprendizaje automático y proporciona un mejor nivel de precisión (Firebase, 2020).

Authentication

Authentication de Firebase ofrece una solución directa, personalizable y de código abierto, Firebase Authentication implementa recomendaciones para la autenticación de credenciales en dispositivos móviles y web, esto ayuda a maximizar las capacidades de la aplicación (Firebase, 2020).

La seguridad de Firebase, ha sido creada por el mismo equipo de Google que desarrollo “Smart Lock”, acceso con Google, entre muchos otros sistemas de seguridad y manejo de contraseñas. Este equipo aplica toda la experiencia de Google en la administración de base de datos de las cuentas más grandes y seguras del mundo (Firebase, 2020).

Por lo general implementar un sistema propio de autenticación puede tomar varios meses, pero con Firebase Authentication esto sólo demanda un par de minutos (Firebase, 2020).

Hosting

Firebase permite alojar una página Web personalizada capaz de integrarse a los dispositivos móviles o a los otros servicios de Firebase. Los contenidos de las webs son actualizados inmediatamente (Firebase, 2020).

Firebase ofrece certificados SSL gratuitos y personalizados para cada sitio web (Firebase, 2020).

Cloud Storage

Cloud Storage permite almacenar rápida y fácilmente todos los contenidos multimedia que los usuarios manejen. La tecnología de Cloud Storage permite pasar la aplicación de prototipo a producción con las mismas tecnologías que respalda Spotify (Firebase, 2020).

Con Cloud Storage se pueden manejar las descargas y cargas de archivos por partes. Cuando el dispositivo móvil pierda conexión, las descargas o cargas se pausarán de manera automática y se reanudarán al momento que se restablezca la conexión (Firebase, 2020).

3.5 Códigos QR

Los códigos QR son imágenes bidimensionales. Cuando son escaneados suelen contener direcciones a páginas web, textos, imágenes, videos, etc. en esencia los QR son links pictográficos (Coleman, 2011).

Los códigos QR fueron inventados por Denso Wave, una compañía subsidiaria de Toyota y fueron liberados al público en 1994. Como se mencionó previamente, los QR contienen la información en dos dimensiones: Horizontal y vertical. Son capaces de codificar 4,296 caracteres alfanuméricos. (Coleman, 2011)

3.6 Scrum

En 1986 Takeuchi y Nonaka publicaron un artículo llamado “The New Product Development Game”. Este artículo da a conocer una nueva forma de gestionar proyectos en la que la agilidad, flexibilidad y la incertidumbre son elementos principales para el desarrollo de proyectos especialmente proyectos de software (Scrum, 2017).

Los autores al comparar empresas relacionadas con tecnología, pudieron ver que había empresas de un mismo entorno que realizaban productos en mucho menos tiempo y con menos costo que sus competidoras, todo esto lograron hacer si tener que bajar el nivel de calidad (Scrum, 2017).

Una de los aspectos más relevantes que pudieron observar los autores, era que no existían fases claras ni especializadas. Estas empresas partían de requisitos generales y sus productos los realizaba un mismo equipo multidisciplinario (Scrum, 2017).

Esta metodología fue denominada SCRUM, debido a que sus formas de trabajo son muy parecidas a los colaborativos que hacen los jugadores de rugby (Scrum, 2017).

Jeff Sutherland y Ken Schwaber presentaron, en 1996, prácticas que las usaban como proceso formal para el desarrollo de software. Estas prácticas, con el tiempo, se incluirían en la alianza de desarrollo ágil (Scrum, 2017).

Scrum es un modelo adecuado para las empresas que se caracterizan por tener productos (Scrum, 2017).

Incertidumbre: Se plantean los objetivos que se quieren alcanzar donde no es necesario presentar un plan detallado (Scrum, 2017).

Auto organización: No es necesario que una persona externa al equipo este realizando controles, si no que los equipos son capaces de organizarse por ellos mismo. Los equipos tienen que reunir las siguientes características: Autonomía, auto superación y auto enriquecimiento (Scrum, 2017).

Control moderado: Se basa en crear escenarios entre iguales para que no se impida la creatividad y la espontaneidad. Se establecerá un control moderado (Scrum, 2017).

Trasmisión del conocimiento: Todos tienen que compartir sus conocimientos con el equipo. Se emplea un desarrollo colaborativo. Cada miembro del equipo puede compartir sus proyectos con otros (Scrum, 2017).

Scrum al ser una metodología de desarrollo ágil, funciona con lo que se denomina iteraciones o Sprint, estos no son nada más que ciclos breves para el desarrollo (Scrum, 2017).

Para entender el ciclo de desarrollo es necesario conocer las 5 fases que define el desarrollo ágil:

- Concepto: Se definen de forma general las características del producto, después se asigna el equipo que se encargaría de su desarrollo (Scrum, 2017).
- Especulación: se establecen los límites y alcances del proyecto, además se creará una agenda y los costos que el proyecto demandará (Scrum, 2017).

- Exploración: se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación (Scrum, 2017).
- Revisión: El equipo revisa todo lo que se ha construido y se compara con el objetivo que se planteó previamente (Scrum, 2017).
- Cierre: Se pasa a la fase de implementación donde se entrega un prototipo funcional, esto no significa que el proyecto termine, si no se podrán realizar cambios o modificaciones al producto (Scrum, 2017).

4 Análisis

Se plantea desarrollar dos aplicaciones móviles para teléfonos inteligentes con sistema operativo Android. Estas aplicaciones van a estar orientadas al facilitar y mejorar la administración y el control de los estacionamientos dispuestos por la alcaldía de la ciudad de La Paz en Bolivia. Tras hacer un análisis de negocio del sistema actual que tiene la alcaldía se plantea cambiar varios aspectos y mejorar otros para que el funcionamiento sea el adecuado. En primera se plantea eliminar a los cobradores ya que este es un sistema que genera muchos gastos y no proporcionan buenas condiciones de trabajo. Se implementará un método de pago funcional y implementado en Bolivia. También se implementará una base de datos en tiempo real y varias funcionalidades más que permitan que el sistema funcionar de manera rápida.

4.1 Usuarios

La metodología enfocada en las personas permite entender a los usuarios que utilizarán la aplicación y, a partir de este análisis, se pueden desarrollar características específicas para satisfacer las necesidades de los usuarios. Se pudo distinguir dos diferentes tipos de usuarios que van a utilizar esta plataforma regularmente, a estos dos grupos vamos a sumarle el grupo específico que serían los trabajadores de la alcaldía.

1. Estudiantes universitarios: Muchos estudiantes viven en zonas alejadas a sus respectivas universidades, tienen clases a altas horas de la noche, viven en zonas

poco seguras. Debido a estos motivos, muchos de ellos prefieren utilizar su propio automóvil para transportarse a sus clases.

2. Trabajadores del sector privado y público: Una gran parte de estos usuarios viven en la zona sur de la ciudad y trabajan en zonas céntricas como Sopocachi. El transporte público en la ciudad tiene varias deficiencias: no es limpio, es poco seguro, poco cómodo y el tiempo de traslado es muy largo en la mayoría de los casos. Por estos motivos, muchos trabajadores que tienen que cumplir ciertos horarios de llegada a sus respectivas fuentes de empleo así como personas que salen de sus oficinas a altas horas de la noche, prefieren utilizar su propio medio de transporte para movilizarse. Estas personas están dispuestas a pagar cierto dinero extra para disponer de un estacionamiento seguro.
3. Controladores: Los controladores tienen como función principal hacer cumplir las reglas y requisitos de la aplicación. Para tal efecto, tendrían que estar verificando y revisando su funcionamiento.

4.2 Planificación

Para realizar la planificación de versiones de la aplicación y en base a las funcionalidades definidas en el siguiente punto, se realizó un User Story Mapping. Este diagrama tiene en el eje vertical las características de la aplicación, mientras que en el eje horizontal están las versiones en orden cronológico. De esta forma se planifica qué historia de usuario se trabajará y en qué orden.

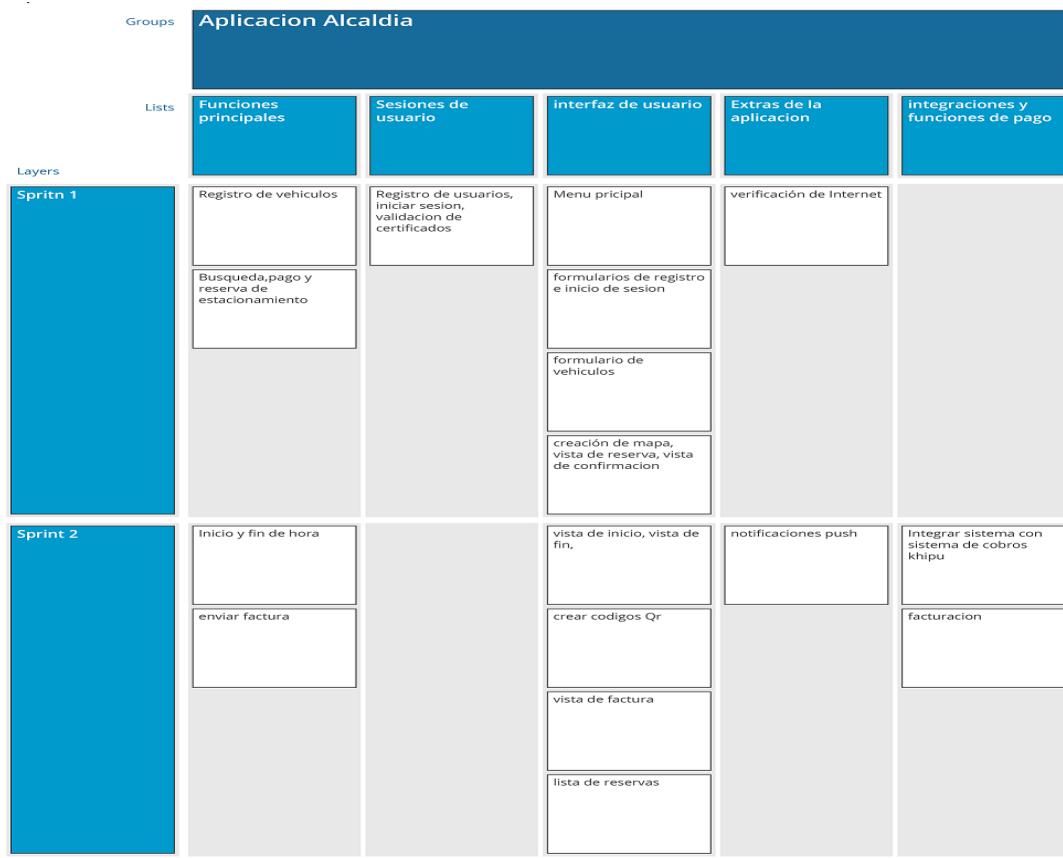


Ilustración 3 Planificación 1

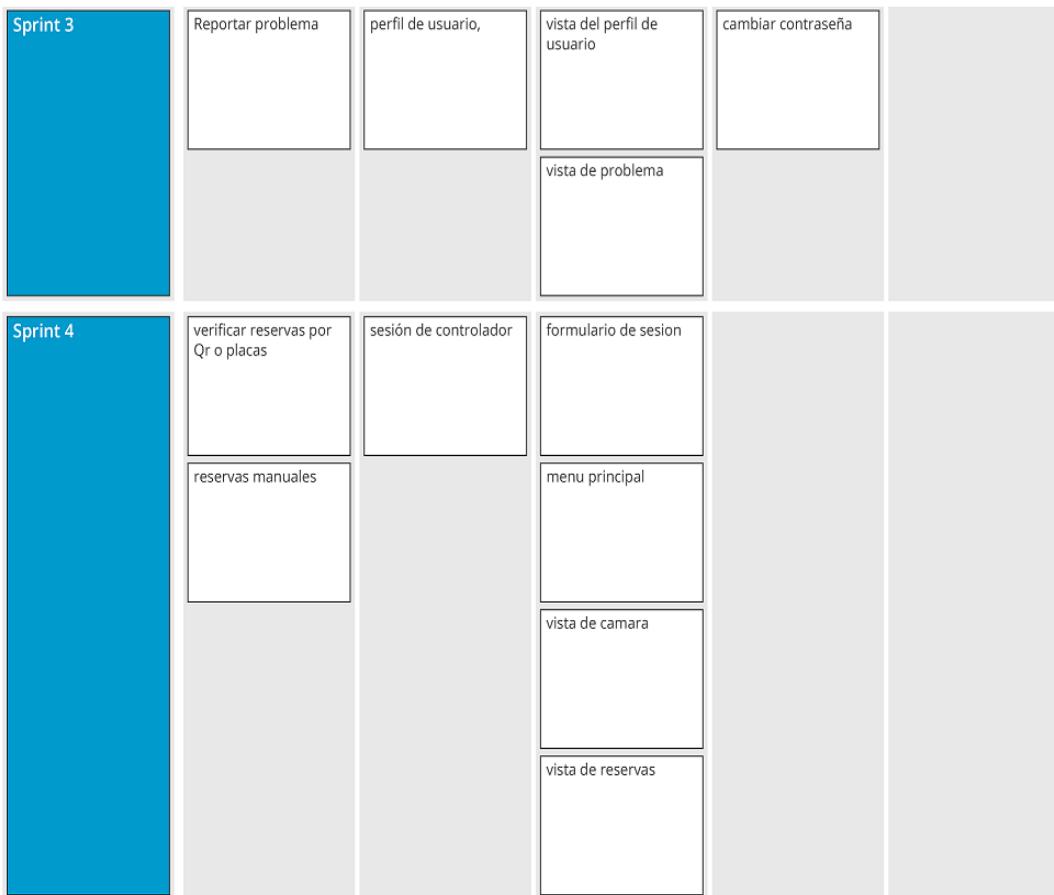


Ilustración 4 Planificación 2

4.3 Historias de usuarios

Las historias de usuario dan una descripción básica de como el sistema de parqueos va a estar armado

4.3.1 Primer sprint

4.3.1.1 Registro de usuarios, inicio de sesión, validación de certificados

Como usuario regular, necesito estar registrado en el sistema para que puedan identificarme

Historia de usuario

Nombre: Registro de usuario, inicio de sesión, validación de certificados

Numero: 1

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: Para que los usuarios puedan usar la aplicación, necesitan registrarse en el sistema llenando los datos que se muestran en la Ilustración 5 Registro de usuarios. Este es un requisito obligatorio. En el campo de la contraseña, el usuario podrá ver si ésta es correcta haciendo click en el icono de la derecha. Cuando los usuarios ya estén registrados o cuando cambien de dispositivo móvil podrán iniciar sesión en cualquier dispositivo Android, llenando los datos de su correo electrónico y contraseña como se muestran en la “Ilustración 6 Iniciar sesión”, para que puedan ser identificados en el sistema y puedan realizar todas las operaciones que ellos deseen. Cuando el usuario inicie sesión en su dispositivo, no tendrá que volver a realizar este proceso otra vez hasta que cierre la sesión de manera intencional, como se muestre en la historia de usuario “Perfil de usuario, cambiar contraseña y cerrar sesión”. Al ingresar a la aplicación, si el usuario está con sesión iniciada el sistema pasará directamente a la Ilustración 7 Menú principal .

Validación:

En la **Error! Reference source not found..**

- Primera parte
 - Se llenan todos los campos
 - Se presiona el botón de registro.
 - Si las contraseñas coinciden se registra al usuario, caso contrario sale error de registro
- Segunda parte
 - Se llenan los campos parcialmente
 - Se presiona el botón de registro
 - Sale mensaje de error “llenar todos los campos”

En la Ilustración 6 Iniciar sesión

- Primera parte
 - Se llenan todos los campos
 - Se presiona el botón de inicio de sesión.

- Si las contraseñas coinciden con la del registro en la Base de Dato se ingresa a Ilustración 7 Menú principal , caso contrario sale error de usuario o de contraseña.
- Segunda parte
 - Se llenan los campos parcialmente
 - Se presiona el botón de Inicio de Sesión
 - Sale mensaje de error “llenar todos los campos”



Ilustración 5 Registro de usuarios

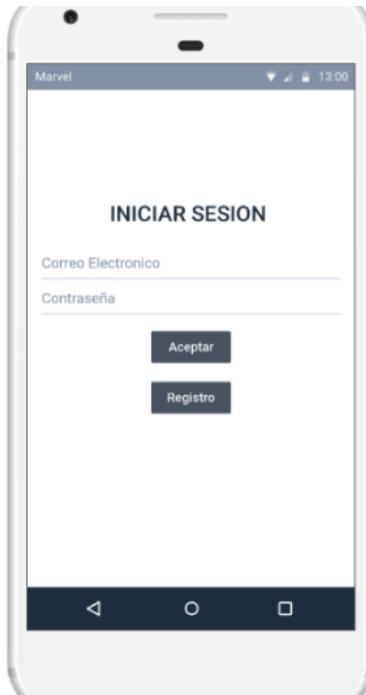


Ilustración 6 Iniciar sesión

4.3.1.2 Menú Principal

Como usuario, debo poder moverme con facilidad entre las pantallas de la aplicación con un menú fácil de usar.

Historia de usuario

Nombre: Menú principal

Numero: 2

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: El usuario debe poder moverse en la aplicación de manera simple y fácil de entender. La aplicación tendrá un menú de navegación con 5 botones como nos muestra la Ilustración 7 Menú principal. Los 6 botones son: 1) Registro de vehículos que nos dirige a la Ilustración 8 Registro de Vehículos, 2) Reserva de parqueos

nos direcciona a Ilustración 9 Mapa de estacionamientos, 3) Saldo a la Ilustración 9 Mapa de estacionamientos, 4) Mis reservas a la Ilustración 18 Lista de reservas, 5) mi perfil a la Ilustración 19 Mi perfil.

Validación:

- Se presiona los todos los botones en la Ilustración 7 Menú principal.
- Se verifica si las direcciones son validas según los requerimientos.

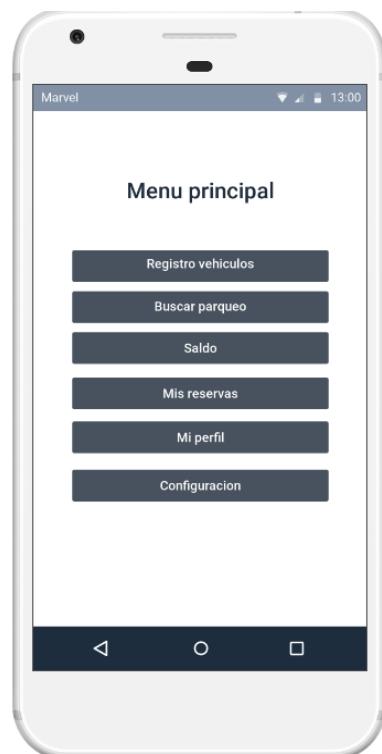


Ilustración 7 Menú principal

4.3.1.3 Registro de vehículos

Como usuario, necesito registrar mis vehículos para que puedan ser identificados por los controladores de la alcaldía.

Historia de usuario

Nombre: Registro de vehículos

Número: 3

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: Cuando el usuario desee reservar un estacionamiento, primero debe de tener por los menos un vehículo registrado en el sistema. Para poder realizar este registro, el usuario debe de llenar todos los campos de la "Ilustración 8 Registro de Vehículos" de manera obligatoria. Una vez llenados, el sistema revisará las bases de datos para ver si el vehículo ya está registrado o no. Si el vehículo no está en el sistema, el sistema procederá a regístralos, caso contrario saltará una alerta notificando la existencia del mismo.

Validación:

- Primera parte
 - Se llenan todos los campos
 - Se presiona el botón de Registro.
 - Se verifica el numero de placa del vehículo, si no existe ningún vehículo con esa placa se registra. Caso contrario sale mensaje de error
- Segunda parte
 - Se llenan los campos parcialmente
 - Se presiona el botón de Registro
 - Sale error de usuario o contraseña

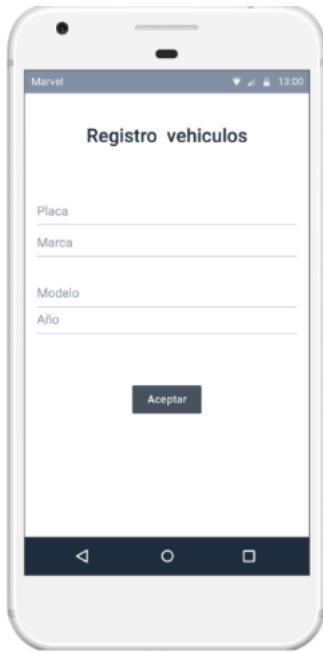


Ilustración 8 Registro de Vehículos

4.3.1.4 Búsqueda, reserva y pago estacionamiento disponible

Como usuario, necesito poder ver todos los parqueos disponibles para poder reservar el más apropiado.

Historia de usuario

Nombre: búsqueda, reserva y pago de estacionamientos disponibles

Número: 4

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: Cuando el usuario necesite de un estacionamiento en alguna zona de la ciudad de La Paz, el usuario debe ingresar a la Ilustración 9 Mapa de estacionamientos, en este mapa se va a mostrar todos los estacionamientos que están disponibles la próxima hora en la ciudad de La Paz. El usuario debe de hacer click en los pinos donde desee reservar el estacionamiento. Una vez seleccionados se desplegará la pantalla que se muestra en la Ilustración 10 Reserva de parqueo, en esta se nos muestra los datos básicos sobre la reserva que se va a realizar, aquí el usuario debe completar dos campos uno es el tiempo que desea el usuario reservar el estacionamiento y además del seleccionar uno de los vehículos que tiene registrados.

Una vez que se completen estos pasos el cliente debe hacer click en el botón reservar. Después nos va a saltar la Ilustración 11 Confirmar y pagar, esta nos muestra el nuestro saldo restante y el costo del total del parqueo, además del tiempo de validez del parqueo. Una vez que tengamos todos los datos confirmados sobre la reserva hacemos click en aceptar para que el sistema realice la reserva. Para que el sistema pueda realizar la reserva, primero tiene que verificar que el saldo de la cuenta es mayor o igual al costo del parqueo. Si estos datos no son válidos, el sistema rechaza la reserva y solicita que se realice una carga para que se pueda continuar con la reserva. Si la reserva es positiva, el sistema mostrará la “Ilustración 12 Inicio de hora”.

Si el usuario salió de la aplicación y ya realizó una reserva, puede volver a entrar a la pantalla ingresando a “Ilustración 9 Mapa de estacionamientos”. En esta pantalla sólo se mostrará la ubicación del parqueo ya reservado. Cuando el cliente haga click en el pin se mostrará la pantalla de la “Ilustración 12 Inicio de hora”.

Validación:

Ilustración 9 Mapa de estacionamientos

- Se navega en el mapa.
- Todos los pines tienen que coincidir con los estacionamientos disponibles.
- Al presionar sobre uno de los pines se ingresa a la Ilustración 9 Mapa de estacionamientos
-
-

Ilustración 9 Mapa de estacionamientos

- Al ingresar se tiene que tener todos los valores correspondientes al estacionamiento seleccionados previamente.
- Se comprueba la hora fecha y los demás datos que tiene que coincidir con el sistema.
- Se selecciona el tiempo de estadía
- Se selecciona el vehículo a usar

- Los vehículos en la lista deben de coincidir con los vehículos registrados por el usuario.
- Al presionar aceptar se debe ingresar a la Ilustración 11 Confirmar y pagar.

Ilustración 11 Confirmar y pagar

- Al ingresar todos los valores que se tenían de la pantalla anterior tienen que coincidir.
- El calculo del precio tiene que ser correcto en la proporción de tiempo y hora
- Al presionar aceptar se tienen que realizar la reserva y descontar el monto.
- En caso de que el monto del usuario sea menor al requerido sale mensaje de error
- Al presionar cancelar el sistema se dirige a **Error! Reference source not found.**

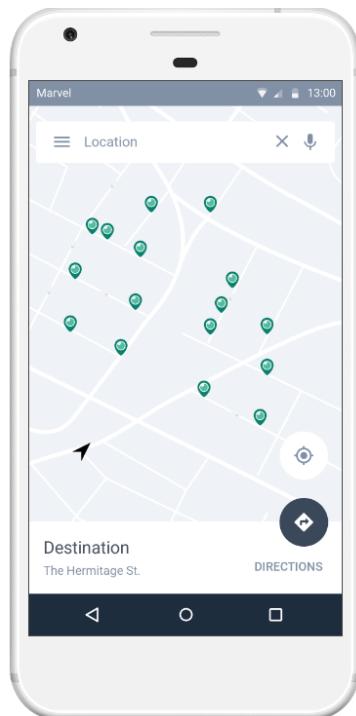


Ilustración 9 Mapa de estacionamientos



Ilustración 10 Reserva de parqueo



Ilustración 11 Confirmar y pagar

4.3.2 Segundo sprint

4.3.2.1 Inicio y fin de Hora

El usuario debe mostrar el código QR que se genera para que la hora del parqueo contratado inicie oficialmente.

Historia de usuario

Nombre: Inicio y fin de hora

Número: 5

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: Despues de realizar la reserva, cuando el usuario llegue al estacionamiento ya pagado, podrá dar inicio a su hora mostrando el código QR que se muestra en la “Ilustración 12 Inicio de hora”. En esta ilustración se puede observar la pantalla que muestra los datos esenciales del parqueo y el código QR que tiene que ser escaneado por la aplicación del controlador que se muestra en la historia de usuario Verificar reserva por QR y placas (controlador).

Una vez iniciada la hora, se verá la “Ilustración 13 Fin de hora” en la pantalla. Se puede ampliar el tiempo de reserva si fuese necesario o bien finalizarlo.

Para ampliar el tiempo, el sistema verifica que no exista otra reserva en la siguiente hora. Si así fuese, no se aceptará la ampliación, pero se puede realizar un cambio de ubicación.

Cuando el tiempo de la reserva se está acabando, el sistema mandará una notificación informando que el tiempo se agota y que se debe ampliar la hora o finalizar la misma.

En caso de que el usuario llegue y el estacionamiento este ocupado, el usuario podrá reservar otro espacio de manera simple como se muestra en la historia de usuario “Reportar problema”.

Validación:

Ilustración 12 Inicio de hora

- El dato de la fecha tiene que coincidir con el de la fecha del día.
- Los datos de Hora de inicio y Hora de fin tienen que coincidir con los datos del sistema
- El cálculo del precio tiene que ser el mismo ya hecho previamente.
- El código QR tiene que ser un código especial para dicha reserva.

- Una vez que el código QR sea leído por la aplicación de controlador se tiene que pasar a la Ilustración 13 Fin de hora

Ilustración 13 Fin de hora

- Al ingresar a esta pantalla se comienza a transcurrir el tiempo del usuario.
- Tiempo transcurrido es el tiempo que ya paso el usuario usando el estacionamiento y tiempo faltante es el tiempo que falta.
- Todos los demás datos deben de coincidir con los datos de la ilustración previa.
- Al presionar los botones de mas tiempo se comprueba si el usuario tiene saldo disponible, si el usuario tiene saldo el tiempo disponible faltante se aumenta según lo necesario. Si no hay saldo disponible sale mensaje de error.



Ilustración 12 Inicio de hora



Ilustración 13 Fin de hora

4.3.2.2 Carga de crédito

Los usuarios necesitan poder realizar los pagos al sistema de una forma fácil y eficiente.

Historia de usuario

Nombre: Carga de crédito

Numero: 6

Usuario: Cliente

Prioridad: Alta

Desarrollador: José Estenssoro

Descripción: Cuando los usuarios deseen pagar es necesario que puedan acceder a una plataforma de pago fácil y eficiente. Es importante, además, que esta plataforma esté regulada por las leyes bolivianas.

Para realizar los pagos, los usuarios se podrán conectar a la plataforma “Khipu” de Red Enlace. Para ello, primero tienen que ingresar a la Ilustración 14 Sistema de carga. El usuario tiene que hacer click en el botón “ir al sistema de carga”. Este botón los direccionara al sistema de cobros Khipu como se puede ver en la “Ilustración 15 sistema de carga Khipu”, donde tienen que seguir las instrucciones para que la carga sea realizada con éxito.

También se puede ver que existe un segundo botón con el nombre “Historial de carga”, este direcciona a la “Ilustración 16 lista de recargas” y su función será explicada en “Historial de carga y facturación”.

Validación:

Ilustración 14 Sistema de carga

- Los datos de saldo tienen que coincidir con los datos del sistema.
- Todas las transacciones de carga y pagos se las registra en la base de datos, al sumar y restar estos valores tiene que coincidir con el valor del sistema.

Ilustración 15 sistema de carga Khipu

- Una vez realizadas todas las operaciones requeridas por “Khipu” se presiona el botón finalizar.
- Una vez finalizada la operación, el sistema validará los datos e informará si la operación fue exitosa. Si así fuese, el sistema realizará la carga en la base de datos y hará los registros correspondientes.
- Se validará la operación si al salir del sistema no existe ninguna carga pendiente. La operación quedará invalidada si no se concretan todos los requerimientos exigidos por “Khipu”



Ilustración 14 Sistema de carga

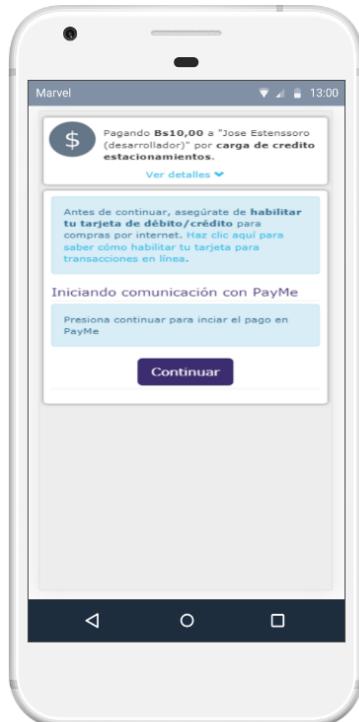


Ilustración 15 sistema de carga Khipu

4.3.2.3 Historial de carga y facturación

Los usuarios precisan conocer todos sus gastos.

Historia de usuario

Nombre: Historial de Carga de crédito y facturación

Numero: 7

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: Esta historia de usuario muestra una lista de todas las recargas que se hicieron en el sistema señalando los montos y las fechas de las operaciones tal como se muestra en la “Ilustración 16 lista de recargas”. En esta parte, el sistema también va a generar una factura similar a la de la “Ilustración 17 Factura”.

Validación:

Ilustración 16 lista de recargas

- Los datos que muestra la pantalla deben coincidir con los datos del usuario en el sistema.
- Al presionar cualquier dato de la lista, el sistema ingresa a “Ilustración 17 Factura”.

Ilustración 17 Factura

- Los datos se tienen que generar según los requisitos de Impuestos Nacionales.

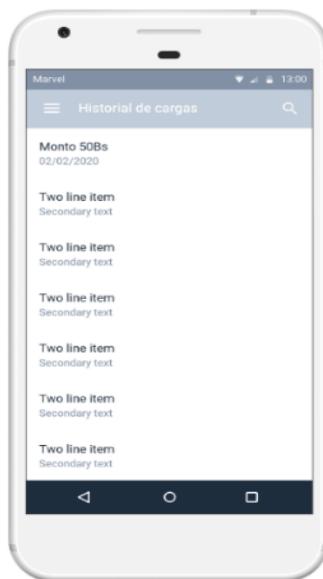


Ilustración 16 lista de recargas

**GOBIERNO AUTONOMO
MUNICIPAL DE LA PAZ**
 Sucursal Nro. 142
 Av. Montes Nro. 451 Z. Challapán
 La Paz - Bolivia
FACUTRA
 Nit: 1029241022
 Nro. Factura: 36355
 Nro. Autorización: 4753019000
 11000
 Total: 3.00
 Por e IVA
 Fecha: 08/07/2019
 Nombre: S/N
 Nit/Vat: 0
 CANT CONCEPTO PRECIO SUBTOTAL
 1 EST-PLUB X300 3.00 3.00
 TOTAL (Bs): 3.00
 IFE (Bs): 0.00
 CAMBIO (Bs): 0.00
 SON: TRES 00/100 Bolivianos
 CÓDIGO DE CONTROL: 00-19-22-19-
 8F
 FECHA LÍMITE DE EMISIÓN: 08/07/2019



ESTA FACTURA CONTIENE AL DESARROLLO DEL 24/15
 P. 100 H. CTO D. ESTA SERA SIN USO AL 30/15
 HABRÁ UN CTO D. ESTA SERA SIN USO AL 30/15
 Let N. 451 S. 10. La licencia de la actividad
 para el uso de la licencia de la actividad

Ilustración 17 Factura

Nota: La Ilustración 17 Factura es una fotografía de una factura real del municipio.

4.3.2.4 Lista de reservas y gastos

Los usuarios precisan estar bien informados sobre la utilización de los estacionamientos que realizó y su correspondiente gasto.

Historia de usuarios

Nombre: Lista de reservas y gastos

Numero: 8

Prioridad: alta

Programador: José Estenssoro

Descripción: El usuario va a poder ver su “Historial de reservas”. Es decir, sus gastos y cuanto tiempo estuvo en los estacionamientos. El usuario tendrá un resumen muy claro de todas sus actividades en el sistema.

Ilustración 18 Lista de reservas

Validación:

- Los datos deben de coincidir con los datos del sistema.

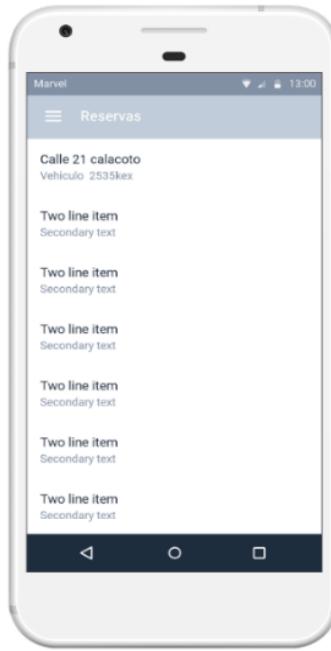


Ilustración 18 Lista de reservas

4.3.3 Tercer sprint

4.3.3.1 Perfil de usuario, cambiar contraseña y cerrar sesión

El usuario precisa actualizar sus datos personales para que este bien identificado en el sistema.

Historia de usuario

Nombre: Perfil de usuario, cambiar contraseña y cerrar sesión

Número: 9

Usuario: Cliente

Prioridad: Baja

Programador: José Estenssoro

Descripción: En la Ilustración 19 Mi perfil de la presente historia de usuario, se puede apreciar la información básica del usuario que consiste en el nombre completo, la licencia de conducir, su correo electrónico, además de una foto que si el desea puede subir en cualquier momento. También se puede observar que el usuario tiene su lista de vehículos registrados y además puede eliminar dichos vehículos simplemente haciendo click en el basurero.

El usuario también puede realizar el cambio de su contraseña, escribiendo su contraseña actual y una nueva.

También podemos ver que el usuario puede salir del sistema haciendo click en “cerrar sesión”.

Validación:

- Los datos del usuario tienen que coincidir con los datos registrados.
- En la sección de lista de vehículos se tendrá un listado de todos los vehículos registrados por el usuario.
- Al presionar en el botón del basurero, el sistema tiene que eliminar el vehículo de la lista del usuario.
- Luego del cambio de la contraseña, la antigua debe coincidir con la introducida para el cambio de la misma.

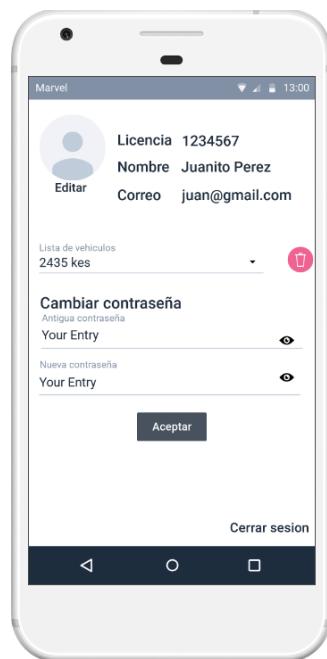


Ilustración 19 Mi perfil

4.3.3.2 Reportar problema

El usuario, al encontrar algún problema, necesita reportarlo para que el mismo sea resuelto a la brevedad posible.

Historia de usuario

Nombre: Reportar un problema

Número: 10

Usuario: cliente

Prioridad: alta

Programador: José Estenssoro

Descripción: Cuando el usuario llega a su estacionamiento y ve que el usuario anterior u otra persona lo está utilizando, debe realizar la correspondiente denuncia y reclamo. Debe hacer click en el botón de reportar que está en la “Ilustración 19 Mi perfil”, presionando el botón de solicitar. El sistema va a buscar un nuevo estacionamiento cercano.

Validación:

- Primera parte
 - Se llenan los campos requeridos y se los almacenan en el sistema.
 - El primer campo es obligatorio el segundo es opcional.
- Segunda parte
 - Al solicitar un nuevo estacionamiento se ingresa a la “Ilustración 9 Mapa de estacionamientos” para que el usuario pueda elegir un nuevo estacionamiento.
 - Una vez seleccionado el estacionamiento el usuario ya no tiene que realizar ninguna otra transacción y el sistema se dirige a la Ilustración 12 Inicio de hora

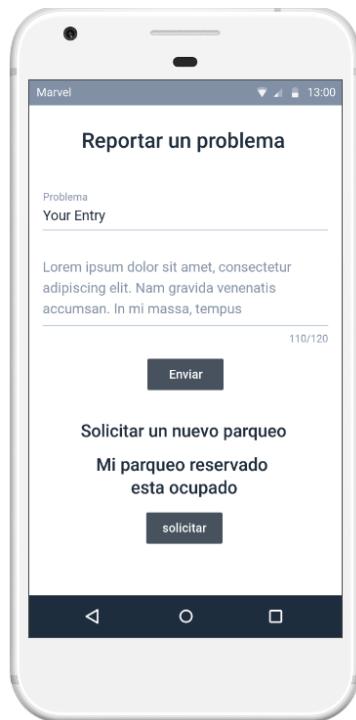


Ilustración 20 Reportar un problema

4.3.3.3 Notificaciones Push

El usuario desea configurar el sistema a su gusto

Historia de usuario

Nombre: notificaciones “Push”

Numero: 11

Usuario: Cliente

Prioridad: Alta

Programador: José Estenssoro

Descripción: La aplicación podrá informar cuando se está acabando el tiempo de reserva. Esto sirve para que el usuario pueda estar bien informado sobre su tiempo restante y pueda prevenir sus tiempos y si desea pagar por una ampliación. Para que el sistema pueda lanzar la notificación, la aplicación no necesita estar abierta. Las notificaciones van a saltar con la aplicación estando abierta o cerrada.

Validación:

- Se envían notificaciones directo desde el servidor de Firebase
- Primera parte
 - Se envían notificaciones con la aplicación cerrada
 - Las notificaciones tienen que saltar
- Segunda parte
 - Se envían notificaciones con la aplicación en segundo plano
 - Las notificaciones tienen que saltar
- Tercera parte
 - Se envían las notificaciones con la aplicación abierta.
 - Las notificaciones tienen que saltar



Ilustración 21 notificaciones Push

4.3.4 Cuarto Sprint

4.3.4.1 Menú Principal (controlador)

Como controlador, necesito un menú de navegación por el sistema

Historia de usuario

Nombre: menú principal

Numero: 12

Usuario: controlador

Prioridad: alta

Programador: José Estenssoro

Descripción: Esta Historia de usuario es muy similar a la historia de usuario numero 2 Menú Principal. El usuario podrá moverse por el sistema de manera rápida y eficiente con este menú. En el Menú Principal (controlador), tenemos 3 botones principales: El primero es “Leer QR” que direcciona a la Ilustración 23 Cámara QR. El segundo “Buscar placa” nos dirige a la Ilustración 25 Reserva por Placa (Controlador). Finalmente, tenemos un tercer botón “Realizar reserva” que nos direcciona a la Ilustración 26 reserva manual.

Validación:

- Al presionar los botones, el sistema tiene que lanzar todas las direcciones correspondientes.



Ilustración 22 Menú Principal (controlador)

4.3.4.2 Verificar reserva por QR y placas (controlador)

El controlador necesita leer los QR que le muestran los clientes para verificar si realizaron los pagos correspondientes.

Historia de usuario

Nombre: verificar reserva por QR y por placa

Numero: 13

Usuarios: Controlador

Prioridad: alta

Programador: José Estenssoro

Descripción: El usuario podrá escanear su código QR mostrando su código a la aplicación del controlador. Una vez escaneado el código, la aplicación mostrará la pantalla de la Ilustración 24 Confirmación de Reserva (Controlador). Esta pantalla nos da detalle del cliente que hizo la reserva y nos da confirmación de que se hizo el respectivo pago. Si el usuario también lo desea puede iniciar su hora introduciendo su placa tal como se muestra en la Ilustración 25 Reserva por Placa (Controlador). De la misma manera, si el sistema confirma la reserva, la aplicación mostrará en la Ilustración 24 Confirmación de Reserva (Controlador).

Validación:

Validación por código QR

- Se ingresa a la Ilustración 23 Cámara QR
- Se realiza la lectura de el QR en la Ilustración 12 Inicio de hora
- El sistema debe lanzar la Ilustración 24 Confirmación de Reserva (Controlador)
- Los datos deben coincidir con los datos de Ilustración 13 Fin de hora y con los datos del usuario.

Validación por placa

- Se ingresa a la Ilustración 25 Reserva por Placa (Controlador)
- Se ingresa el numero de la placa. Si la placa que realizo la reserva el sistema coincide, pasa a Ilustración 24 Confirmación de Reserva (Controlador). Caso contrario sale mensaje de Error.
- Si la placa es existente los datos de la Ilustración 24 Confirmación de Reserva (Controlador) deben coincidir con los datos de Ilustración 13 Fin de hora y con los datos del usuario.

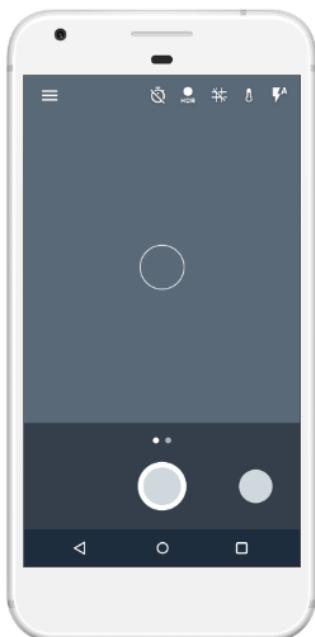


Ilustración 23 Cámara QR



Ilustración 24 Confirmación de Reserva (Controlador)



Ilustración 25 Reserva por Placa (Controlador)

4.3.4.3 Reserva manual (controlador)

El controlador necesita de un plan de respaldo ante cualquier contingencia, como la caída de Internet, para poder controlar que todo el sistema esté en orden.

Historia de usuario

Nombre: Reserva manual

Número: 14

Usuario: Controlador

Prioridad: Alta

Programador: José Estenssoro

Descripción: En casos de que exista una desconexión de los dispositivos o de que un usuario no tenga un teléfono con acceso a Internet, a reserva del parqueo se la va a realizar mediante la aplicación del controlador. Para realizar la reserva se tienen que llenar todos los campos de Ilustración 26 reserva manual, el usuario tiene que pagar el monto necesario al controlador.

Validación:

- Primera parte
 - Se llenan todos los campos
 - Se presiona el botón de aceptar.
 - El sistema realiza la reserva manual.
- Segunda parte
 - Se llenan los campos parcialmente
 - Se presiona el botón de aceptar.
 - Sale mensaje de error.

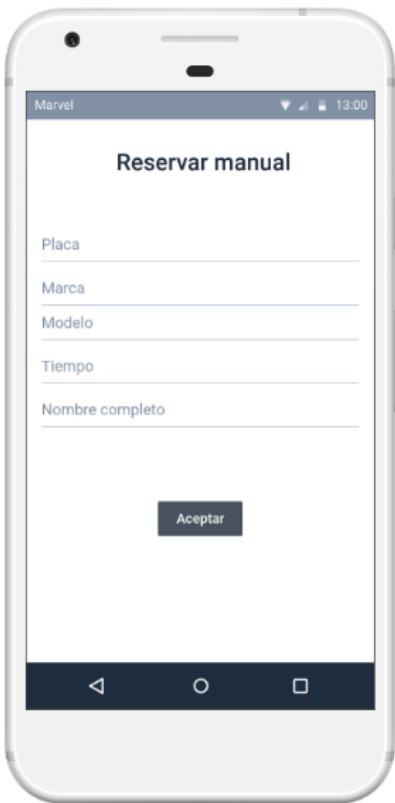


Ilustración 26 reserva manual

5 Diseño

5.1 Casos de uso

5.1.1 Caso de uso 1 Registro, Login, y autenticación de usuarios.

5.1.1.1 Descripción del caso de uso

Para que los usuarios puedan utilizar la aplicación, deben registrarse en la aplicación. Una vez registrados los usuarios podrán iniciar sesión en aplicación con su correo electrónico y su contraseña, si los usuarios ya iniciaron sesión previamente esta se guardará y no necesitarán volver a hacerlo.

5.1.1.2 Actores

- Usuarios: El usuario debe ingresar sus datos como correo electrónico licencia de conducir, contraseña entre otros en la aplicación para registrarse. Si ya están registrados en la plataforma debe ingresar su correo electrónico y contraseña.

- API de Firebase: Mediante código en Java se obtienen los datos de ingresados en la aplicación, Firebase los valida en la nube y manda una respuesta. Si el usuario ya está registrado se recopilan los datos en el cache del teléfono.

5.1.1.3 Caso de uso

Ingresar datos de inicio de sesión

- Descripción: En la pantalla de inicio de sesión el usuario ingresa los datos de su correo electrónico y su contraseña.
- Requisito: El usuario debe tener estar registrado.

Validación de datos de inicio de sesión

- Descripción: Mediante código se verifica que los campos estén llenos y después el servidor realiza la validación de los campos con su sistema de autenticación y manda una respuesta.
- Requisitos: El usuario debe ingresar los datos y enviarlos mediante un botón. También debe haber una conexión a internet estable.
- Restricciones: Firebase determina la validez de los datos

Mostrar errores de validación

- Descripción: Cuando existan errores de validación captados por la aplicación o por el servidor, la aplicación debe mostrar mensajes relacionados al error para que el usuario evalúe cómo corregirlos.
- Requisitos: Deben existir los errores para que el sistema pueda mostrarlos con mensajes.
- Restricciones: Solo se deben mostrar los mensajes de errores que se manejen en el servidor.

Ingresar datos de registro

- Descripción: En la pantalla de registro, el usuario ingresa los datos de licencia de conducir, nombre, apellido, correo electrónico y contraseña.
- Requisito: El usuario no debe estar registrado.

Validación de datos.

- Descripción: Mediante código se verifica que los campos estén llenos y después el servidor realiza la validación de los campos con su sistema de autenticación y manda una respuesta.
- Requisitos: El usuario debe ingresar a la pantalla de registro mediante un botón en la pantalla de inicio de sesión, ingresar los datos solicitados y enviarlos mediante un botón. También debe haber una conexión a internet estable.
- Restricciones: Firebase determina la validez de los datos

Mostrar errores de validación

- Descripción: Cuando existan errores de validación captados por la aplicación o por el servidor, la aplicación debe mostrar mensajes relacionados al error para que el usuario evalúe cómo corregirlos.
- Requisitos: Deben existir los errores para que el sistema pueda mostrarlos con mensajes.
- Restricciones: Solo se deben mostrar los mensajes de errores que se manejen en el servidor.

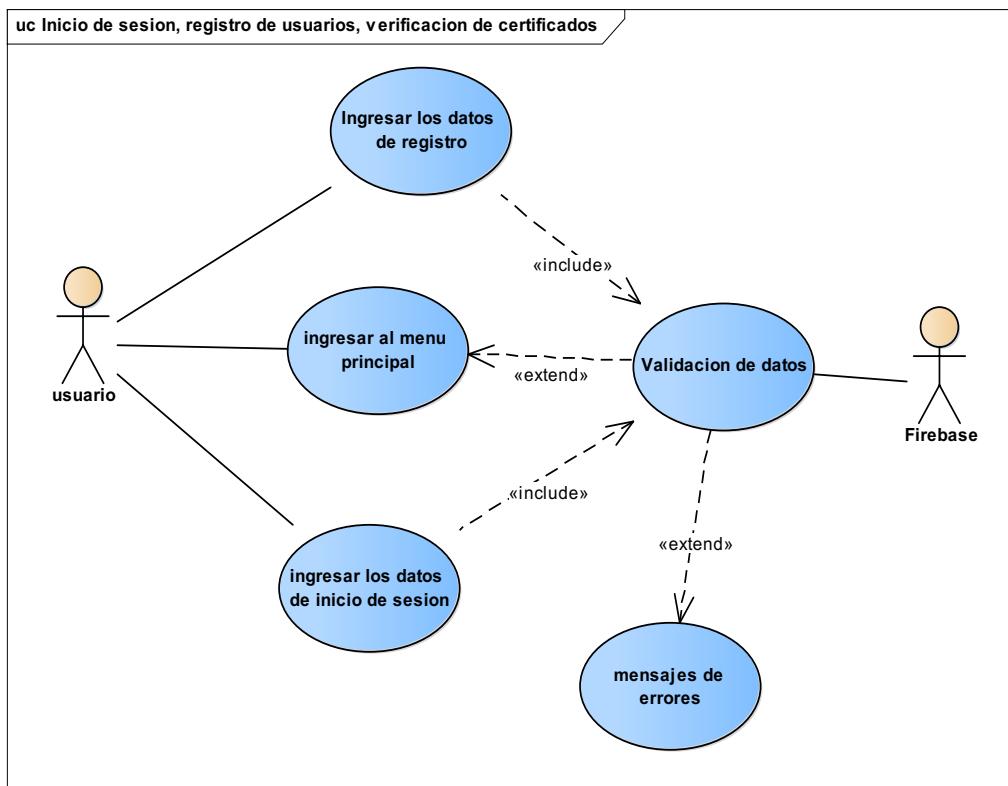
Ingresar al menú de usuario

- Descripción: Si los datos ingresados en inicio de sesión son correctos, Firebase lo confirmará al usuario y mediante código ingresará a la pantalla de usuario. Si el usuario se registra en el sistema, Firebase registrará sus datos en su sistema de autenticación y si todos los datos son válidos enviará un mensaje de confirmación. Esto se mostrará en la pantalla de menú de usuario.
- Requisitos: Debe existir una conexión a internet estable.

Validación de certificados

- Descripción: Si el usuario ya ingreso sus datos, estos son almacenados en el teléfono para que no tenga que volver a ingresarlos cada vez que usa la aplicación.
- Requisitos: Iniciar sesión por lo menos una vez en un dispositivo móvil y no cerrarla.

5.1.1.4 Diagrama caso de uso



Caso de uso 1 Inicio de sesión, registro, verificación de credenciales

5.1.1.5 Flujo del caso de uso

Caso de Uso	C.U.: Registro, Login y autenticación de usuarios	
Actores	Usuario, Firebase	
Propósito	Permitir al usuario registrarse e ingresar al sistema.	
Tipo	Primario	
CURSO NORMAL DE EVENTOS		
Acción de Actores	Respuesta del Sistema	
1 inicio sesión		
1.1 El usuario ingresa a la pantalla de 1. Inicio de sesión	1.3. La aplicación valida que se hayan llenado los campos.	

<p>1.2. Ingresa su correo electrónico y contraseña</p> <p>Flujo alterno</p> <p>1.6a. Ingresa nuevamente sus datos o los corrige.</p> <p>2. Registro de usuarios</p> <p>2.1 El usuario ingresa a la pantalla de registro</p> <p>2.2 El usuario ingresa todos los campos requeridos para realizar el registro</p> <p>Flujo alterno</p> <p>2.6 Ingresar nuevos datos</p> <p>3. Verificación de certificados</p>	<p>1.4. Los datos son enviados a Firebase Authentication mediante la API para ser validados.</p> <p>1.5. La plataforma valida los datos y envía la validez de los datos a la aplicación</p> <p>1.6. Se ingresa al menú de usuarios.</p> <p>1.4a. Los datos ingresados por el usuario son incorrectos</p> <p>1.5a. Se muestra un mensaje correspondiente al error del usuario</p> <p>2.3 La aplicación valida que todos los campos sean llenados.</p> <p>2.4 Los datos son enviados a Firebase Authentication y a la base de datos en tiempo real para ser validados y registrados.</p> <p>2.5 Firebase envía una confirmación de datos</p> <p>2.6 Se ingresa al menú principal</p> <p>2.4a. Los datos ingresados ya existen y no se puede registrar al usuario.</p> <p>2.5a Se muestra mensaje de error al usuario</p> <p>3.2 La aplicación verifica sus datos</p>
--	--

3.1 El usuario ingresa a la aplicación y ya inicio sesión en el pasado	3.3 Ingresa al menú principal
--	-------------------------------

5.1.2 Caso de uso 2: Registro de vehículos

5.1.2.1 Descripción del caso de uso

Los usuarios van a poder registrar sus vehículos en la base de datos del sistema para facilitar los controles y el registro de los mismos.

5.1.2.2 Actores

- Usuarios: El usuario debe ingresar en los campos correspondientes los datos de su vehículo como placa, marca y modelo
- API de Firebase: Mediante el código en Java se obtienen todos los datos, para después procesarlos y almacenarlos en la base de datos.

5.1.2.3 Caso de uso

Ingresar datos Vehículo

Descripción: En la pantalla de registro de vehículos se ingresan todos los datos del vehículo.

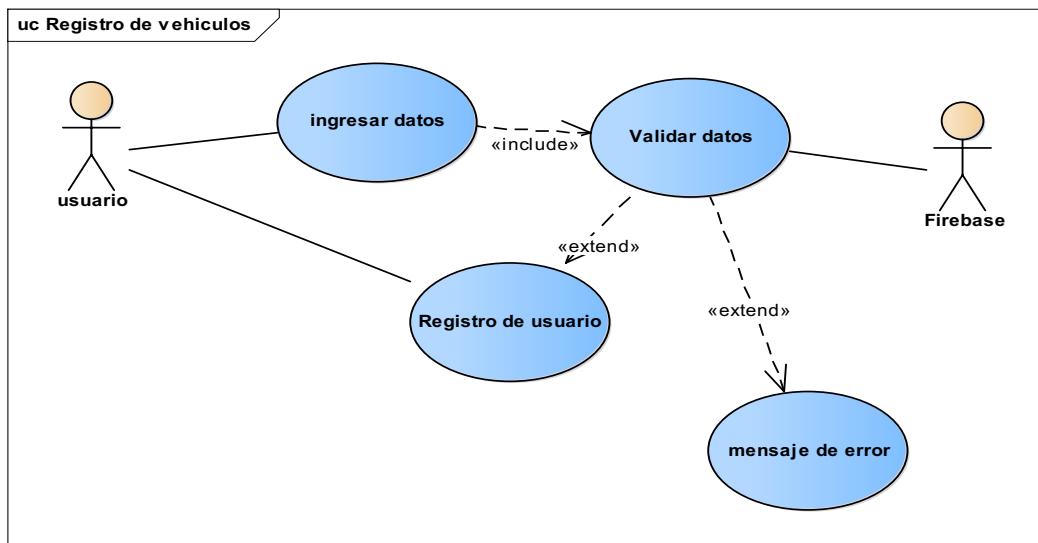
Requisito: El usuario tiene que tener la sesión iniciada.

Validación de datos vehículos

Descripción: Mediante código Java podemos verificar todos los campos.

Requisito: El usuario debe entrar a la pantalla mediante un botón de navegación en el menú principal, ingresar los datos y enviarlos mediante otro botón. Se debe de tener una conexión a internet estable.

5.1.2.4 Diagrama de caso de uso



Caso de uso 2 Registro de vehículos

5.1.2.5 Flujo del caso de uso

Caso de Uso	C.U.: Registro vehículos.
Actores	Usuario, Firebase
Propósito	Permitir registrar a sus vehículos.
Tipo	Primario
CURSO NORMAL DE EVENTOS	
Acción de Actores	Respuesta del Sistema
1. El usuario ingresa a la pantalla de Registro de vehículos.	3. La aplicación valida que se hayan llenado los campos.
2. Ingresar datos del vehículo	4. Los datos son enviados a Firebase Real Time Database mediante la API para ser validados. 5. La plataforma valida los datos y envía la validez de los datos a la aplicación 6. Se registra el vehículo

<p>Flujo alterno</p> <p>6a. Ingrsesa nuevamente sus datos o los corrige.</p>	<p>4a. Los datos ingresados por el usuario no son validos</p> <p>5a. Se muestra un mensaje correspondiente al error del usuario</p>
---	---

5.1.3 Búsqueda, reserva y pago de estacionamiento disponible

5.1.3.1 Descripción del caso de uso

Los usuarios van a poder reservar estacionamientos desde el mapa que se ofrece. Para que los usuarios puedan reservar deben por lo menos tener un vehículo presente además de dinero cargado a la cuenta.

5.1.3.2 Actores

- Usuarios: El usuario puede elegir el estacionamiento más cercano o conveniente.
- Api Firebase: Proporciona todos los datos necesarios para la reserva, además permite almacenar los datos de la reserva
- Api Google Maps: Facilita el mapa en el teléfono y proporciona todos objetos del mapa.

5.1.3.3 Caso de uso

Seleccionar estacionamiento

Descripción: El usuario va a poder seleccionar un estacionamiento desde el mapa, solo se desplegarán los estacionamientos disponibles.

Requisitos: El usuario debe tener una conexión a internet,

Restricciones: El usuario no puede tener un parque ya reservado

Seleccionar tiempo y vehículos

Descripción: El usuario tiene que seleccionar el tiempo de estadía y el vehículo que va a estar estacionado en el estacionamiento.

Requisitos: El usuario debe tener una conexión a Internet, el usuario ya tiene que seleccionar un parqueo disponible.

Restricciones: El usuario no puede tener un parque ya reservado.

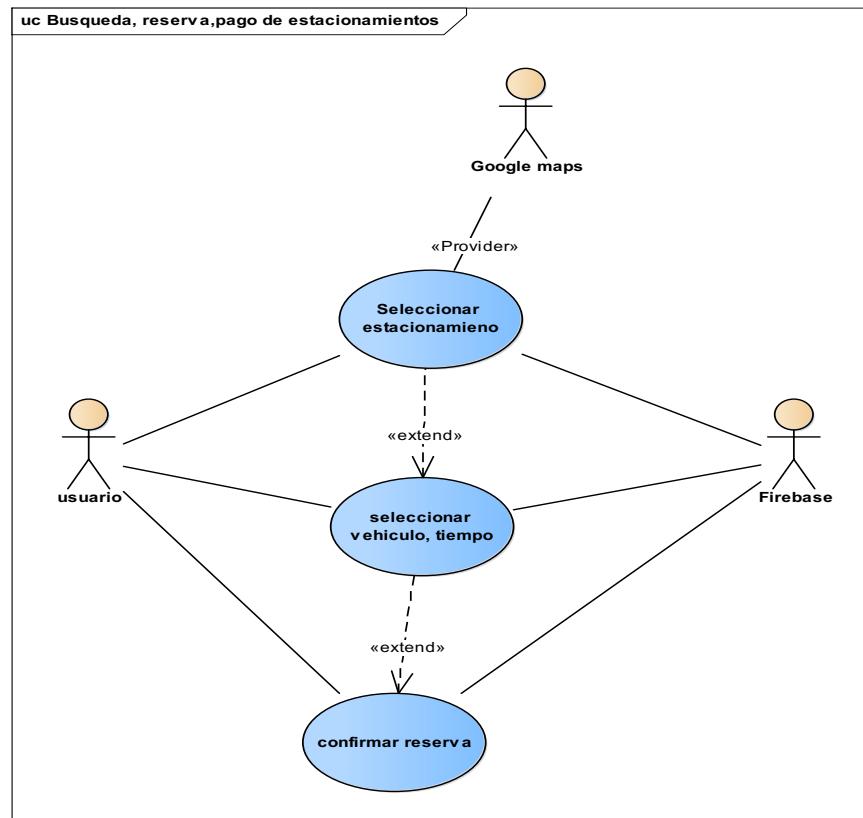
Confirmar reserva y pagar

Descripción: El usuario tiene que pagar el estacionamiento con el saldo que el usuario ya cargo.

Requisitos: El usuario debe tener una conexión a Internet, el usuario ya tiene que seleccionar un parqueo disponible, el usuario tiene que tener el saldo necesario para pagar.

Restricciones: el usuario no puede tener un parque ya reservado.

5.1.3.4 Diagrama de caso de uso



Caso de uso 3 Búsqueda, reserva y pago de estacionamientos

5.1.3.5 Flujo del caso de uso

Caso de Uso	C.U.: Búsqueda, Reserva y pago de estacionamientos
Actores	Usuario, Firebase, Google Maps
Propósito	Permitir al usuario registrarse e ingresar al sistema.
Tipo	Primario
CURSO NORMAL DE EVENTOS	
Acción de Actores	Respuesta del Sistema
1 El usuario ingresa a la pantalla de búsqueda de estacionamiento 3. El usuario selecciona el parque de su conveniencia 5 El usuario selecciona el tiempo de estadía y su automóvil. 6. El usuario confirma la reserva Flujo alterno 8a. cancela operación para cargar crédito.	2. La aplicación con Google Maps despliegan el mapa con todos los parqueos disponibles 4. La aplicación despliega la pantalla para seleccionar el tiempo y el vehículo de reserva 6. La plataforma despliega la pantalla de confirmación con el precio y los datos de pago. 7. La plataforma hace los cálculos requeridos actualiza los datos en la base de datos. 8. se envía mensaje de confirmación. 7a. El usuario no tiene saldo suficiente 8a. Se muestra un mensaje correspondiente al error del usuario

5.1.4 Inicio y fin de Hora

5.1.4.1 Descripción del caso de uso

Cuando el usuario logre estacionar su automóvil, es necesario que pueda informar al sistema que llegó a su puesto para que su tiempo comience a correr. También es necesario que se informe sobre la salida del mismo para que se pueda habilitar el estacionamiento para futuros clientes.

5.1.4.2 Actores

- Usuario: El usuario necesita que la aplicación de controlador lea el código QR que se le proporciona al llegar y salir de su puesto reservado.
- Api Firebase: Mediante consultas a la base de datos en tiempo real se verificará si la reserva es válida o no.
- Aplicación del controlador: Lee el código QR mostrado por el cliente para validar su llegada y que su tiempo comience a transcurrir.

5.1.4.3 Caso de uso

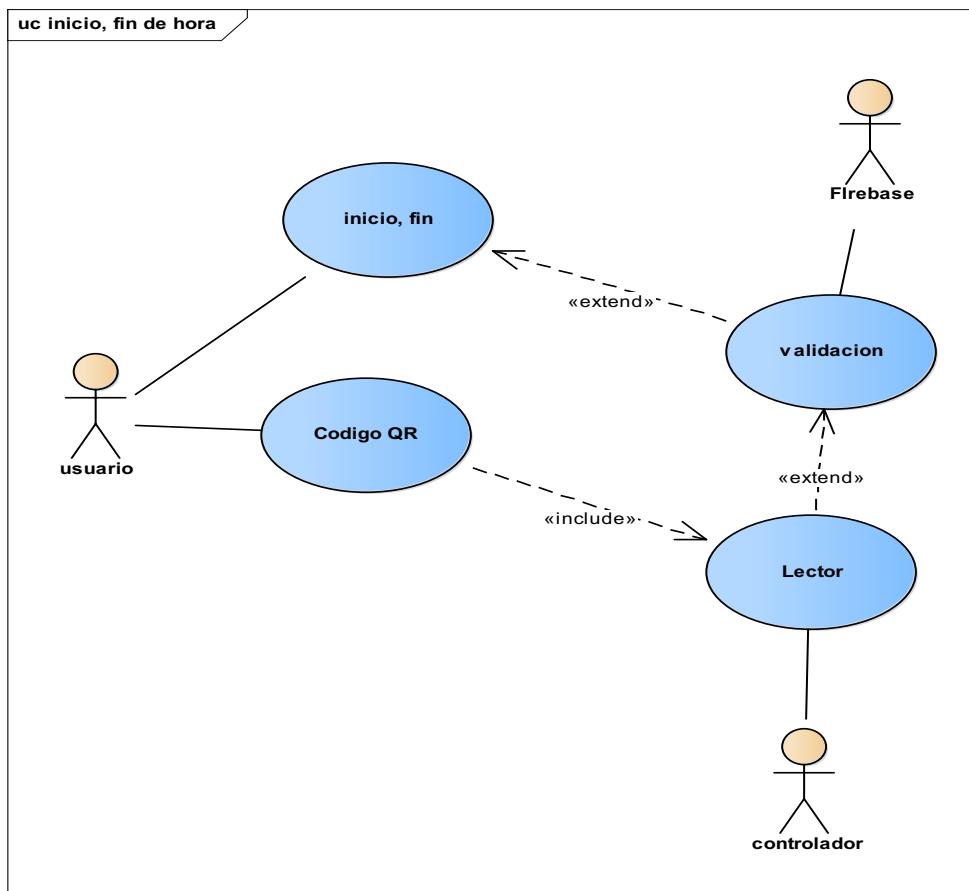
Inicio de Hora y Fin de Hora

Descripción: El usuario tiene que entrar a la aplicación, ir a la sección donde está su código QR y escanearlo en el lector para que su hora comience.

Requisitos: El usuario debe tener una conexión a internet, el usuario ya tiene que tener reservado y pagado un estacionamiento.

Restricciones: Desde la confirmación del estacionamiento, el usuario tiene 15 minutos para llegar si no se le cancela su hora.

5.1.4.4 Diagrama de caso de uso



Caso de uso 4 Inicio, fin de hora

5.1.4.5 Flujo del caso de uso

Caso de Uso	C.U.: Inicio, fin de Hora
Actores	Usuario, Firebase, controlador
Propósito	Poder controlar los tiempos de llegada y salida de los usuarios.
Tipo	Primario
CURSO NORMAL DE EVENTOS	
Acción de Actores	Respuesta del Sistema

	<p>1 El usuario ingresa a la pantalla del QR</p> <p>2. El usuario selecciona el parque de su conveniencia</p> <p>Flujo alterno</p>	<p>2. La aplicación del controlador la lee el Código</p> <p>3. La aplicación realiza consultas a la base de datos.</p> <p>4. La plataforma despliega la pantalla de confirmación con los datos de la reserva, dando inicio a su hora, o el fin a la hora.</p> <p>4a. El usuario nunca hizo la reserva.</p> <p>8a. Se muestra un mensaje correspondiente al error del usuario</p>
--	---	--

5.1.5 Carga de crédito

5.1.5.1 Descripción del caso de uso

Para que los usuarios puedan pagar los estacionamientos es necesario tener una plataforma de pago fácil de usar y eficiente.

5.1.5.2 Actores

- Usuario: El usuario va a poder realizar cargas de crédito a su cuenta de manera rápida para poder facilitar el manejo del dinero dentro de la aplicación.
- Api de Firebase: Firebase permite almacenar todos los movimientos de dinero que se realicen.
- Api de Khipu: Esta plataforma de pago no permite realizar los pagos de dinero de manera rápida y eficiente este es un método de pago boliviano.

5.1.5.3 Caso de uso

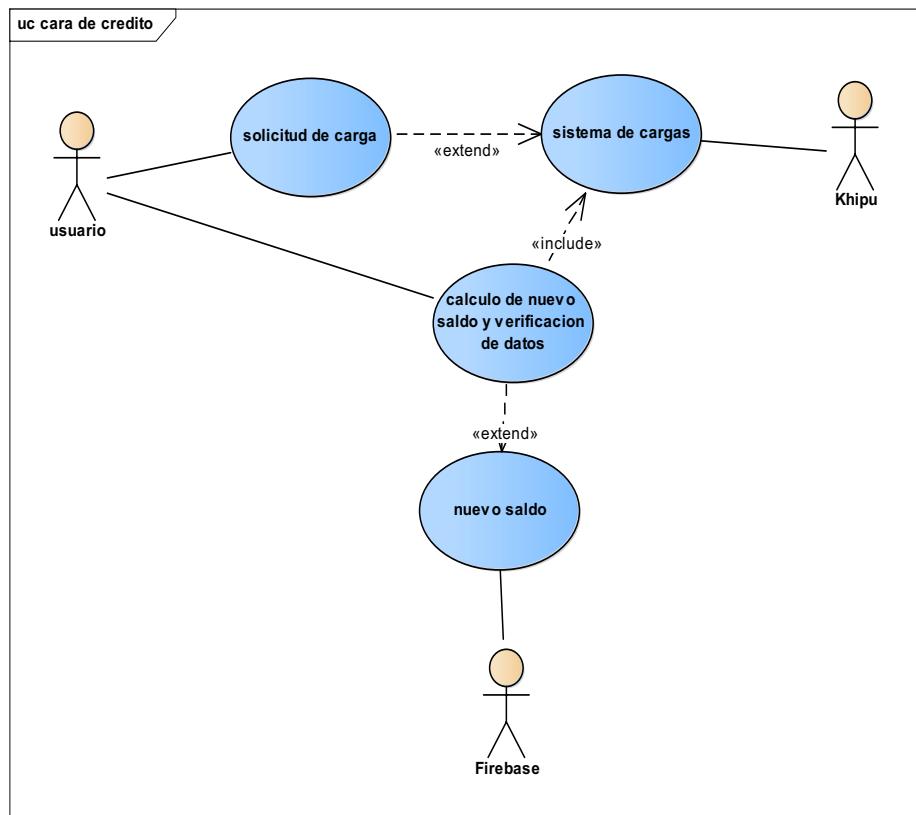
Solicitud de carga

Descripción: El usuario tiene que entrar a la aplicación, ir a la sección de crédito e ingresar al sistema de Khipu, después se tienen que seguir los pasos requeridos por el sistema.

Requisitos: El usuario debe tener una conexión a internet, debe de tener una tarjeta de débito o crédito.

Restricciones: Solo usuarios autenticados pueden realizar esta operación

5.1.5.4 Diagrama de caso de uso



Caso de uso 5 carga de crédito

5.1.5.5 Flujo del caso de uso

Caso de Uso	C.U.: Carga de crédito
Actores	Usuario, Firebase, Khipu
Propósito	Manejar dinero y poder pagar los estacionamientos.
Tipo	Primario
CURSO NORMAL DE EVENTOS	
Acción de Actores	Respuesta del Sistema
<p>1 El usuario ingresa a la pantalla de crédito y dirigiese al sistema Khipu</p> <p>3. El usuario debe llenar los datos de correo electrónico, monto de carga y el de la tarjeta de crédito.</p> <p>5. Al finalizar el usuario debe de salir del sistema y</p>	<p>2. La aplicación desplegará el sistema de cobros Khipu</p> <p>4 El sistema de cobros validara estos datos y realizara el cobro.</p> <p>6. El sistema realizará los cálculos de saldos</p>
Flujo alterno	<p>4a. El usuario no pudo realizar la carga.</p> <p>5a. Se muestra un mensaje correspondiente al error del usuario</p>

5.1.6 Facturación

5.1.6.1 Descripción del caso de uso

Las empresas tanto privadas como públicas tienen que pagar impuestos por los ingresos que reciban. Y tienen que reportar todas sus ganancias con la entrega de facturas a todas las personas de las que reciban dinero.

5.1.6.2 Actores

- Usuario: El usuario puede generar facturas sobre las cargas de crédito que realizó y se las puede descargar al teléfono.
- Api de Firebase: Firebase permite poder realizar las consultas de todas las cargas de crédito que se realizaron.

5.1.6.3 Caso de uso

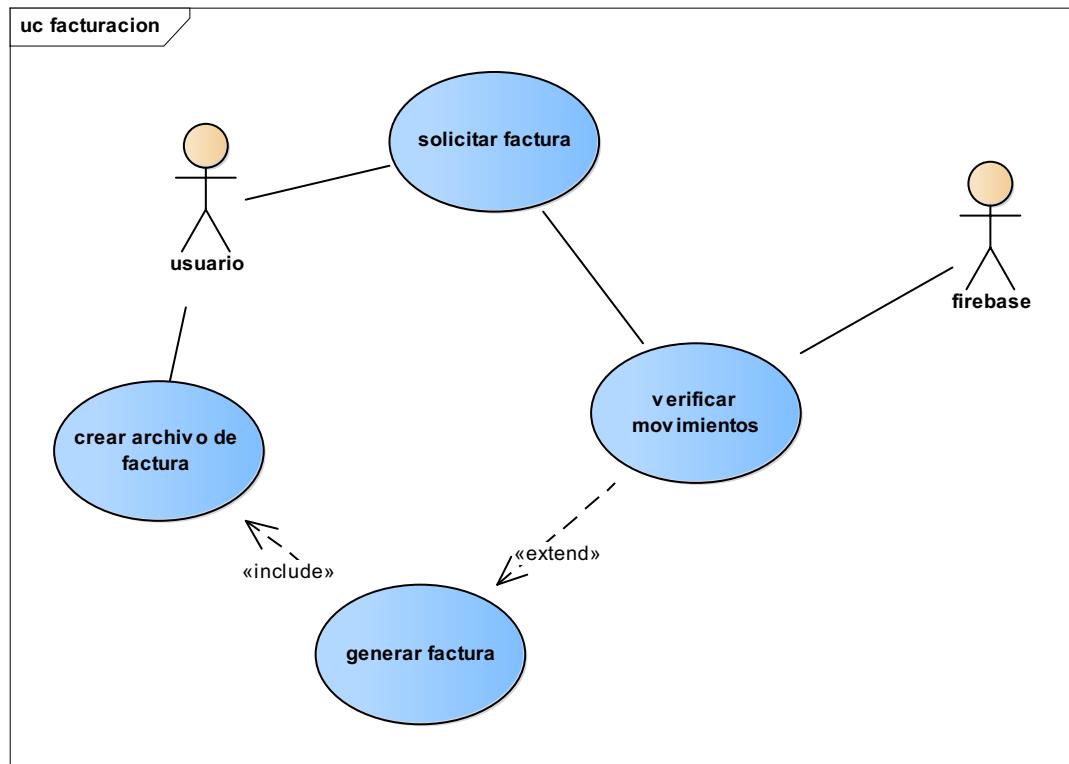
Facturación

Descripción: El usuario tiene que entrar a la aplicación, ir a la sección de facturación, donde se desplegará una vista de todas las operaciones que hizo, selecciona una y envía los datos de la misma apretando el botón que ordena la generación de la factura.

Requisitos: El usuario debe tener una conexión a Internet y tiene que ingresar los datos de facturación.

Restricciones: El usuario tiene que haber realizado por lo menos una (1) carga

5.1.6.4 Diagrama de caso de uso



Caso de uso 6 Facturación

5.1.6.5 Flujo del caso de uso

Caso de Uso	C.U.: Facturación
Actores	Usuario, Firebase
Propósito	Crear facturas para impuestos nacionales.
Tipo	Primario
CURSO NORMAL DE EVENTOS	
Acción de Actores	Respuesta del Sistema

<p>1 El usuario ingresa a la pantalla donde se registran todos los gastos y seleccione el que deseé tener la factura.</p> <p>3. El usuario debe llenar los datos de Razón Social y Nit</p> <p>4. Después se presiona el botón de generar</p> <p>6 El usuario puede enviar su factura a su correo electrónico</p>	<p>2. La aplicación desplegará la pantalla con los principales datos de la factura y solicitará al usuario que ingrese los datos de razón social el NIT o el carnet de identidad</p> <p>5 El código del sistema genera la factura en pdf.</p>
--	---

5.2 Diagrama de comunicación

El usuario se comunica directamente con la pantalla del mapa. El diseño y la estructura del mapa son proporcionados por la api de Google Maps. Para obtener las coordenadas del sistema se realiza una comunicación directa con la base de datos de Firebase. Finalmente, para realizar la búsqueda, reserva y pago de los estacionamientos el sistema realiza carga y lee los datos a la base de datos directamente.

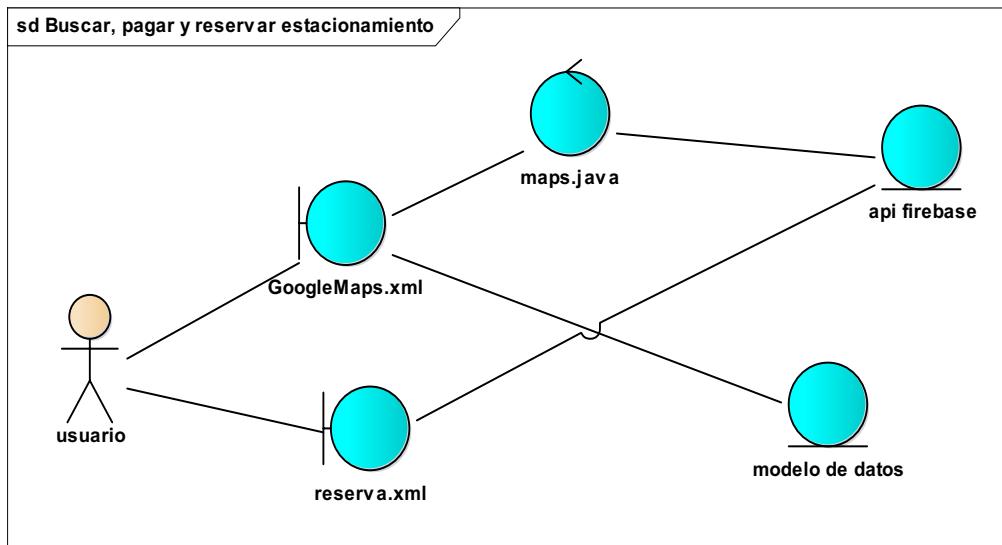


Diagrama de comunicación 1 comunicación 1

5.3 Diagramas de secuencia

Iniciar sesión

En el Diagrama de secuencias 1 Iniciar se establece el proceso que la aplicación realiza cuando un usuario quiere iniciar sesión en el sistema. El usuario primero tiene que ingresar su correo electrónico y contraseña. Se validan los campos y mediante código Java se mandan los datos al sistema de autenticación de Firebase. Si el usuario y la contraseña son correctas, Firebase manda una respuesta mediante un Snapshot, indicando que el usuario y la contraseña son validadas. Además en el Snapshot se obtiene el UID (User ID), una vez verificados los datos la aplicación muestra la pantalla de menú principal.

Si el usuario no existe en la base de datos o ingreso mal la contraseña, Firebase manda un mensaje de error.

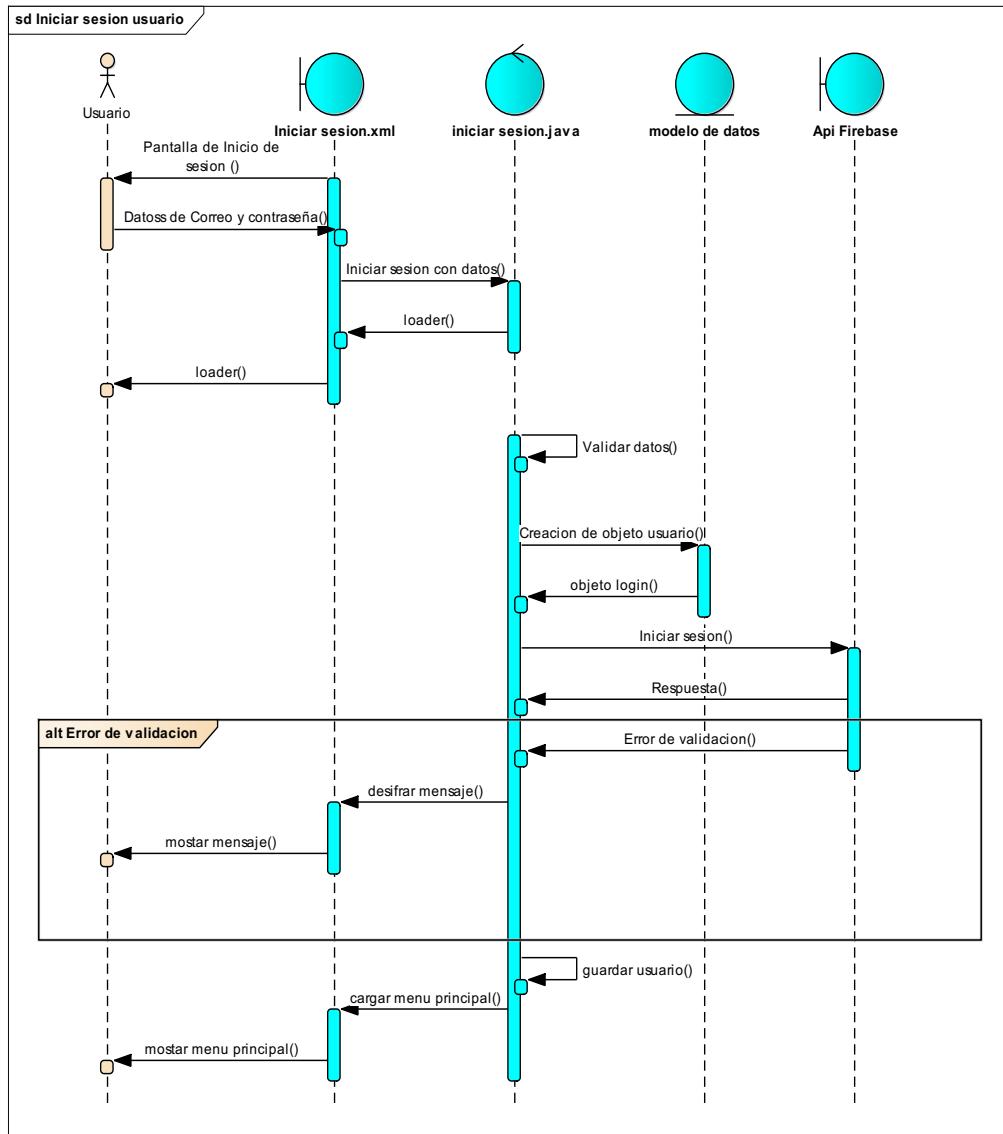


Diagrama de secuencias 1 Iniciar sesión

Registrar usuario

En el Diagrama de secuencias 2 Registro de usuarios se ve el proceso que la aplicación móvil realiza cuando los usuarios se registran en el sistema. El usuario tiene que ingresar sus datos personales como licencia de conducir, correo electrónico, nombre

completo, contraseña, etc., se validan los campos y mediante código Java se los envía a Firebase para verificar si el usuario ya estaba previamente registrado en el sistema de autenticación de Firebase. Si el usuario no está registrado se almacenan sus datos en la base de datos.

Si el sistema de Firebase encuentra un error, manda un mensaje de error.

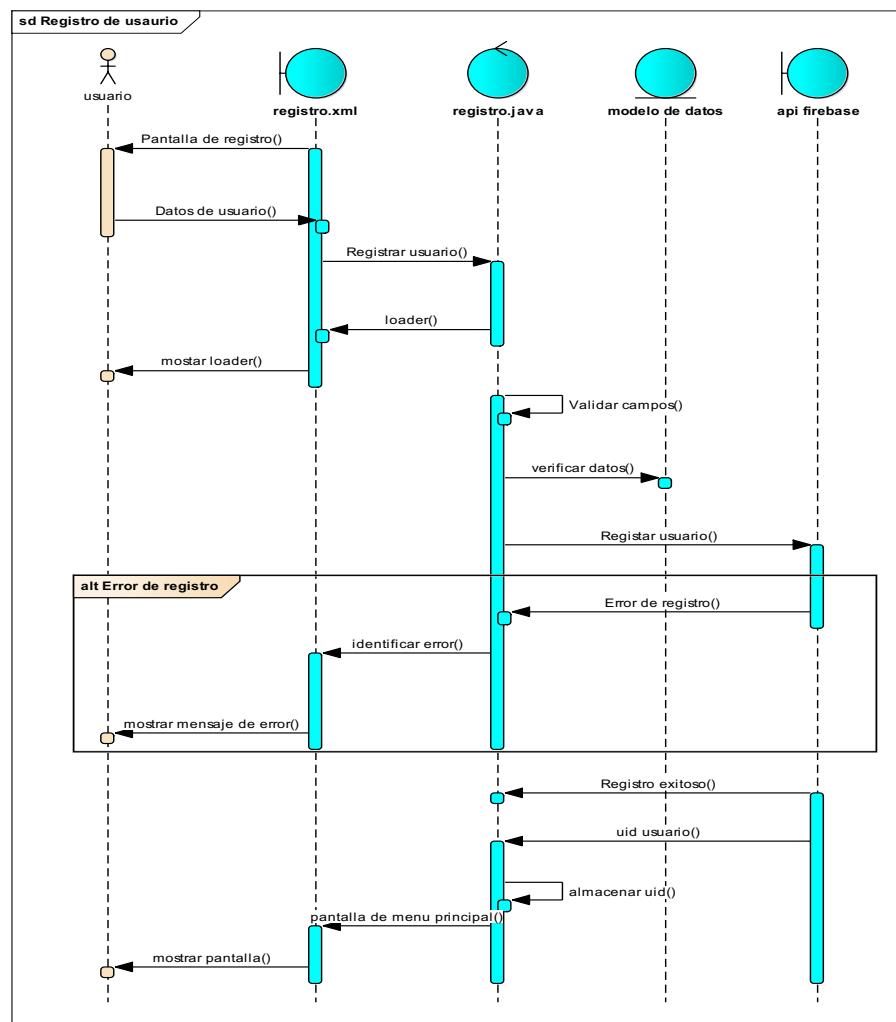


Diagrama de secuencias 2 Registro de usuarios

Registro de vehículos

En el “Diagrama de secuencias 3 Registro de vehículos” se muestra la secuencia que realiza el sistema cuando los usuarios registran uno o más vehículos. El usuario debe ingresar el número de la placa, marca y modelo del vehículo. El sistema mediante código Java valida los datos y son almacenados en la Base de datos en tiempo real de Firebase.

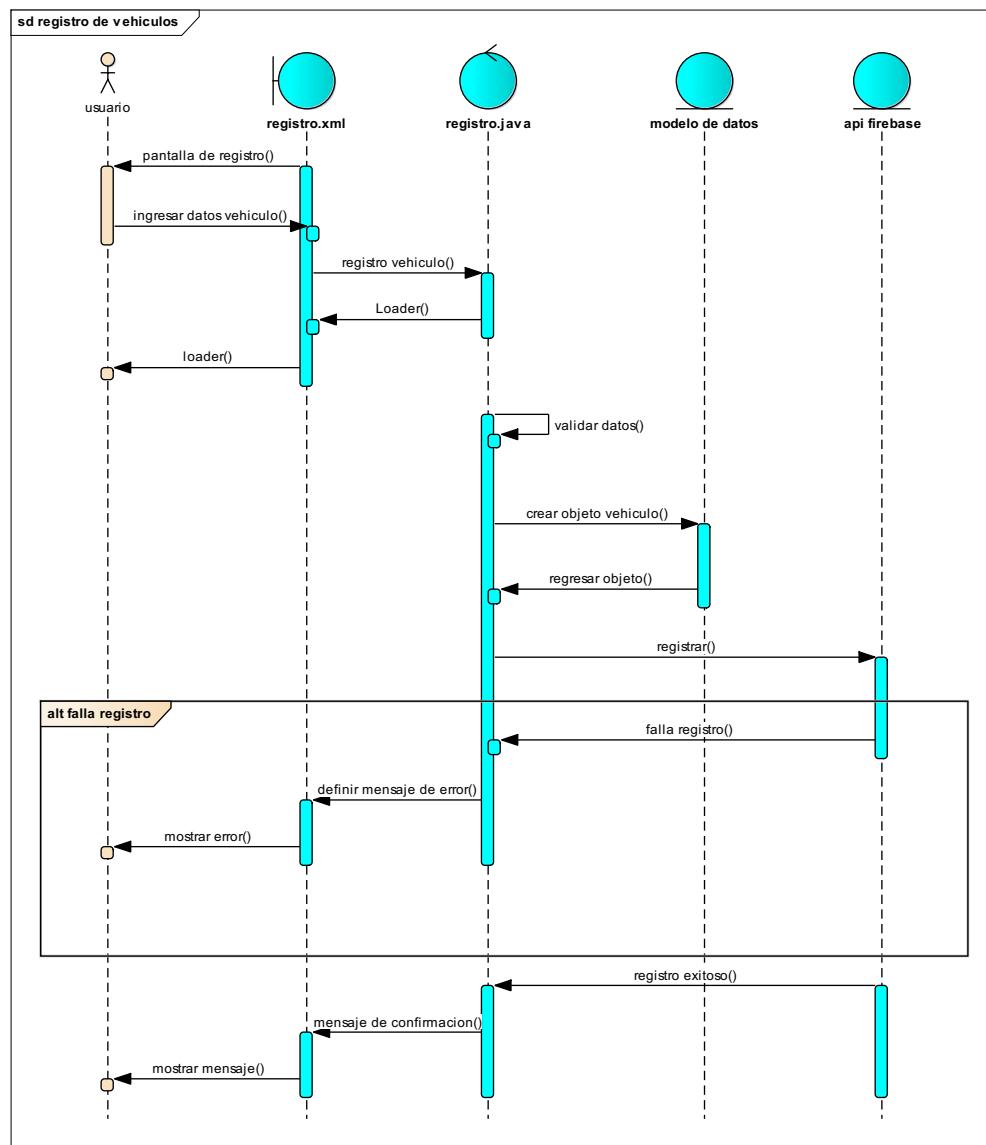


Diagrama de secuencias 3 Registro de vehículos

Búsqueda, reserva y pago de estacionamientos.

En el “Diagrama de secuencias 4 Búsqueda, reserva y pago de estacionamientos” se establece el procedimiento que el sistema utiliza cuando los usuarios necesiten realizar las reservas y pagos de los estacionamientos. Cuando el usuario ingrese a la sección del mapa, la aplicación se conecta con la api de Google Maps. Esto permite mostrar en pantalla el mapa de la ciudad para que los usuarios puedan buscar un estacionamiento mas fácil y rápido. Mediante Código Java se realiza una solicitud a la base de datos de Firebase para obtener las coordenadas de los estacionamientos disponibles y toda la información de los mismos para poder mostrarlos en el mapa. Una vez que los

estacionamientos disponibles se los muestra en el mapa, el usuario hace click sobre uno, la aplicación cambia de pantalla para que el usuario pueda seleccionar el vehículo que desea utilizar. Para finalizar se muestra una ultima pantalla donde se calcula el costo de la reserva en código Java, el usuario acepta la reserva y paga.

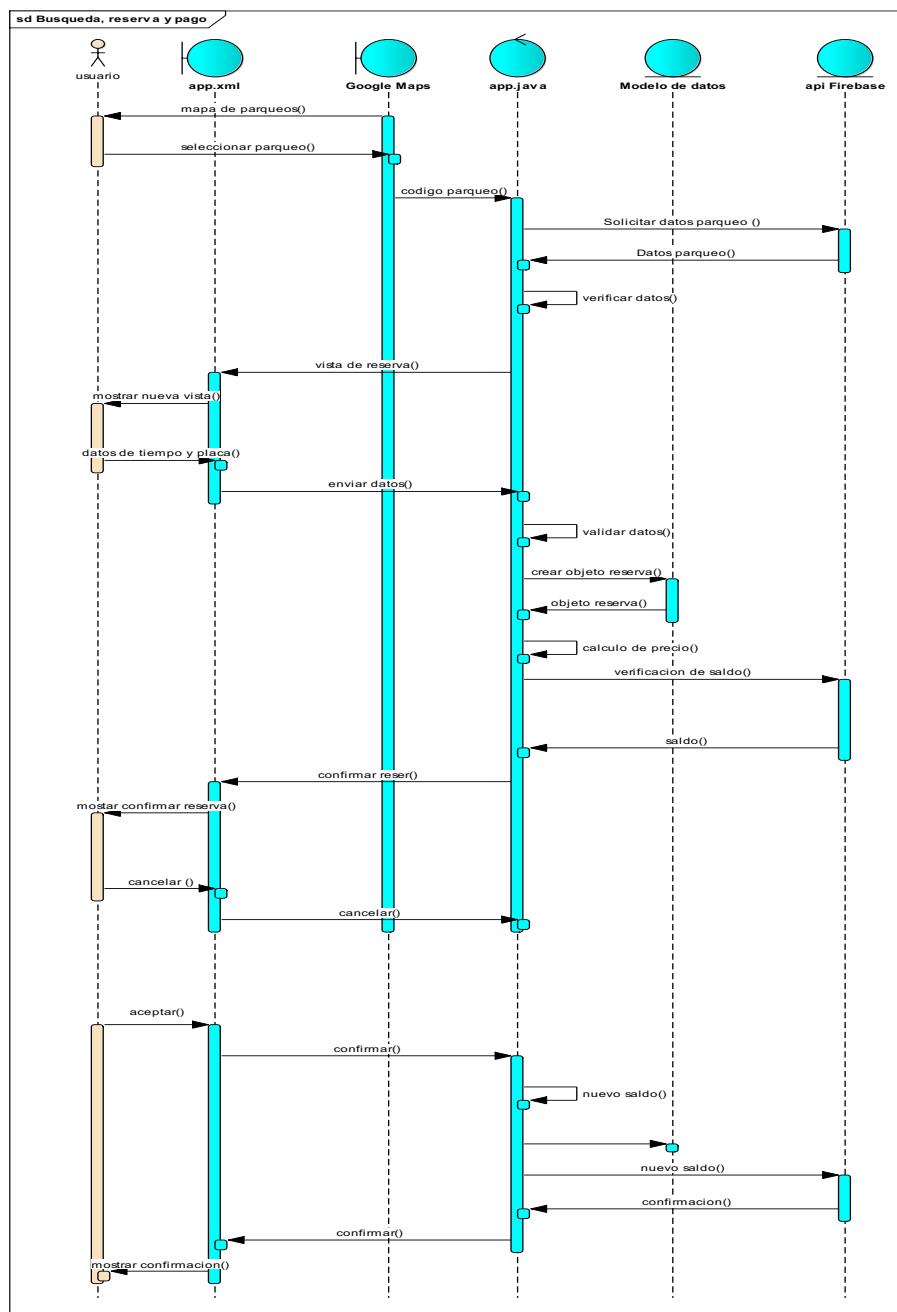


Diagrama de secuencias 4 Búsqueda, reserva y pago de estacionamientos

Inicio y fin de hora

En el “Diagrama de secuencias 5 Inicio, fin de Hora” se ve el procedimiento de la aplicación cuando un usuario solicita realizar el escaneo de su código QR. Al ingresar a la pantalla donde se confirma la reserva se genera un código QR con el UID del usuario.

En la aplicación controlador se habilita el permiso para ejecutar la cámara del dispositivo. Cuando la cámara esta mostrando un código QR con Código Java se lo lee y se obtienen los datos del mismo. Se verifica si los datos que contiene el código QR son validos. Si los datos son validos se consulta a la base de datos si el usuario realizo una reserva. Si la reserva es válida, se ejecuta una pantalla color verde confirmando la reserva con los datos mas importantes de las misma. Este servicio funciona tanto para inicio de hora como para el fin.

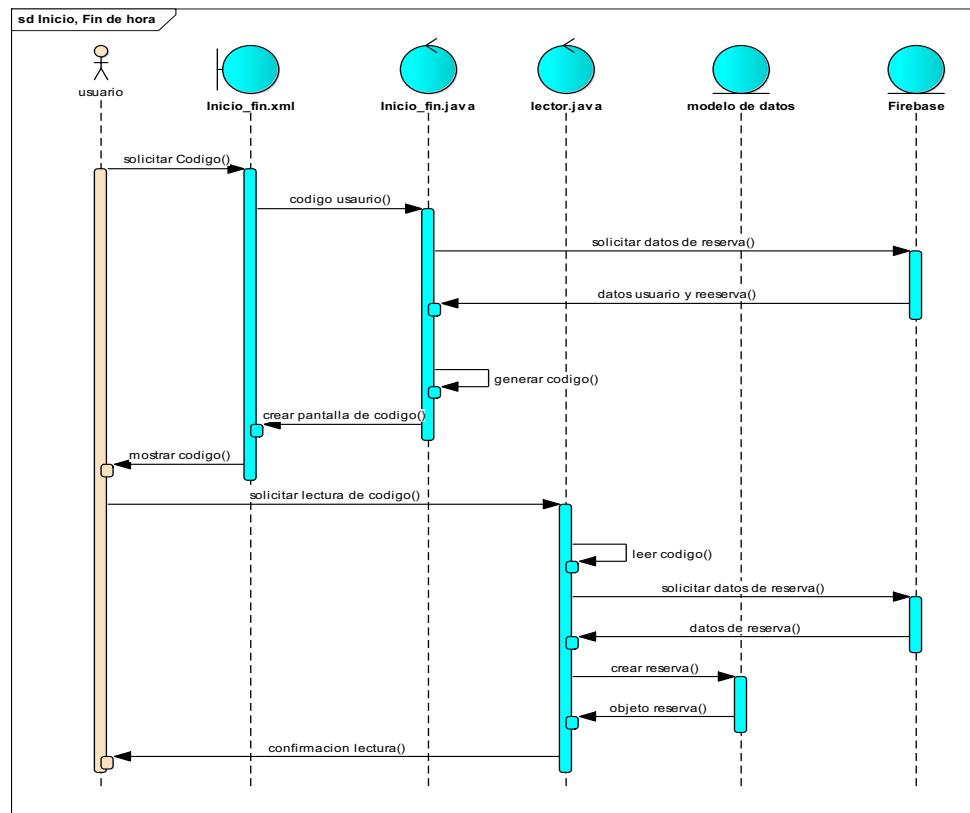


Diagrama de secuencias 5 Inicio, fin de Hora

Carga de Crédito

En el “Diagrama de secuencias 6 Carga de crédito” se establecen los procesos utilizados cuando un usuario utiliza la función de “cargar crédito” en la aplicación. La mayor parte de este proceso es realizado por la plataforma de pago.

Al entrar al sistema el usuario debe ingresar el monto que desea cargar, la plataforma verifica el monto y solicita un correo electrónico y verifica si el mismo es valido, después el usuario debe ingresar los datos de su tarjeta de crédito o debito y el sistema realiza el cobro.

El monto es verificado por la aplicación y es actualizado en la base de datos.

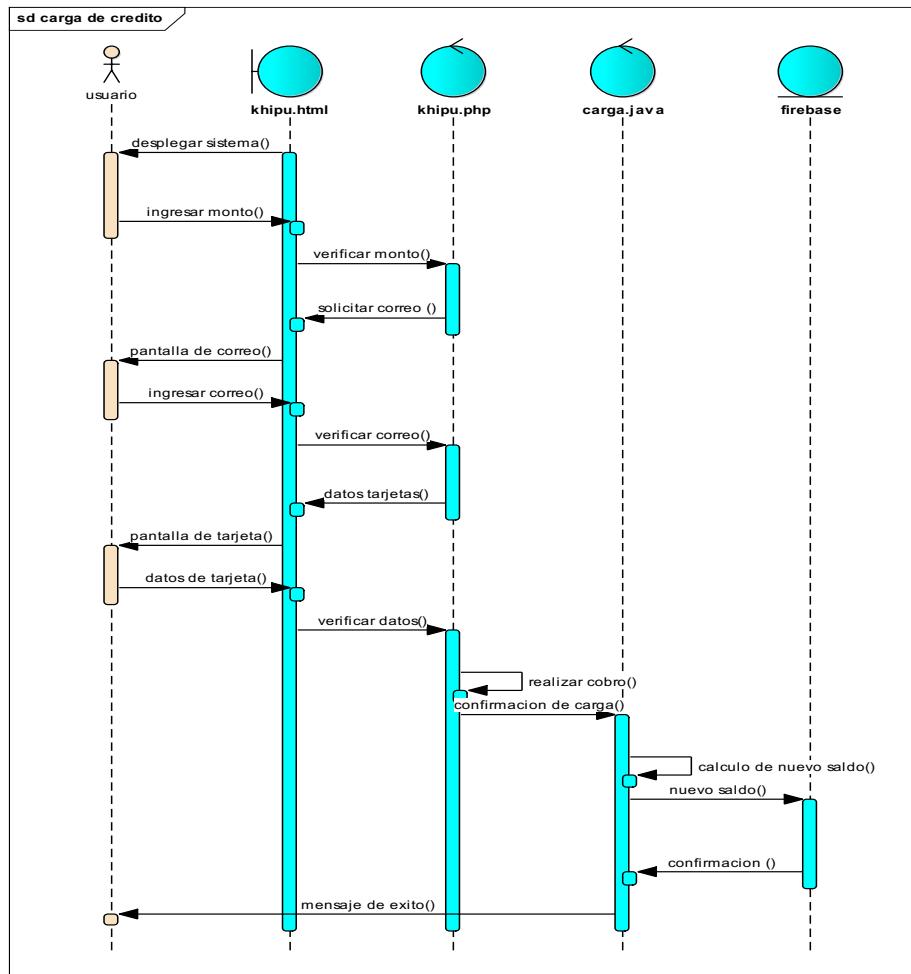


Diagrama de secuencias 6 Carga de crédito

Facturación

El “Diagrama” muestra el proceso que la aplicación sigue para la generación de facturas. Al ingresar a la sección de facturación, el sistema solicita el Nit y el nombre para la factura. Mediante código Java se consulta la base de datos para obtener la información necesaria sobre la factura. Con código se genera un documento en PDF con todos los datos de la factura y se lo guarda en el dispositivo.

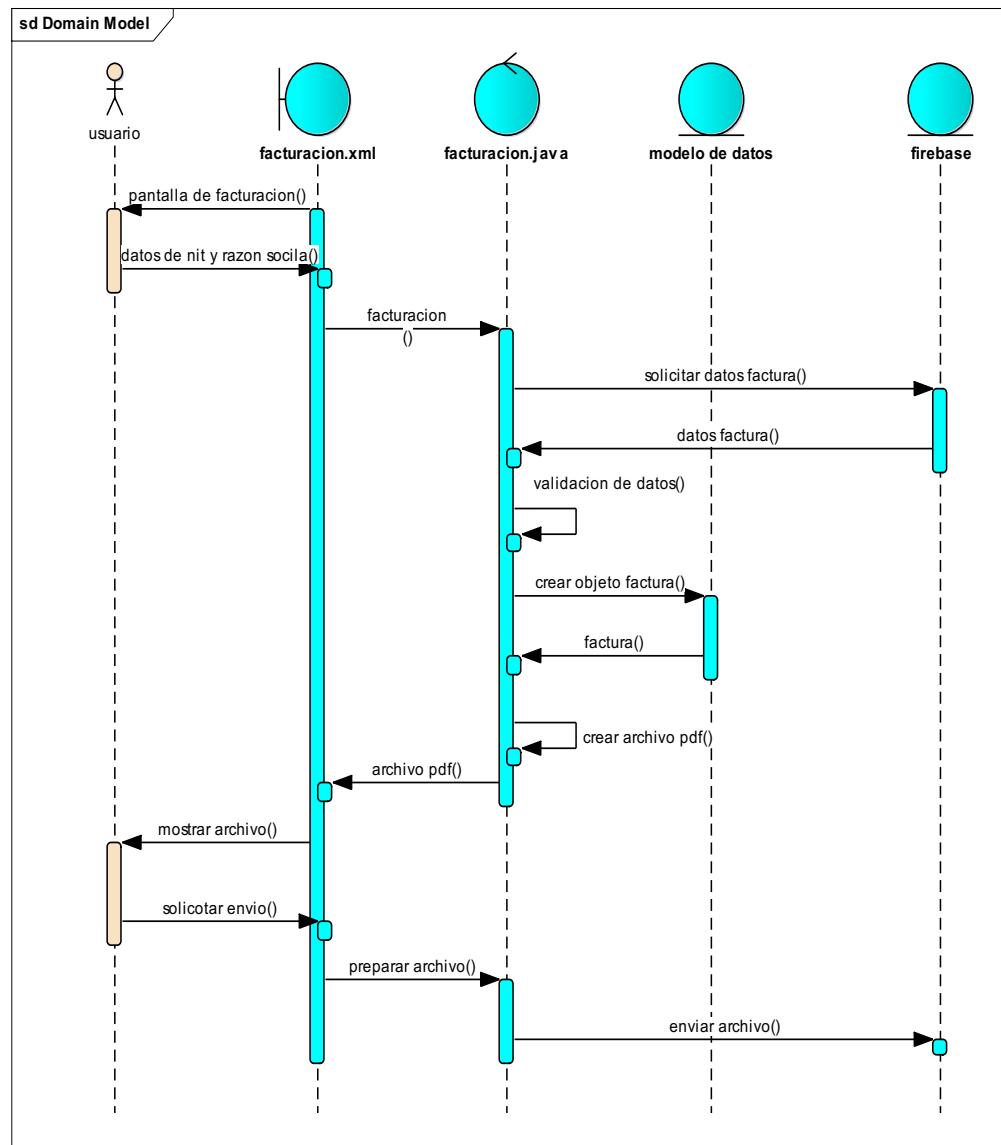


Diagrama de secuencias 7 Facturación

6 Estructura del prototipo

6.1 Base de datos en tiempo real

6.1.1 Configuración de la base de datos al proyecto

Para integrar la base de datos en la aplicación se deben agregar las líneas de código que se ven en la siguiente ilustración (archivo “Build.gradle” de la aplicación):

```
dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebaseio:firebase-bom:26.1.1')

    // Declare the dependency for the Realtime Database library
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebaseio:firebase-database'
}
```

Ilustración 27 Agregar Base de datos

Nota: Ilustración 27 Agregar Base de datos imagen obtenida de la documentación oficial de Firebase (Firebase, 2020).

Después se configurarán las reglas de escritura y lectura de la base de datos.

En la “Ilustración 28 Regla de escritura y lectura falso”, se indica que en la base de datos del proyecto ningún usuario puede escribir o leer en ningún nodo de toda la base de datos (Firebase, 2020).

```
{
  "rules": {
    ".read": false,
    ".write": false
  }
}
```

Ilustración 28 Regla de escritura y lectura falso

Nota: Ilustración 28 Regla de escritura y lectura falso, imagen obtenida de la documentación oficial de Firebase (Firebase, 2020).

En la “Ilustración 29 Solo usuarios autenticados pueden leer y escribir” se indica que únicamente los usuarios que realizaron su autenticación en el servicio de autenticación de Firebase son los que pueden realizar movimientos en las bases de datos (Firebase, 2020).

```
{  
  "rules": {  
    ".read": "auth.uid != null"  
    ".write": "auth.uid != null"  
  }  
}
```

Ilustración 29 Solo usuarios autenticados pueden leer y escribir

Nota: Ilustración 29 Solo usuarios autenticados pueden leer y escribir, Reglas de Firebase, imagen obtenida de la documentación oficial de Firebase (Firebase, 2020).

En la “Ilustración 30 Dueños de contenido” se indica que los únicos usuarios que pueden realizar las operaciones de lectura o escritura en los nodos de las bases de datos son únicamente aquellos que están autenticados y además tienen que ser dueños del contenido (Firebase, 2020).

```
{
  "rules": {
    "some_path": {
      "$uid": {
        // Allow only authenticated content owners access to their data
        ".read": "request.auth != null && request.auth.uid == $uid"
        ".write": "request.auth != null && request.auth.uid == $uid"
      }
    }
  }
}
```

Ilustración 30 Dueños de contenido

Nota: Ilustración 30 Dueños de contenido, Reglas de Firebase, imagen obtenida de la documentación oficial de Firebase.

En la “Ilustración 31 Todos Pueden leer, solo dueños de contenido escriben” se indica que se puede variar las configuración de contenido. En este caso, la regla indica que en la base de datos todos pueden leer el contenido, pero únicamente los dueños del contenido pueden realizar las modificaciones (Firebase, 2020).

```
{
  // Allow anyone to read data, but only authenticated content owners can
  // make changes to their data

  "rules": {
    "some_path": {
      "$uid": {
        ".read": true,
        // or ".read": "auth.uid != null" for only authenticated users
        ".write": "auth.uid == $uid"
      }
    }
  }
}
```

Ilustración 31 Todos Pueden leer, solo dueños de contenido escriben.

Nota: Ilustración 31 Todos Pueden leer, solo dueños de contenido escriben, Reglas de Firebase, imagen obtenida de la documentación oficial de Firebase (Firebase, 2020).

6.1.2 Estructura de una base de datos de Firebase

La información de la base de datos de Firebase es almacenada como objeto JSON. A diferencia de las bases de datos relacionales que tienen como estructura básica a tablas y registros, las bases de datos de Firebase se conceptualizan como árboles JSON alojados en la nube. Al agregar datos a un árbol, este se convierte en un nodo existente con una clave asociada. Se puede agregar una clave propia como un ID o se la puede obtener de manera aleatoria (Firebase, 2020).

Las bases de datos de Firebase usan árboles estilo JSON, donde los datos almacenados pueden representarse como tipos de datos nativos que corresponden a JSON. De esta manera se puede escribir más fácilmente (Firebase, 2020).

```
{  
  "users": {  
    "alovelace": {  
      "name": "Ada Lovelace",  
      "contacts": { "ghopper": true },  
    },  
    "ghopper": { ... },  
    "eclarke": { ... }  
  }  
}
```

Ilustración 32 Estructura Base de Datos

Nota: Ilustración 32 Estructura Base de Datos, Estructura de la base de datos, imagen obtenida de la documentación oficial de Firebase (Firebase, 2020).

Anidación de datos

La base de datos en tiempo real de Firebase permite que los datos puedan ser anidados a un máximo de 32 niveles. Cuando se especifica la ubicación de un nodo en la base de datos, se puede recuperar todos los nodos secundarios que están dentro de el nodo superior. Cuando se otorga a un usuario el acceso de lectura o escritura en

un nodo de la base de datos, también se le está dando acceso a todos los datos que se encuentran en ese nodo (Firebase, 2020).

```
{  
  // This is a poorly nested data architecture, because iterating the children  
  // of the "chats" node to get a list of conversation titles requires  
  // potentially downloading hundreds of megabytes of messages  
  "chats": {  
    "one": {  
      "title": "Historical Tech Pioneers",  
      "messages": {  
        "m1": { "sender": "ghopper", "message": "Relay malfunction found. Cause: moth." },  
        "m2": { ... },  
        // a very long list of messages  
      }  
    },  
    "two": { ... }  
  }  
}
```

Ilustración 33 Anidación de datos

Nota: Ilustración 33 Anidación de datos: anidación de datos, imagen obtenida de la documentación oficial de Firebase (Firebase, 2020).

6.1.3 Diseño de la base de datos

La ‘Ilustración 34 estructura de la base de datos’ muestra la estructura que tiene la base de datos del proyecto. La base de datos va a almacenar toda la información bajo un único componente que será el “Root”. El componente principal va a desglosar dos componentes secundarios: El componente de usuarios y el componente de estacionamientos.

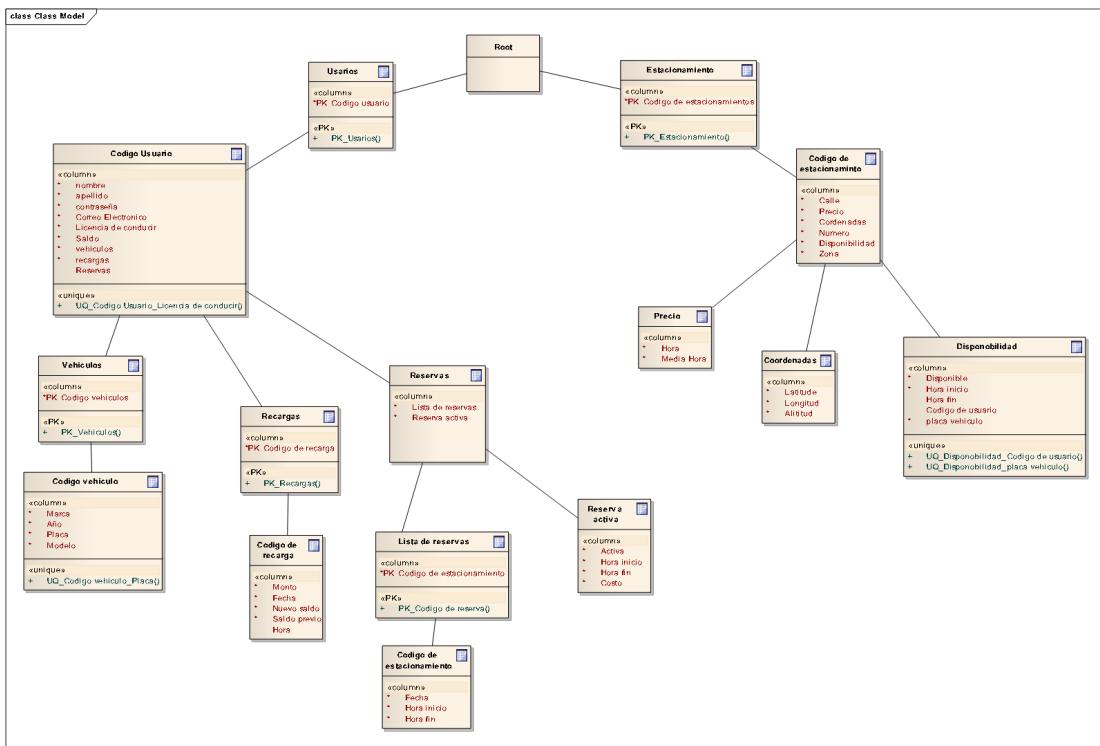


Ilustración 34 estructura de la base de datos.

Componente usuarios

Este componente tiene una lista de los componentes, llamados código de usuario. Estos van a tener como valor un código único asignado por “Firebase Authentication”, este código es el identificador único del usuario. Este código se genera automáticamente cuando el usuario se registra por primera vez en el sistema.

Componente código usuario

El “Componente código del usuario” almacena la información personal de los usuarios. es decir, nombre, apellido, licencia de conducir, correo electrónico, saldo, contraseña y los componentes de recargas, vehículos y reservas. Los datos personales son obtenidos al momento del registro del usuario.

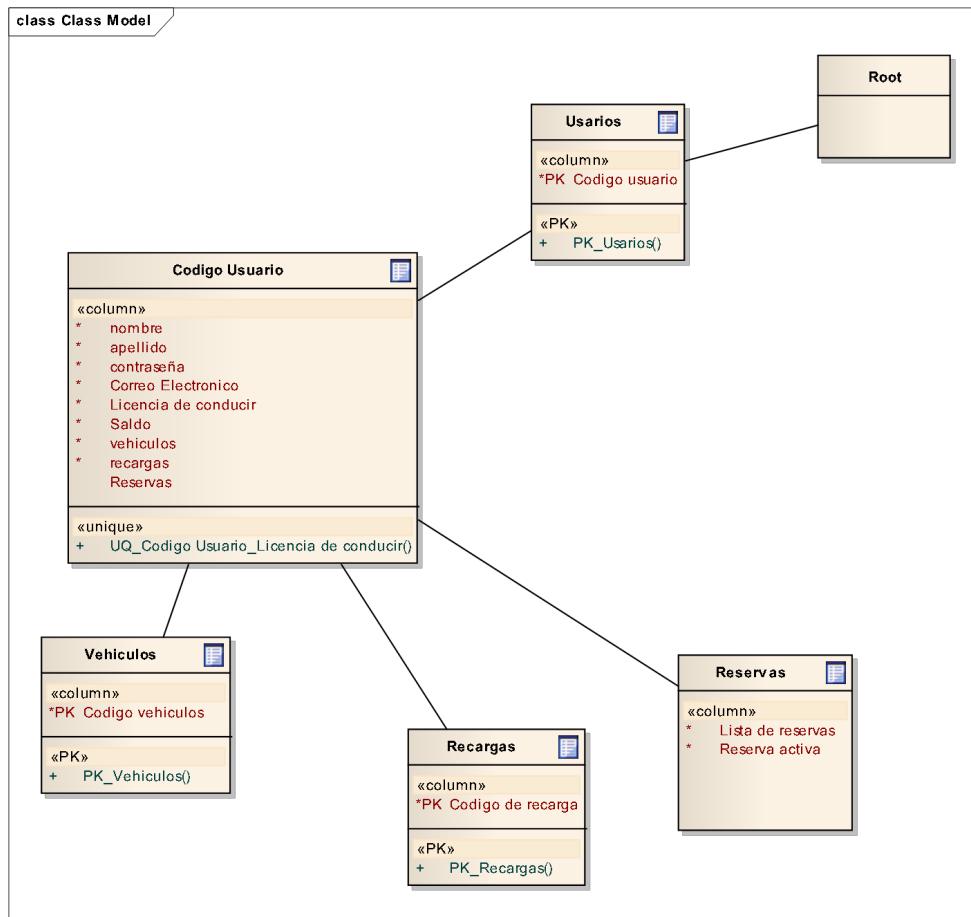


Ilustración 35 componentes usuarios

Componente vehículos

El “componente vehículos” almacena un listado de componentes, llamados código vehículo. Los componentes de esta lista tienen como valor un código único que es generado por Firebase. Este código es el identificador único de los vehículos en el sistema, este se genera cuando el usuario registra por primera vez su vehículo.

Componente código de vehículo

El “componente código de vehículo” registra la información proporcionada por los usuarios de sus vehículos. Los datos en este nodo son el numero de la placa, marca modelo y año de fabricación. El usuario únicamente va a poder realizar transacciones en este componente cuando este debidamente registrado en el sistema.

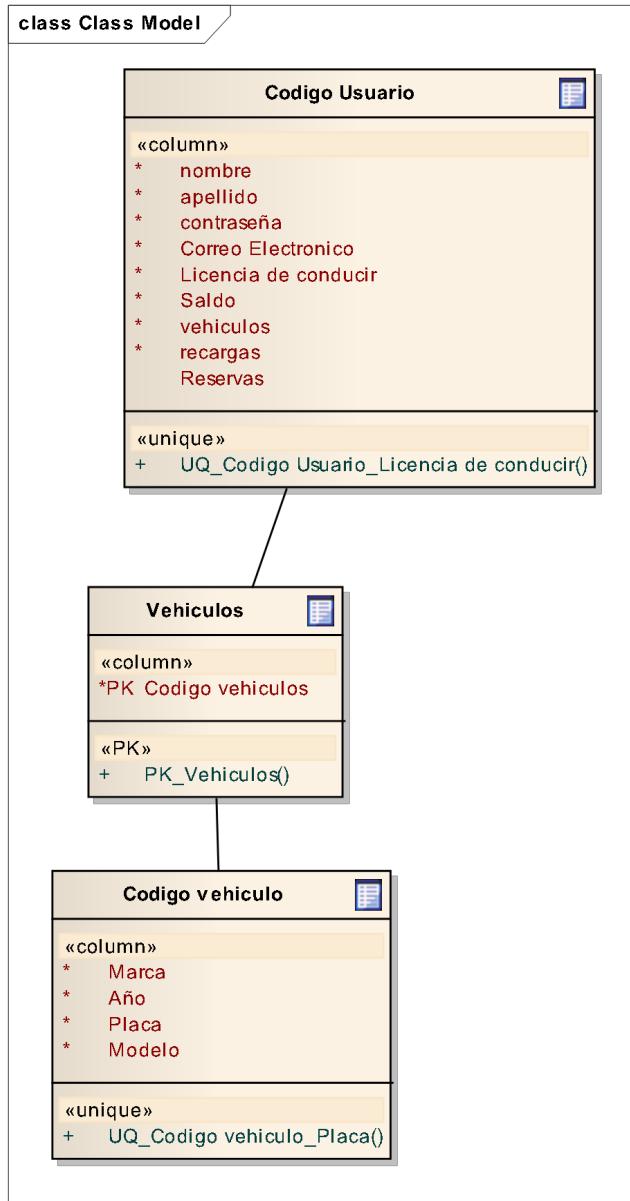


Ilustración 36 componentes vehículos

Componente recargas

El “componente recargas” tiene un listado de componentes, llamados código de recarga. Estos componentes tienen como valor un código único generado por Firebase, cuando el usuario realiza una recarga de efectivo con el sistema Khipu, Firebase genera este código.

Componente código de recarga.

El “componente código de recarga” almacena la información proporcionada por Khipu y por el sistema. Los datos almacenados son el monto de la recarga, la fecha, hora, el nuevo saldo y el saldo previo. Estos datos son importantes para mantener un registro claro y conciso de todas las transacciones realizadas por los usuarios.

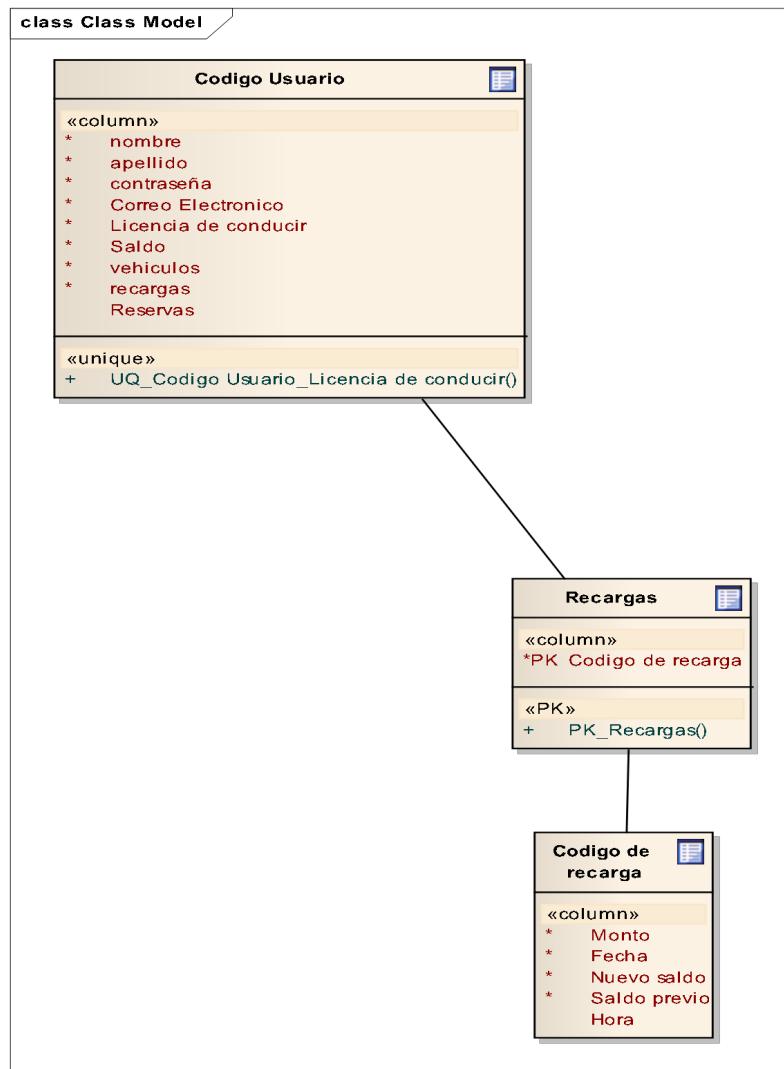


Ilustración 37 componentes recargas

Componente reservas.

El “componente reservas” esta compuesto por otros dos nodos: El “Componente listado de reservas” y el “Componente reserva activa”.

Componente listado de reservas

El componente “listado de reservas” contiene un listado de componentes, llamado código de reserva. Estos componentes tienen como valor un código único de las reservas realizados por el usuario, el código es generado por Firebase.

Componente código de reserva

El “Componente código de reserva” almacena la información sobre las reservas realizadas por el usuario. Hora de inicio, hora fin y código de estacionamiento son los datos que son almacenados al momento en el que un usuario realiza una reserva. Esto es con el fin de tener en registro todas las operaciones realizadas.

Componente reserva activa

El “Componente reserva activa” informa si el usuario esta haciendo uso en ese momento de un estacionamiento e informa si ingreso, la hora en la que finalizará el uso del estacionamiento y el código de estacionamiento que se esta usando.

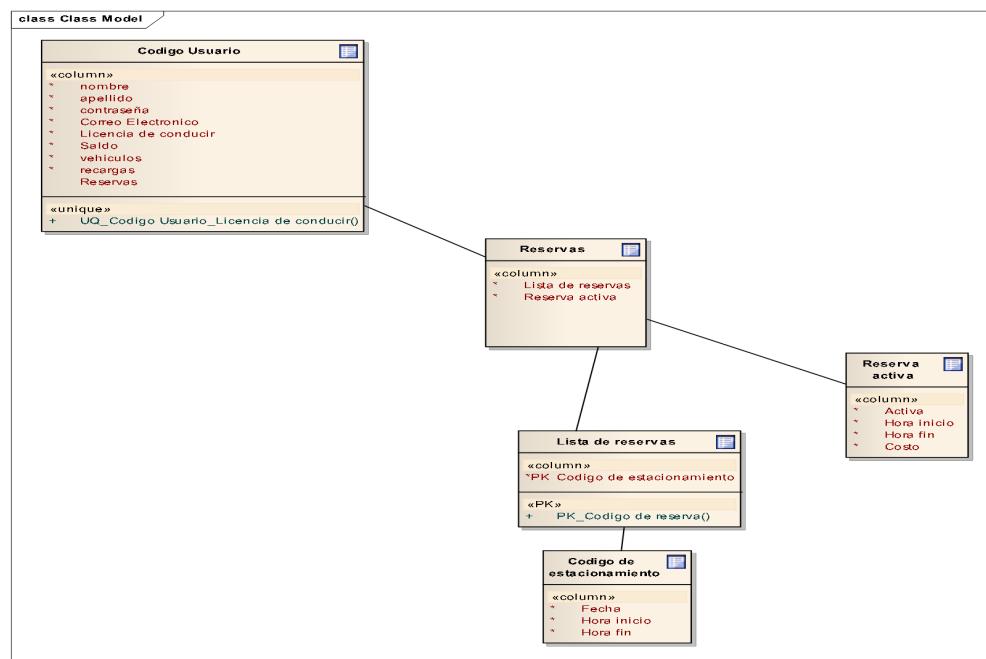


Ilustración 38 componentes reserva

Componente estacionamientos

El “Componente estacionamiento” contiene un listado de componentes, llamados código de estacionamiento. Estos nodos tienen como valor el código único de cada estacionamiento.

Componente código de estacionamiento

El “Componente código de estacionamiento” almacena los datos de los estacionamientos. El contenido de este componente es la zona del estacionamiento, la calle, el numero de estacionamiento, el nodo de las coordenadas, el nodo del precio y finalmente el nodo de la disponibilidad.

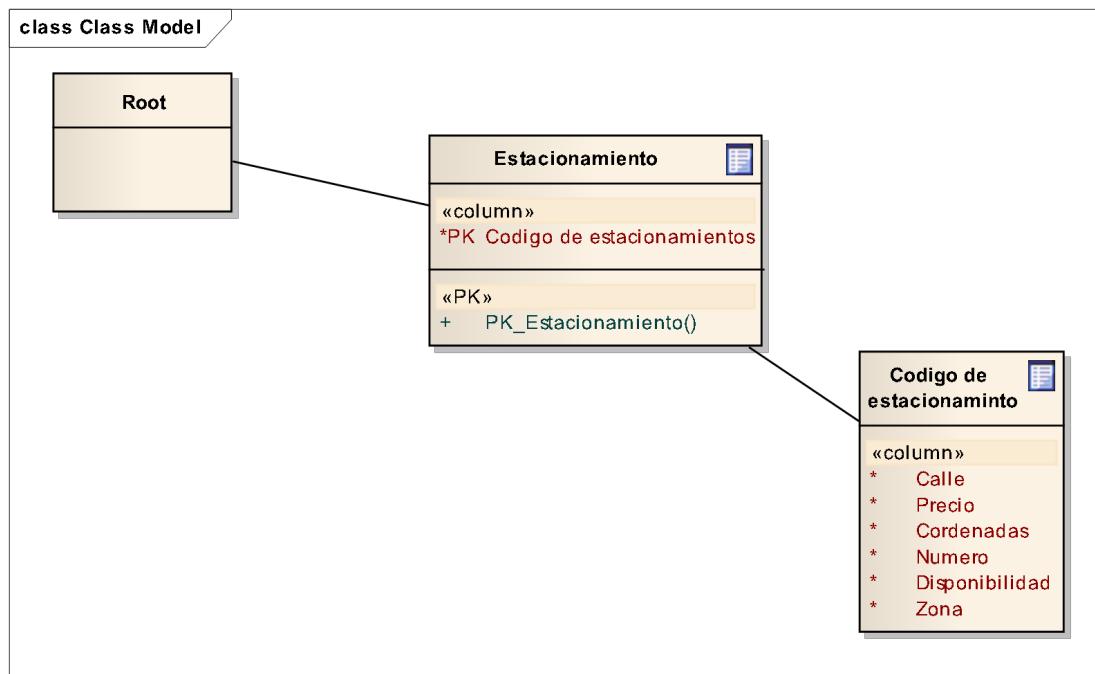


Ilustración 39 Componente Código estacionamiento

Componente coordenadas

En el “Componente coordenadas” se almacenan las coordenadas de altitud, longitud y latitud de los estacionamientos para que puedan ser desplegados en el sistema de Google Maps de manera fácil y eficiente.

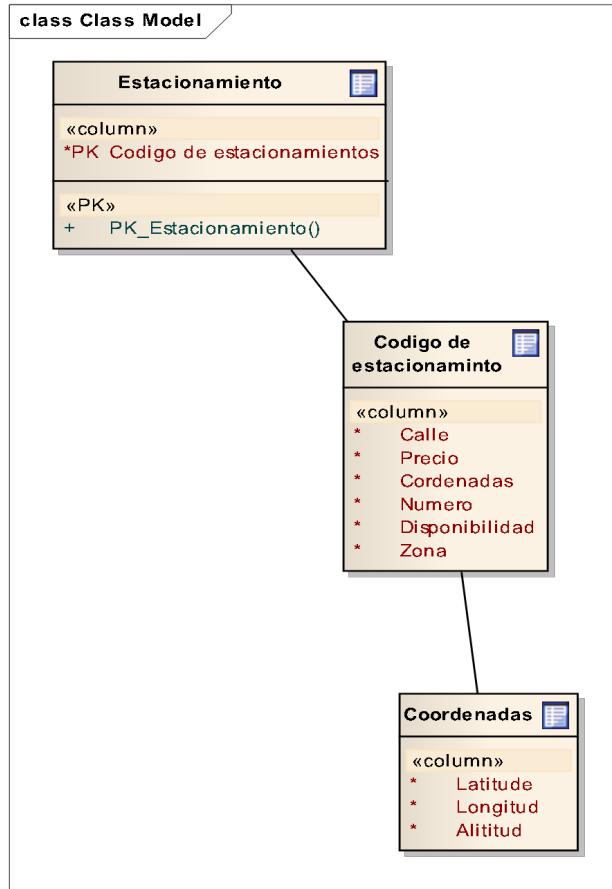


Ilustración 40 componente estacionamiento

Componente disponibilidad

El “Componente disponibilidad” almacena la información de sobre el uso del estacionamiento, la disponibilidad del estacionamiento requerido, hora de inicio, hora de finalización y código de vehículo.

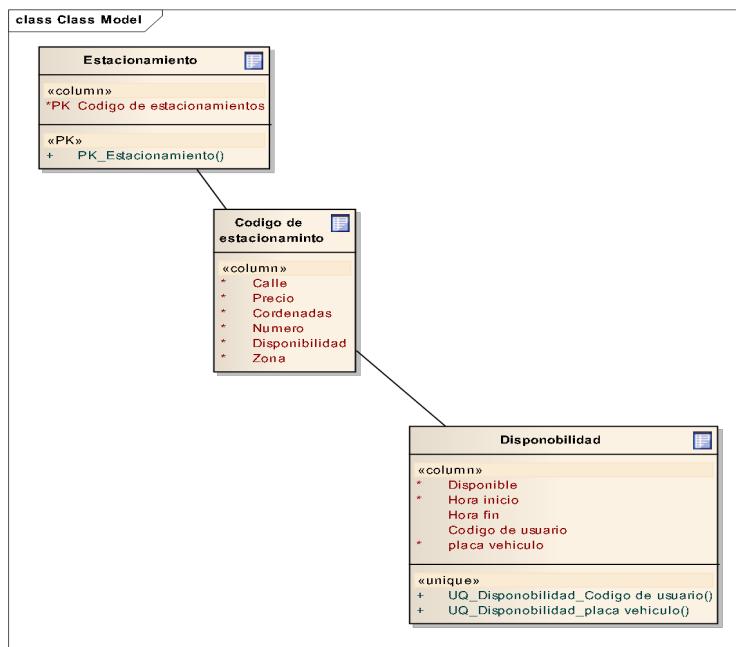


Ilustración 41 componente disponibilidad

Componente Precio

Este componente almacena el costo del estacionamientos por media hora y el costo del estacionamiento por una hora.

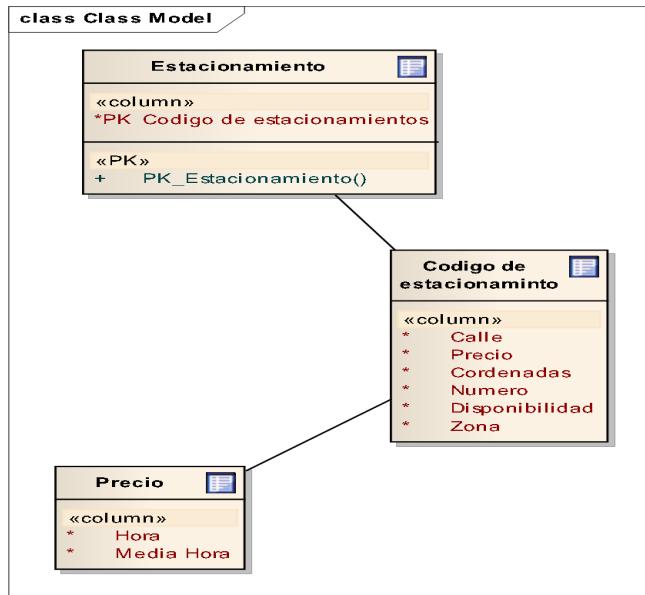


Ilustración 42 componente precio

6.1.4 Prueba de carga a la base de datos

Para probar la eficiencia y el rendimiento de la base de datos de Firebase, se realizo dos pruebas de carga a la base de datos. para realizar estas pruebas se utilizo la herramienta Jmeter.

Apache Jmeter es un software gratuito desarrollado en Java para realizar pruebas de carga y medir el rendimiento. Esta aplicación en sus inicios fue desarrollada para realizar pruebas a aplicaciones web, pero esta fue creciendo y realiza pruebas en una mayor cantidad de software.

Firebase en su versión gratuita establece que el sistema puede soportar una cantidad máxima de 100 usuarios por segundo. Para comprobar esto se realizaron dos pruebas de carga a la base de datos una de 100 usuarios por segundo y la segunda de 500 usuarios por segundo.

Carga de 100 usuarios

En la Ilustración 43 Prueba de 100 están los resultados de la primera prueba de carga realizada al sistema. Jmeter muestra los resultados en forma de tabla. En la columna de Status se ve una imagen de color verde, esto significa que el resultado de esa prueba fue positivo. Existen otras 9 columnas en la tabla que proporciona Jmeter. La primera columna indica el numero de prueba, la segunda la hora que se inicio cada prueba, la tercera el nombre de la prueba, la cuarta el tipo de prueba, la quinta el tiempo de ejecución, la sexta los bytes, la séptima los bytes enviados y las octava y novena son la latencia y el tiempo de conexión respectivamente.

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(…
1	11:30:15.199	Thread Group 1...	HTTP Request	2203	✓	77881	619	553	416
2	11:30:15.199	Thread Group 1...	HTTP Request	2405	✓	78177	619	561	423
3	11:30:15.280	Thread Group 1...	HTTP Request	2339	✓	78021	619	631	450
4	11:30:15.244	Thread Group 1...	HTTP Request	2448	✓	77909	619	541	401
5	11:30:15.255	Thread Group 1...	HTTP Request	2512	✓	77969	619	660	482
6	11:30:15.673	Thread Group 1...	HTTP Request	2204	✓	77961	619	576	413
7	11:30:15.851	Thread Group 1...	HTTP Request	2202	✓	78017	619	589	448
8	11:30:15.300	Thread Group 1...	HTTP Request	2930	✓	77989	619	613	437
9	11:30:15.233	Thread Group 1...	HTTP Request	3364	✓	77921	619	546	399
10	11:30:15.261	Thread Group 1...	HTTP Request	3575	✓	78057	619	652	471
11	11:30:15.629	Thread Group 1...	HTTP Request	3463	✓	77981	619	629	456
12	11:30:15.859	Thread Group 1...	HTTP Request	3453	✓	78101	619	596	453
13	11:30:15.212	Thread Group 1...	HTTP Request	4476	✓	77885	619	551	406
14	11:30:15.375	Thread Group 1...	HTTP Request	4360	✓	78165	619	565	421
15	11:30:16.033	Thread Group 1...	HTTP Request	3709	✓	78125	619	1938	1510
16	11:30:15.824	Thread Group 1...	HTTP Request	3928	✓	77777	619	2103	1173
17	11:30:15.354	Thread Group 1...	HTTP Request	4494	✓	77917	619	564	423
18	11:30:15.182	Thread Group 1...	HTTP Request	5745	✓	77877	619	580	436
19	11:30:15.225	Thread Group 1...	HTTP Request	5703	✓	77933	619	555	407
20	11:30:15.323	Thread Group 1...	HTTP Request	5727	✓	78129	619	591	414
21	11:30:15.362	Thread Group 1...	HTTP Request	5762	✓	77869	619	558	416
22	11:30:15.649	Thread Group 1...	HTTP Request	5484	✓	78033	619	609	436
23	11:30:15.873	Thread Group 1...	HTTP Request	5441	✓	79001	619	590	451
24	11:30:15.701	Thread Group 1...	HTTP Request	5692	✓	78029	619	613	468
25	11:30:16.050	Thread Group 1...	HTTP Request	5457	✓	78005	619	556	413
26	11:30:16.109	Thread Group 1...	HTTP Request	5544	✓	77957	619	3916	3739

Ilustración 43 Prueba de 100

Nota: La *Ilustración 43 Prueba de 100* es obtenida del software Jmeter tras desarrollar la prueba de carga.

Todas las pruebas que están en la tabla anterior están divididas en tres partes y todas tienen el nombre de “Http Request”, estos “Request” son enviados por el Jmeter, si el servidor puede con la cantidad de “Request” enviados, envía una respuesta que el Jmeter lee.



Ilustración 44 http request

Nota: La *Ilustración 44 http request* es obtenida de la aplicación Jmeter tras realizar la prueba de carga.

En la *Ilustración 45 resultados http 1* se tienen los resultados recibidos del servidor de Firebase, se puede apreciar datos como el nombre, la fecha en la que se realizó la prueba, tiempo de carga, tiempo de conexión, la latencia, los bytes enviados, el tamaño de los bytes, el tipo de datos, errores, entre otros datos importantes.

```
Thread Name:Thread Group 1-221
Sample Start:2021-04-05 11:31:18 BOT
Load time:114792
Connect Time:73117
Latency:73283
Size in bytes:77985
Sent bytes:619
Headers size in bytes:2247
Body size in bytes:75738
Sample Count:1
Error Count:0
Data type ("text"|"bin"|" "):text
Response code:200
Response message:OK
```

```
HTTPSSampleResult fields:
ContentType: text/html; charset=UTF-8
DataEncoding: UTF-8
```

Ilustración 45 resultados http 1

Nota: La Ilustración 45 resultados http 1 es obtenida de la aplicación Jmeter tras realizar la prueba de carga.

En la Ilustración 46 resultados http 2 de igual manera que en la anterior ilustración se tiene los mismos datos pero con diferentes valores.

```
Thread Name:Thread Group 1-221
Sample Start:2021-04-05 11:32:31 BOT
Load time:39263
Connect Time:38344
Latency:39263
Size in bytes:839
Sent bytes:166
Headers size in bytes:839
Body size in bytes:0
Sample Count:1
Error Count:0
Data type ("text"|"bin"|" "):text
Response code:302
Response message:Found
```

```
HTTPSSampleResult fields:
ContentType: application/binary
DataEncoding: null
```

Ilustración 46 resultados http 2

Nota: La *Ilustración 46 resultados http 2* es obtenida de la aplicación Jmeter tras realizar la prueba de carga

Prueba de carga de 1000

Después de realizar la prueba con 100 usuarios, se realiza una segunda prueba de carga pero con 1000 usuarios por segundo. En este caso, los primeras solicitudes resultaron positivas, pero desde la prueba numero 145 el sistema comenzó a mostrar resultados negativos y después de las pruebas 200 todos los resultados salieron negativos.

Como en la prueba de carga anterior Jmeter nos devuelve los datos en una tabla, sin embargo, en esta segunda prueba los resultados fueron en su mayoría negativos. La tabla que Jmeter proporciona, en la sexta columna se puede apreciar una figura de color rojo, esta figura indica que no existe respuesta del servidor. En la Ilustración 47 Prueba de 1000 se puede apreciar parte de la tabla obtenida.

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(…
337	11:31:18.231	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
338	11:31:18.230	Thread Group 1...	HTTP Request	81391	✗	2426	0	0	81391
339	11:31:18.236	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
340	11:31:18.236	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
341	11:31:18.238	Thread Group 1...	HTTP Request	81388	✗	2426	0	0	81388
342	11:31:18.237	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
343	11:31:18.242	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
344	11:31:18.243	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
345	11:31:18.246	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
346	11:31:18.246	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
347	11:31:18.246	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
348	11:31:18.248	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
349	11:31:18.248	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
350	11:31:18.249	Thread Group 1...	HTTP Request	81389	✗	2426	0	0	81389
351	11:31:18.248	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
352	11:31:18.248	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
353	11:31:18.249	Thread Group 1...	HTTP Request	81391	✗	2426	0	0	81391
354	11:31:18.250	Thread Group 1...	HTTP Request	81391	✗	2426	0	0	81391
355	11:31:18.252	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
356	11:31:18.250	Thread Group 1...	HTTP Request	81392	✗	2426	0	0	81392
357	11:31:18.259	Thread Group 1...	HTTP Request	81392	✗	2426	0	0	81392
358	11:31:18.259	Thread Group 1...	HTTP Request	81392	✗	2426	0	0	81392
359	11:31:18.259	Thread Group 1...	HTTP Request	81392	✗	2426	0	0	81392
360	11:31:18.259	Thread Group 1...	HTTP Request	81392	✗	2426	0	0	81392
361	11:31:18.261	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390
362	11:31:18.261	Thread Group 1...	HTTP Request	81390	✗	2426	0	0	81390

Ilustración 47 Prueba de 1000

Nota: La *Ilustración 47 Prueba de 1000* es obtenida de la aplicación Jmeter tras realizar la prueba de carga.

En la Ilustración 48 Resultados negativos están los resultados negativos de unas de las pruebas. Igual que en los anteriores resultados, se tienen valores como el nombre, la fecha en la que se realizo la prueba, tiempo de cargado, tiempo de conexión, la latencia, etc.. Sin embargo, en las ultimas líneas se aprecia el mensaje de error recibido.

```
Thread Name:Thread Group 1-968
Sample Start:2021-04-05 11:31:18 BOT
Load time:81397
Connect Time:81397
Latency:0
Size in bytes:2426
Sent bytes:0
Headers size in bytes:0
Body size in bytes:2426
Sample Count:1
Error Count:1
Data type ("text"|"bin "|" "):text
Response code:Non HTTP response code: java.net.NoRouteToHostException
Response message:Non HTTP response message: No route to host

HTTPSampleResult fields:
ContentType:
DataEncoding: null
```

Ilustración 48 Resultados negativos

Nota: La Ilustración 48 Resultados negativos es obtenida del software Jmeter después de realizar la prueba de carga

6.2 Arquitectura del prototipo

6.2.1 Modelo vista controlador

El sistema esta desarrollado con el Modelo Vista Controlador (MVC), MVC es un tipo de arquitectura para software que ayuda a los desarrolladores a separar la información del sistema, las vistas o interfaz de usuario y la lógica del sistema en tres partes. (Universidad Alicante, 2015)

- **Modelo** representa los datos que maneja el sistema, la lógica de negocio, y sus mecanismos de persistencia.
- **Vista**, o interfaz de usuario, conforma la información que se envía al cliente y los mecanismos interacción con éste. Es todo la parte visible por el usuario al momento del uso del sistema.

- **Controlador**, actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno (Universidad Alicante, 2015).

En la aplicación desarrollada, el modelo corresponde a las clases en Java donde se manejan la lógica, los cálculos de precios, la búsqueda de estacionamientos, etc. La vista esta en los archivos XML. El controlador son clases en Java donde se gestionan la conexión entre los archivos XML y la lógica de negocio.

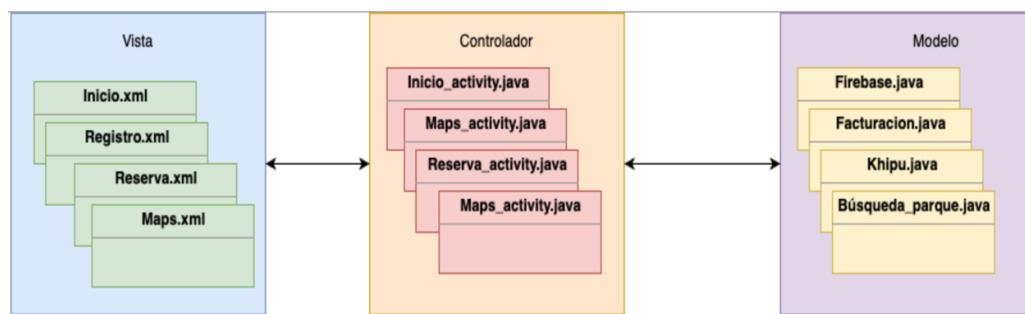


Ilustración 49 Modelo Vista Controlador

6.2.2 Diagrama de C4

En la “Ilustración 50 Diagrama c4 nivel 1” el sistema esta compuesto por 4 componentes principales. Los usuarios son los que van a utilizar el sistema. Estos pueden ser tanto los clientes, los estacionamientos de la alcaldía o los controladores contratados. El sistema de estacionamientos será usado tanto por los clientes como por los controladores. El sistema de Firebase permitirá al sistema estar conectado en la nube y con actualizaciones en tiempo real. Finalmente, la plataforma de pagos permite poder efectuar las transacciones económicas.

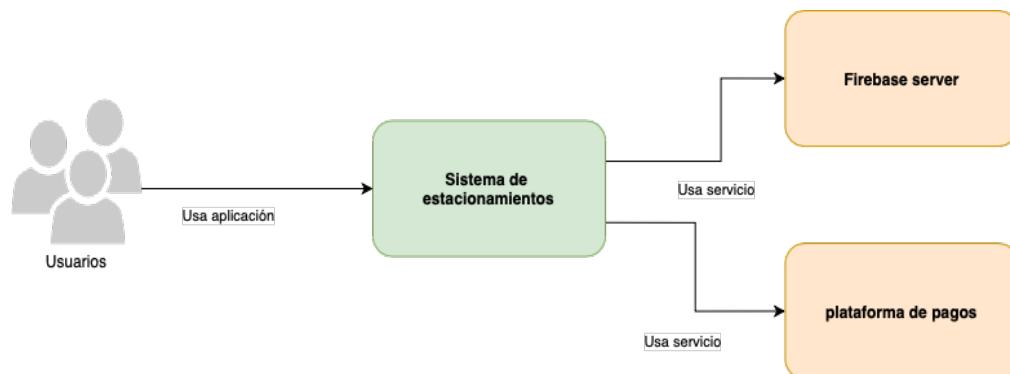


Ilustración 50 Diagrama c4 nivel 1

En la “Ilustración 51 Diagrama c4 nivel 2” se muestran los subcomponentes del sistema de estacionamientos. Este sistema esta integrado por cinco subcomponentes. La “aplicación para clientes” es usada para que se conecten a la plataforma de pagos y a Google Maps. La aplicación de los controladores es usada para que los funcionarios de la alcaldía realicen sus controles. El sistema de Google Maps permite a la aplicación de clientes mostrar los mapas de la ciudad. La Api de Firebase de Firebase se utiliza para que exista conexión entre las aplicaciones de cliente y de controlador. La Api de Khipu conecta en la aplicación de cliente con la plataforma de pago.

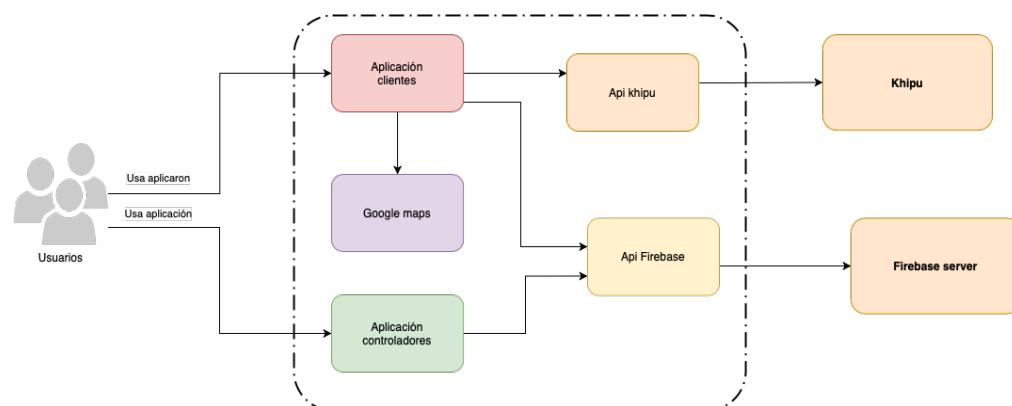


Ilustración 51 Diagrama c4 nivel 2

En la “Ilustración 52 Diagrama c4 nivel 3” se muestra los subcomponentes de la aplicación clientes y la aplicación de controladores. La Aplicación Clientes se descompone en los componentes de sistema de pagos que es la conexión entre la Api de Khipu, Registro de vehículos, registro de usuarios y el sistema de reserva para estacionamientos.

La aplicación de controladores se divide en 2 subcomponentes, el sistema de control y el sistema de reservas manuales.

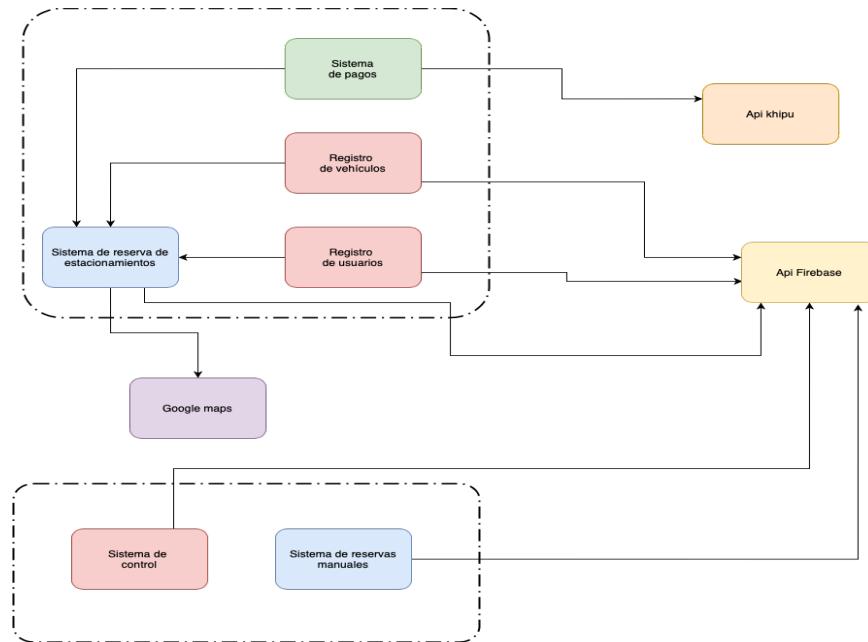


Ilustración 52 Diagrama c4 nivel 3

6.3 Estructura de la aplicación

Todos los proyectos desarrollados en Android estudio tienen una estructura definida que se crea automáticamente al crear el proyecto. Las aplicaciones se dividen en 4 partes:

- El código en los archivos XML.
- Archivos en Java.
- El archivo manifiesto.
- Los archivos Gradle Scripts.

6.3.1 Diseño de interfaces graficas

Las interfaces graficas en los dispositivos Android están formadas por un conjunto de pantallas, llamadas actividades, que interactúan entre ellas y son objetos que se extienden a la clase “Activity”. Las actividades son autónomas, cada uno de estos se encargan de sus componentes y pueden llamar a ciertas actividades de otras aplicaciones (Institut Puig Castellar, 2020).

Los componentes de visuales, como “Button”, “EditText”, “ImageView”, en las actividades, son extensiones de la clase “View”, y los componentes que permiten agrupar otros componentes, como “LinearLayout”, “GridLayout”, “ConstraintLayout” se extienden de la clase “ViewGroup” (Institut Puig Castellar, 2020).

Las actividades en Android se desarrollan en ficheros XML. Estos ficheros son totalmente independientes del código. Gracias a esto se gana flexibilidad y claridad. Generalmente es muy complicado y largo escribir código para construir la interfaz gráfica creando instancias de objetos para agruparlos (Institut Puig Castellar, 2020) al utilizar los ficheros XML para definir la interfaz gráfica. Simplemente se describe como los elementos se agrupan en las actividades y se utilizan etiquetas similares a las del desarrollo web. Las herramientas de desarrollo interpretan este fichero y generan el código necesario para la creación de la interfaz gráfica (Institut Puig Castellar, 2020).

Las ventajas de utilizar los ficheros XML son:

- Simple y claro en la descripción de las interfaces gráficas.
- Fácil y rápido modificar la ubicación de los objetos en la interfaz gráfica.
- El código y los ficheros están separados.

View

Visualización de Vistas

TextView

La función principal del “TextView” es mostrar el texto en la pantalla. Esta clase está compuesta por una lógica compleja que ayuda a tener una visualización del texto subrayado, hyperlinks, números, correos electrónicos, contraseñas entre otras funcionalidades. (Trebilcox-Ruiz, 2016)

ImageView

El “ImageView” tiene la función específica para mostrar imágenes en la pantalla, esto puede ser para visualizar imágenes desde la aplicación tanto como imágenes descargadas del internet (Trebilcox-Ruiz, 2016) .

Entrada y controles

Los objetos “View” no solo son utilizados para mostrar contenido estático a los usuarios, también son utilizados para recibir algún tipo de contenido para poder realizar el control de la aplicación (Trebilcox-Ruiz, 2016).

Button

La clase “Button” es probablemente el control mas básico de las aplicaciones, recibe los click de los usuarios para poder ejecutar un método en el código (Trebilcox-Ruiz, 2016).

Switch y CheckBox

Estas clases tienen dos estados activos e inactivo estos dos estados pueden alternarse. Esto es de gran ayuda para que el usuario pueda realizar ajustes en la aplicación (Trebilcox-Ruiz, 2016).

EditText

Los “EditText” son una extensión del la clase “TextView”, permite a los usuarios actualizar el texto que figura en la aplicación por una entrada de teclado (Trebilcox-Ruiz, 2016).

Adapter Based Views

Todos los objetos vistos anteriormente son elementos individuales, pero en ocasiones se tienen que utilizar colecciones de elementos. Estas clases de “View” generalmente utilizan un adaptador para manejar la manera en la que se muestran en la pantalla (Trebilcox-Ruiz, 2016).

ListView

La clase “ListView” es usada para ejecutar una colección de elementos en una vista lineal, de una sola columna y desplegable. Los elementos pueden ser seleccionados para realizar una acción (Trebilcox-Ruiz, 2016).

GridView

La clase “GridView” emplea un adaptador y ejecuta los elementos en múltiples columnas en la pantalla (Trebilcox-Ruiz, 2016).

Spinner

El “Spinner” es el ultimo objeto de la sección de “View”. Estos aceptan adaptadores que les permiten mostrar elementos en un menú desplegable (Trebilcox-Ruiz, 2016).

ViewGroups

Los “ViewGroup” son objetos que no son visibles, se los usa para poder contener a los objetos “View” y también a otros “ViewGroup” con el fin de organizar el contenido de la aplicación (Trebilcox-Ruiz, 2016).

LinearLayout

Este objeto permite mostrar los elementos en un orden. Tanto horizontal como vertical. También son capaces de asignar un peso específico a los objetos que están dentro del mismo para que puedan estar alineados de forma pareja (Trebilcox-Ruiz, 2016).

RelativeLayout

Permite relacionar a los objetos que están adentro de este layout, dándoles mas flexibilidad y libertad para poder ubicarlos (Trebilcox-Ruiz, 2016).

FrameLayout

Utilizado para mostrar solo una vista hijo a la vez, este layout muestra los elementos en una pila y provee la manera más fácil de mostrar un elemento en diferentes tipos de pantalla (Trebilcox-Ruiz, 2016).

ScrollView

Estas es una extensión del “FrameLayout”, esta clase maneja el despliegue de los objetos dentro de del mismo (Trebilcox-Ruiz, 2016).

ViewPager

El “ViewPager” es utilizado para administrar varias vistas mientras que solo esta visualizando una a la vez, esta clase permite usar una adaptador y también permite a los usuarios deslizarse de derecha a izquierda y viceversa (Trebilcox-Ruiz, 2016).

RecyclerView

Esta clase esta relacionada a las clases de “ListView” y de “GridView” (Trebilcox-Ruiz, 2016).

6.3.1.1 Formularios

Los formularios de la aplicación están desarrollados con un conjunto de la clase “EditText” de Android. Tienen un diseño simple y amigable, no tienen colores muy fuertes, son fáciles de entender y usar. En la Ilustración 53 Formularios se puede ver un ejemplo de los formularios de la aplicación. Los “EditText” de los formularios son blancos con bordes azules y tienen forma ovalada. Las letras están posicionadas al centro del “EditText” y son de color azul. Y finalmente tienen un icono animado al lado izquierdo.



Ilustración 53 Formularios

Nota: La Ilustración 53 Formularios es obtenida del prototipo desarrollado.

6.3.1.2 Botones

Los botones de la aplicación están hechos con la clase “Button” de Android. Estos son de color azul y de forma ovalada. Su texto esta centrado y es de color blanco, tienen un icono animado a lado derecho de cada botón que hace referencia a la acción que

cada botón ejecuta. En la Ilustración 54 Botones se puede ver con claridad como los botones están diseñados.



Ilustración 54 Botones

Nota: La Ilustración 54 Botones es obtenida del prototipo desarrollado.

6.3.1.3 Menú principal

El menú de la aplicación utiliza la clase de “GridLayout” que esta dividido en dos columnas y tres filas. En cada posición esta compuesta por un “CardView” un “Button” y por un “TextView”. El botón tiene como fondo un ícono grande que hace referencia a la acción que realiza y debajo esta un “TextView” que indica la función que el botón va a realizar.

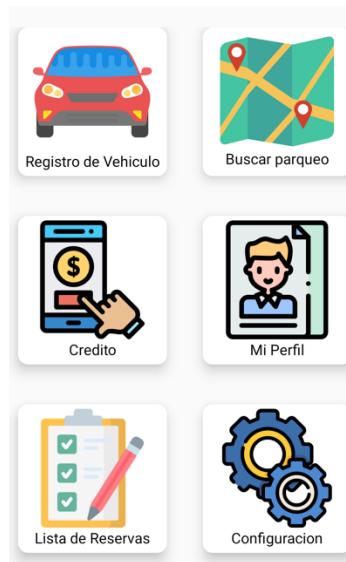


Ilustración 55 CardView

Nota: La Ilustración 55 CardView es obtenida del prototipo desarrollado.

6.3.1.4 Mapa de Google

El mapa a usar en la aplicación es el mapa que viene por defecto en Google Maps en la Ilustración 56 Mapa de Google muestra un ejemplo de una sección de como el mapa esta conformado. Google Maps ya te proporciona pines preestablecidos como se

puede apreciar. Estos identifican negocios, bancos, restaurantes de la zona, también nos muestra el nombre de las zonas, calles o avenidas según el usuario haga "zoom" en el mapa.

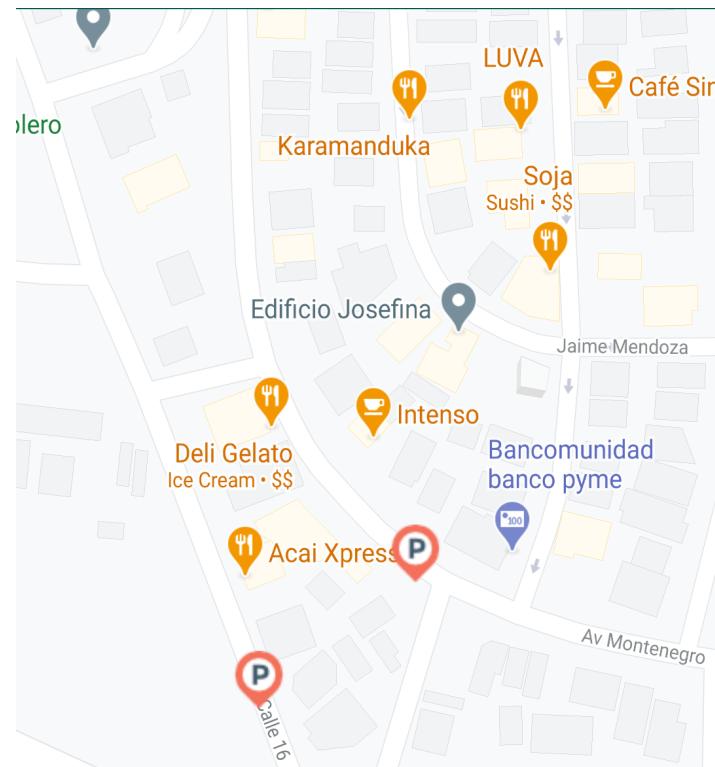


Ilustración 56 Mapa de Google

Nota: La Ilustración 56 Mapa de Google es obtenida del prototipo desarrollado.

A todo este contenido proporcionado por Google se le agrego el pin ilustrado en la Ilustración 57 Pin. Este pin mostrará la ubicación exacta de los estacionamientos disponibles. El pin tiene un borde externo rojo con un interior blanco y la letra P en color negro que hace referencia a parqueo.



Ilustración 57 Pin

Nota: La Ilustración 57 Pin es obtenida del prototipo desarrollado.

6.3.2 Funciones aplicación

6.3.2.1 Diagrama de clases

En la “Ilustración 58 Diagrama de clases” se observan las clases que están siendo utilizadas en la aplicación y las relaciones entre ellas. El sistema esta compuesto por 11 clases en la aplicación principal.

Clase Reserva: Administra toda la lógica sobre las reservas de los estacionamientos. Esta clase genera las reservas de los estacionamientos, realiza cambios en las reservas si fuera necesario y genera los códigos QR para iniciar o finalizar las reservas, entre otras funcionalidades.

Esta clase se relaciona con las clases Firebase, usuario y Google Maps. Firebase es utilizado para realizar las consultas a la base de datos. La clase usuario permite a la aplicación obtener los datos de los usuarios y Google Maps permite obtener las coordenadas y la información de los estacionamientos.

Clase Firebase: Maneja todas las conexiones con el servidor. Con esta clase se puede subir y/o crear nuevos nodos con información en la base de datos. Realiza la lectura de los nodos necesarios según sean especificados. También se pueden realizar modificaciones. Maneja la gestión de usuarios, como la creación de uno nuevo y a verificar si el usuario ya tiene la sesión iniciada.

Clase Factura: Aquí se genera la factura en PDF requerida por el usuario. Las facturas son descargadas y almacenadas en la memoria interna del dispositivo en el dispositivo.

Esta clase tiene relación con la clase usuario. Esta clase permite obtener la información del usuario para la generación de la factura.

Clase Google Maps: Realiza todas las gestiones con el mapa del sistema. En esta clase se obtiene la ubicación en tiempo real del dispositivo, se generan los pines del dispositivo y se los ubica según las coordenadas obtenidas.

Esta clase tiene relación con las clases Firebase y estacionamientos. La clase Firebase es utilizada para realizar las consultas a la base de datos. la clase estacionamientos permite obtener la información de los parqueos todos los estacionamientos de la alcaldía.

Clase Estacionamientos: Esta clase maneja y gestiona todos los atributos de los estacionamientos en el sistema.

La clase estacionamiento se relaciona con Firebase para poder realizar las consultas a la base de datos.

Clase Usuario: Esta clase maneja y gestiona todos los atributos de los usuarios en la aplicación.

La clase usuario se relaciona con la clase Firebase para poder realizar las consultas a la base de datos. también se relaciona con la clase automóvil para poder obtener la información sobre los vehículos utilizados.

Clase Automóvil: Esta clase maneja y gestiona todos los atributos de los automóviles de los usuarios.

Clase Carga_Credito: Esta clase administra todos los movimientos que se gestionan con la plataforma de pago; obtiene los montos que el usuario cargo y genera la listas de cargas entre otras operaciones.

Esta clase se relaciona con la clase Firebase para poder realizar las consultas a la base de datos.

Clase IniciarSesion: Esta clase obtiene los datos ingresados por el usuario para el inicio de sesión para gestionarlos con la clase de Firebase.

Esta clase se relaciona con la clase Firebase para poder realizar las consultas a la base de datos. Además, se necesita relacionar con la clase usuario para poder validar y verificar los datos del usuario.

Clase Registrar_usuario: Esta clase gestiona a los nuevos usuarios con la clase Firebase y la clase usuario.

Esta clase se relaciona con la clase Firebase para poder realizar las consultas a la base de datos. Además, se necesita relacionar con la clase usuario para poder validar y verificar los datos del usuario.

Clase Notificaciones: Gestiona todas las notificaciones generadas por el sistema.

La clase notificaciones se relaciona con la clase Firebase para poder realizar las consultas a la base de datos.

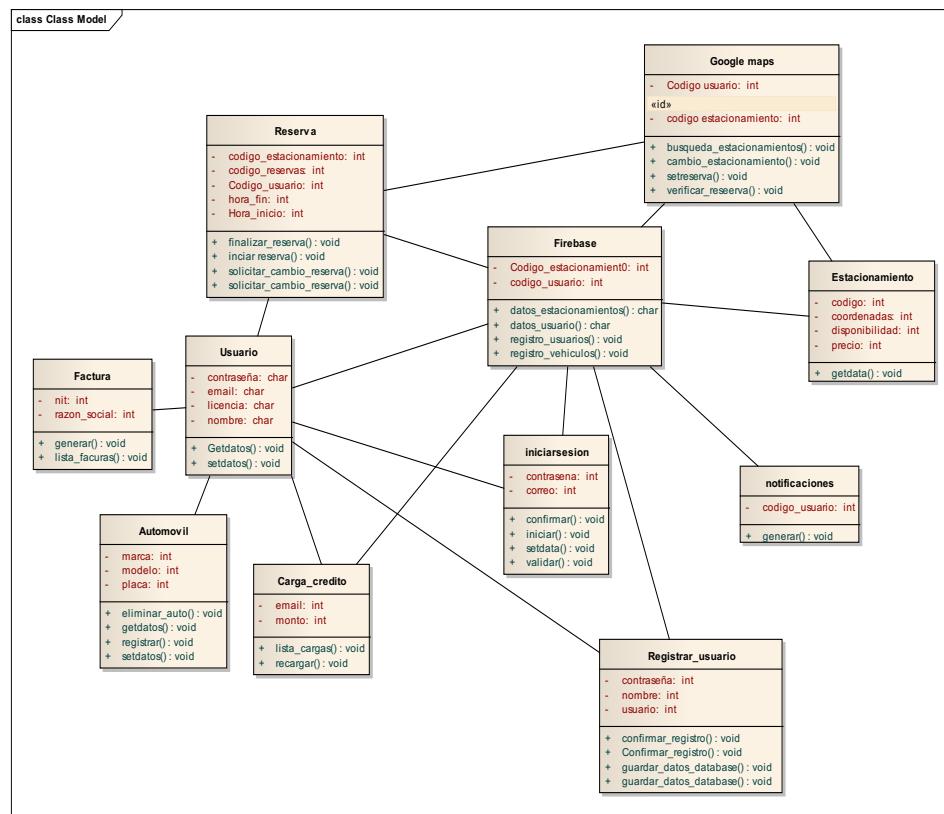


Ilustración 58 Diagrama de clases

6.3.2.2 Autenticación de usuarios

Una vez se importadas las librerías de “FirebaseAuth”, se declara la instancia de Firebase como se ilustra en la Ilustración 59 Declarar Firebase.

```
 FirebaseAuth auth;
```

Ilustración 59 Declarar Firebase

Nota: La Ilustración 59 Declarar Firebase es obtenida del código del proyecto.

Después de declarar se tiene que inicializar la instancia.

```
auth=FirebaseAuth.getInstance();
```

Ilustración 60 FirebaseAuth instance

Nota: La Ilustración 60 FirebaseAuth instance es obtenida de el código del proyecto.

Creación de usuarios

Para registrar a nuevos usuarios en el sistema de autenticación Firebase. Se tiene que utilizar el método “CreateUserWithEmailAndPassword”. En la Ilustración 61 Crear usuario, se muestra un ejemplo de como se utiliza este método. Este es la forma mas simple para la creación de usuarios nuevos que Firebase ha establecido. Para que los usuarios sean registrados se obtiene el correo electrónico y la contraseña mediante los formularios de registro. Después si se desea se puede utilizar el método “addOnCompleteListener”. Este método permite a la aplicación en Android recibir una alerta de Firebase que alerta si las operaciones realizadas fueron exitosas o no. Firebase devuelve un objeto de tipo TASK.

```

auth.createUserWithEmailAndPassword(correo1,contra1).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if(task.isSuccessful()){
            FirebaseUser user = auth.getCurrentUser();
            String UserId = user.getUid();

        }else {

        }

    }
});
```

Ilustración 61 Crear usuario

Nota: La Ilustración 61 Crear usuario es obtenida del código del proyecto.

Iniciar sesión

Debido a que en el registro de los usuarios se esta utilizando el método de correo y contraseña, para iniciar sesión se necesita utilizar un método parecido. En este caso se esta utilizando el método “signInWithEmailAndPassword”, que se utiliza por defecto. En este método, el usuario solo necesita ingresar los datos de correo electrónico y su contraseña.

También se utiliza el método “addOnCompleteListener”. Este método permite a la aplicación en Android recibir una alerta de Firebase si se pudo Iniciar sesión, el método te devuelve una objeto de tipo TASK. Este objeto contiene los Logs del registro del usuario.

```

auth.signInWithEmailAndPassword(correo,contra).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {
        if (task.isSuccessful()){

        }else {

        }

    }
});
```

Ilustración 62 Iniciar Sesión Firebase

Nota: La Ilustración 62 Iniciar Sesión Firebase es obtenida del código del proyecto.

Verificar sesión

Para verificar si el usuario inicio sesión en el dispositivo se tiene que usar el código que se puede ver en la Ilustración 63 Verificar . Primero se tiene que crear el objeto “usuario” y se le asigna un valor de FirebaseAuth.getCurrentUser. Si el valor es nulo no existe ningún usuario registrado, caso contrario el objeto FirebaseAuth tiene los valores de FirebaseAuthUser.

```
 FirebaseAuthUser firebaseUser;
firebaseUser=firebaseAuth.getCurrentUser();
if (firebaseUser!=null){  
  
}else {  
  
}
```

Ilustración 63 Verificar sesión

Nota: La Ilustración 63 Verificar sesión es obtenida del código del proyecto.

6.3.2.3 Base de datos.

Una vez importadas las librerías de “FirebaseDatabase”, se declara un objeto tipo “DatabaseReference”.

```
 DatabaseReference reference;
```

Ilustración 64 Reference

Nota: La Ilustración 64 Reference es obtenida del código del proyecto.

A esta instancia se le asigna una ruta de referencia donde esta ubicada la base de datos. Esta se la obtiene de la clase “FirebaseDatabase”.

```
reference = FirebaseDatabase.getInstance().getReference();
```

Ilustración 65 FirebaseDatabase

Nota: La Ilustración 65 FirebaseDatabase es obtenida del código del proyecto.

Lectura de datos

Para leer un nodo en la base de datos se especifica la dirección del nodo a leer indicando cuales son los nodos padres. En la Ilustración 66 Lectura de datos Firebase se ve un ejemplo donde se indica que la ruta objetivo es ir al registro de un usuario. Los nodos padres son “app”, “usuarios”, “uid” (el código único de usuario). Para la recuperación de los datos se utiliza el método “addValueEventListener”. Este método es el que permite la utilización de la base de datos en tiempo real. Cuando en la dirección del nodo se realiza un cambio se manda una señal al dispositivo para que realice la actualización del dispositivo sin tener que actualizar la app.

“addValueEventListener” no es el único método que permite realizar la lectura de la base de datos. Si en el sistema es requerida una única lectura de la base de datos, se puede utilizar el método “addListenerForSingleValueEvent”.

```
reference.child("app").child("usuarios").child(uid).addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        if(dataSnapshot.exists()) {

        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
    }
});
```

Ilustración 66 Lectura de datos Firebase

Nota: La Ilustración 66 Lectura de datos Firebase es obtenida del código del proyecto.

Escritura de datos

Para escribir en un conjunto de datos en la base de datos. se crea un objeto de tipo “Map”. Este maneja una arquitectura similar a la arquitectura de la base de datos, a este objeto “Map” se le introduce la información que se desea introducir a la base de datos. Para agregar un dato se ingresa una llave de tipo “String” y después se le da el objeto en tipo “Object”.

```
Map<String, Object> map=new HashMap<>();
map.put( k: "nombre", nombre1);
map.put( k: "apellido", apellido1);
map.put( k: "carnet", carnet1);
map.put( k: "correo", correo1);
map.put( k: "contra", contra1);
```

Ilustración 67 Map Firebase

Nota: La Ilustración 67 Map Firebase es obtenida del código del proyecto.

Para escribir el “Map” dentro de un nodo específico se tiene que establecer la ruta de escritura. Igual de cuando se lee información, y se utiliza el método “setvalue” para que se escriban los datos. En la Ilustración 68 Escribir datos Firebase se da un ejemplo de cómo se realiza la lectura de los datos.

```
databaseReference.child("app").child("usuarios").child(UserId).setValue(map).addOnCompleteListener(new OnCompleteListener<Void>() {
    @Override
    public void onComplete(@NonNull Task<Void> task) {
        if(task.isSuccessful()){
            ...
        }else {
            ...
        }
    }
});
```

Ilustración 68 Escribir datos Firebase

Nota: La Ilustración 68 Escribir datos Firebase es obtenida del código del proyecto.

6.3.2.4 Código del sistema

Ubicación sistema

Para obtener la ubicación del dispositivo se utilizan las clases “LocationManager” y “Location”. Estas dos clases ayudan a manejar el GPS del dispositivo para poder obtener las coordenadas. En la Ilustración 69 Obtener ubicación se muestra un ejemplo: Primero se verifica si la aplicación tiene los permisos necesarios para acceder al GPS. Despues se obtienen los servicios del sistema con la clase “LocationManager” y en la Clase Location se da el valor de la ubicación.

```
private void setlocationManager(){  
  
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&  
        ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED){  
        return;  
    }  
    LocationManager manager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);  
    Location location = manager.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);  
    setLocation(location);  
    // Toast.makeText(this, String.valueOf(location.getLatitude()), Toast.LENGTH_SHORT).show();  
    manager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 15000, 0, locationListener);  
}
```

Ilustración 69 Obtener ubicación

Nota: La Ilustración 69 Obtener ubicación es obtenida del código del proyecto.

Una vez obtenidas las coordenadas del dispositivo, se muestra la ubicación obtenida con un Pin en le mapa. En la Ilustración 70 setmarker muestra la función necesaria para poder agregar el Pin con las coordenadas obtenidas. La función recibe los valores de latitud y longitud, con estas dos variables se crea la variable “LatLng”. Para actualizar la posición de la cámara en el mapa se tiene que utilizar la clase “CamaraUpdate” y se manda las variable “LatLng” y el zoom que se deseé. Para actualizar la posición de un Pin es necesario retirar el anterior que esta en las coordenadas anteriores, para esto el “Marker” se tiene que ejecutar la función remove y finalmente a este “Marker” se le pasa las nuevas coordenadas con el titulo deseado para ubicarlo en una nueva posición.

```

private void Setmarker (double lat, double lon){

    LatLng coordenadas = new LatLng(lat,lon);
    CameraUpdate miUbicacion = CameraUpdateFactory.newLatLngZoom(coordenadas, v: 16);
    if (marker!= null){
        marker.remove();
    }

    marker=mMap.addMarker(new MarkerOptions().position(coordenadas).title("mi ubicacion").icon(BitmapDescriptorFactory.defaultMarker()));
    mMap.animateCamera(miUbicacion);

}

```

Ilustración 70 setmarker

Nota: La Ilustración 70 setmarker es obtenida del código del proyecto.

Crear código QR

En la Ilustración 71 crear QR se puede apreciar la función necesaria para generar los códigos QR necesarios para poder utilizar la aplicación. Se obtiene el UID del usuario y se obtiene las dimensiones de la pantalla, después con la clase “QRencoder” se le pasa el UID, las dimensiones de la pantalla y el tipo archivo que se tiene que codificar. El resultado se lo guarda en un “Bitmap”. Al contenedor de la imagen de lo pone el la imagen en “Bitmap”.

Primero se tiene que obtener el UID del usuario, después se tienen que verificar y obtener las dimensiones de la pantalla del dispositivo, cuando ya se tengan

Para generar los códigos QR utilizados para el inicio y el cierre de hora se tiene que utilizar el código del usuario. Este código, mediante la librería “QREncoder”, codifica el código del usuario. Después lo transforma en un objeto “Bitmap”.

```

public void crearqr(){
    if (uid.length() > 0) {
        WindowManager manager = (WindowManager) getSystemService(WINDOW_SERVICE);
        Display display = manager.getDefaultDisplay();
        Point point = new Point();
        display.getSize(point);
        int width = point.x;
        int height = point.y;
        int smallerDimension = width < height ? width : height;
        smallerDimension = smallerDimension * 3 / 4;

        qrgEncoder = new QRGEcoder(
            uid, bundle: null,
            QRGContents.Type.TEXT,
            smallerDimension);

        try {

            bitmap = qrgEncoder.encodeAsBitmap();
            imageView.setImageBitmap(bitmap);

        } catch (WriterException e) {

        }
    } else {
    }
}

```

Ilustración 71 crear QR

Nota: La Ilustración 71 crear QR es obtenida del código del proyecto.

Lectura de códigos QR

Para la lectura de los códigos QR se importa la librería “ZXing”, para importarla es necesario utilizar “implements” en la clase como se puede ver en la Ilustración 72 ZXing.

implements ZXingScannerView.ResultHandler :

Ilustración 72 ZXing

Nota: La Ilustración 72 ZXing es obtenida del código del proyecto.

En la Ilustración 73 variable myscan se crea una variable privada tipo “ZXingScanenerView” con el nombre myscan.

```
private ZXingScannerView myscan;
```

Ilustración 73 variable myscan

Nota: La Ilustración 73 variable myscan es obtenida del código del proyecto

En la Ilustración 74 código inicial se crea un método llamado scan que es llamado cuando el usuario presione el botón de escanear QR. Este método crea el objeto “ZXingScannerView”, después se llama al método “setContentView” que recibe el valor de “myscan”, se llama el “setResultHandler” y finalmente se inicia la cámara.

```
public void scan(View view){  
    myscan=new ZXingScannerView(context: this);  
    setContentView(myscan);  
    myscan.setResultHandler(this);  
    myscan.startCamera();  
}
```

Ilustración 74 código inicial

Nota: La Ilustración 74 código inicial es obtenida del código del proyecto.

Cuando el dispositivo detecta el código QR, lo descifra y ejecuta la función “handleResult”. Esta función recibe un objeto de tipo Result de este objeto se obtiene el código del usuario. El código es almacenado en una variable tipo String. Después se ejecuta la función “resumeCamaaraPreview”. y finalmente se llama la función validar. Esta función ayuda a determinar si el texto codificado en el QR es valido.

```

@Override
public void handleResult(com.google.zxing.Result result) {

    String id = result.getText().toString();
    myscan.resumeCameraPreview(resultHandler: this);
    validar(id);

}

```

Ilustración 75 Handle Result

Nota: La *Ilustración 75 Handle Result* es obtenida del código del proyecto.

Pasarela de pagos

En diciembre del 2020 la pasarela de pagos Khipu dejó de funcionar en Bolivia, por lo que se usará PayPal como método de pago para el presente proyecto.

Para conectar una cuenta de PayPal con una aplicación en Android, primero se crea una variable privada tipo “PaymentsClients”, después se une una variable privada tipo entera con el valor 1111 y finalmente se crea un objeto que posee toda la configuración del sistema de pagos, tiene valores como el ambiente de ejecución y el código de la cuenta de PayPal.

```

private PaymentsClient paymentsClient;
public static final int paypalcode=1111;
private PayPalConfiguration paypalConfiguration =
    new PayPalConfiguration().environment(PayPalConfiguration.ENVIRONMENT_NO_NETWORK).clientId(clientID);

```

Ilustración 76 Creación de instancias

Nota: La Ilustración 76 Creación de instancias es obtenida del código del proyecto.

En la Ilustración 77 Activar modo Sandbox se crea un objeto tipo “WalletOptions”, a este objeto se configura en un ambiente de prueba. Despues a “PaymentsClient” recibe la configuración de este objeto.

```
Wallet.WalletOptions walletOptions =  
    new Wallet.WalletOptions.Builder().setEnvironment(WalletConstants.ENVIRONMENT_TEST).build();  
paymentsClient = Wallet.getPaymentsClient( activity: this, walletOptions);
```

Ilustración 77 Activar modo Sandbox

Nota: La Ilustración 77 Activar modo Sandbox es obtenida del código del proyecto.

Cuando el usuario ingresa el monto deseado y presiona el botón para dirigiese al sistema de PayPal se ejecuta el código en la Ilustración 78 Configuración Básica. En esta parte se especifica el tipo de moneda a utilizar, el titulo de la operación y el monto de carga. Después se configura el “Intent” con toda la configuración previamente realizada y finalmente se inicia el “Intent”

```
PayPalPayment paypalPayment =  
    new PayPalPayment(new BigDecimal(a), s: "USD", s1: "Carga credito",PayPalPayment.PAYMENT_INTENT_SALE);  
Intent intent = new Intent( packageContext: this, PaymentActivity.class);  
intent.putExtra(PayPalService.EXTRA_PAYPAL_CONFIGURATION,payPalConfiguration);  
intent.putExtra(PaymentActivity.EXTRA_PAYMENT,paypalPayment);  
startActivityForResult(intent,paypalcode);
```

Ilustración 78 Configuración Básica

Nota: La Ilustración 78 Configuración Básica es obtenida del código del proyecto.

Cuando se termina la operación de la carga el sistema ejecuta la siguiente función

En la “Ilustración 79 Obtener resultados” el Activity recibe los resultados de los pagos, se generan dos condicionales “if” para verificar los datos recibidos de PayPal y en un Intent se recibe la información del pago.

```

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {

    if (requestCode == paypalcode){
        if (resultCode == RESULT_OK){

            PaymentConfirmation paymentConfirmation = data.getParcelableExtra(PaymentActivity.EXTRA_RESULT_CONFIRMATION);
            if(paymentConfirmation!=null){

                try {
                    String paydetails = paymentConfirmation.toJSONObject().toString( indentSpaces: 4);

                }catch (Exception e){

                }
            }
        }
    }

    super.onActivityResult(requestCode, resultCode, data);
}

```

Ilustración 79 Obtener resultados

Nota: La Ilustración 79 Obtener resultados es obtenida de código del proyecto.

En la Ilustración 80 Detener servicio se ve la función “onDestroy” esta función se llama cuando se cancela el pago.

```

@Override
protected void onDestroy() {

    stopService(new Intent( packageContext: this,PayPalService.class));

    super.onDestroy();
}

```

Ilustración 80 Detener servicio

Nota: La Ilustración 80 Detener servicio es obtenida del código del proyecto.

Sistema web

Para generar informes en la aplicación web, la Aplicación web utiliza la Api de Firebase para poder acceder a la base de datos con facilidad. También se utiliza la Api de Google Maps para que los usuarios puedan ver los estacionamientos en un mapa de esta forma se podrá manejar e interpretar los datos mas fácil y rápidamente.

En la “Ilustración 81 Imports web” se importan los módulos a usar en la aplicación, los principales son los módulos de Firebase y de Google Maps. En algunos casos es necesario realizar la instalación de las librerías utilizando la consola del sistema.

```
import { AngularFireModule } from '@angular/fire';
import { AngularFireAnalyticsModule } from '@angular/fire/analytics';
import { environment } from '../environments/environment';
import { AgmCoreModule } from '@agm/core';
```

Ilustración 81 Imports web

Nota: La Ilustración 81 Imports web es obtenida del código del proyecto.

Para que Angular reconozca las importaciones previas, se las tiene que especificar en la sección de “Imports” del archivo “AppModule.ts”. En esta misma sección en la línea 4 se inicializa la aplicación de Firebase. En la Ilustración 82 imports muestra a detalle el código necesario.

```
imports: [
  BrowserModule,
  AppRoutingModule,
  RouterModule.forRoot(routes),
  AngularFireModule.initializeApp(environment.firebaseio),
  AngularFireAnalyticsModule,
  FormsModule,
  AgmCoreModule.forRoot({
    apiKey: 'AIzaSyByMgvQHg-haXgn5n0S7hIjL0FeGWTcJnA'
}),
]
```

Ilustración 82 imports

Nota: La Ilustración 82 imports es obtenida del código del proyecto.

En la Ilustración 83 Firebase module se tiene las direcciones URL, apiKey, códigos de proyecto entre otras configuraciones de nuestro proyecto en Firebase. Esta sección permite a angular poder conectarse con la Api del proyecto de Firebase.

```
export const environment = {
  production: false,
  firebase: {
    apiKey: "AIzaSyByMgvQHg-haXgn5n0S7hIjL0FeGWTcJnA",
    authDomain: "parqueos-b95f8.firebaseio.com",
    databaseURL: "https://parqueos-b95f8.firebaseio.com",
    projectId: "parqueos-b95f8",
    storageBucket: "parqueos-b95f8.appspot.com",
    messagingSenderId: "331109847772",
    appId: "1:331109847772:web:b4f85479be859895be9a19",
    measurementId: "G-KDJ12H6B70"
  }
};
```

Ilustración 83 Firebase module

Nota: La Ilustración 83 Firebase module es obtenida del código del proyecto.

Para realizar consultas a la base de datos de Firebase se recibe un objeto tipo “AngularFireDatabase” en el constructor. Este objeto tiene la dirección de la base de datos para realizar las consultas. Existen dos tipos de consultas, una es obteniendo una lista de datos esta consulta se la hace con el método “List” y la otra se la realiza con el método “Object”. Esta consulta obtiene la información de la base de datos en un objeto tipo JSON. Cuando se realiza una consulta en una String se tiene que introducir la dirección al nodo de la consulta.

```
constructor(db: AngularFireDatabase) {  
  
    this.itemRef = db.object('app/parqueos');  
    this.items = db.list('app/parqueos').snapshotChanges();  
  
}
```

Ilustración 84 lectura datos

Nota: La Ilustración 84 lectura datos es obtenida del código del proyecto.

Para mostrar los datos en pantalla en una etiqueta HTML se utiliza la instancia “ngFor” que permite recorrer las lista de objetos obtenidos en formato JSON.

```
*ngFor="let a of items | async"
```

Ilustración 85 ngfor

Nota: La Ilustración 85 ngfor es obtenida del código del proyecto.

Finalmente, para mostrar los datos en la pantalla. En el HTML en una lista dentro de dos llaves se escribe los datos como se ven en la Ilustración 86 Datos

```
<li class="list-group-item"> Zona: {{ a.payload.child("zona").val() }} </li>
<li class="list-group-item">Calle: {{ a.payload.child("calle").val() }} </li>
<li class="list-group-item">Número: {{ a.payload.child("numero").val() }} </li>
```

Ilustración 86 Datos HTML

Nota: La Ilustración 86 Datos HTML es obtenido del código del proyecto.

6.3.3 Archivo Build.gradle

En esta sección se da toda la configuración de los proyectos en Android. Se cargan librerías externas y la versión de la aplicación, etc..

En la “Ilustración 87 Android” se puede apreciar que esta sección esta dedicada a toda la configuración del proyecto. En la aplicación se esta utilizando como versión mínima de Android 19. La versión Android objetivo es la 29, es la versión numero 1 del código, el nombre de la versión es 1.0 y las herramientas de compilación están en su versión 29.0.3.

```
android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"
    defaultConfig {
        applicationId "com.alcaldia.myapplication"
        minSdkVersion 19
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}
```

Ilustración 87 Android

Nota: La Ilustración 87 Android es obtenida del código del proyecto.

En la sección de dependencias se cargan todas las librerías externas a usar como se puede apreciar en la “Ilustración 88 Dependencias”. Para el proyecto se están cargando

4 librerías de Firebase para la base de datos, el sistema de autenticación, y para el almacenamiento de archivos digitales como fotos, videos y las librerías de analíticas de Firebase. También se puede apreciar que se carga la librería de Google Maps, la librería para la generación de códigos QR, entre otro tipos de archivos.

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'androidx.appcompat:appcompat:1.1.0'  
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'  
    implementation 'com.google.firebaseio:firebase-core:16.0.4'  
    implementation 'com.google.firebaseio:firebase-auth:16.0.5'  
    implementation 'com.google.android.gms:play-services-maps:16.1.0'  
    implementation 'com.google.firebaseio:firebase-database:16.0.4'  
    implementation 'com.google.android.material:material:1.0.0'  
    implementation 'androidx.preference:preference:1.1.0-alpha05'  
    implementation 'androidx.mads.library.qrgenerator:QRGenearator:1.0.3'  
    implementation 'com.google.firebaseio:firebase-storage:16.0.4'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'  
}
```

Ilustración 88 Dependencias

Nota: La Ilustración 88 Dependencias es obtenida del archivo “Build.gradle” de la aplicación Android.

6.3.4 Manifiesto de la aplicación

Las aplicaciones en Android tienen que tener el archivo “AndroidManifest.xml” exactamente con ese mismo nombre. Esta ubicado en la raíz del proyecto. Este archivo contiene información esencial de la aplicación para la creación de Android, el sistema operativo, y Google Play (Android developers, 2020).

Las principales declaraciones que contiene este manifiesto son:

- El nombre de la aplicación.
- Los componentes de la aplicación, estos incluyen actividades, servicios, receptores de emisiones y proveedores de contenido.
- Los permisos que la aplicación necesita para acceder a las partes protegidas del sistema o a otras aplicaciones.
- Los permisos de acceso al hardware que la aplicación requiera.

En la primera parte del manifiesto se está dando los permisos de la aplicación. En la Ilustración 89 Permisos se aprecian todos los permisos requeridos por el proyecto. Estos son:

- Dos permisos de Ubicación o del uso del GPS
- Permiso para el uso del internet en la aplicación.
- Permiso de almacenamiento interno.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Ilustración 89 Permisos

Nota: La Ilustración 89 Permisos es obtenida del archivo del manifiesto de la aplicación Android.

En la Ilustración 90 se están dando los detalles de la aplicación:

- Se especifica la dirección de la imagen del ícono principal
- Se da el nombre de la aplicación.
- El tema de la aplicación.

```
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="Parqueostaller"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme.NoActionBar">
```

Ilustración 90 Especificaciones

Nota: La Ilustración 90 Especificaciones es obtenida del manifiesto de la aplicación en Android.

```

<activity android:name=".ReportarProblema"/>
<activity android:name=".Factura" />
<activity android:name=".PreFactura" />
<activity android:name=".HistorialCargas" />
<activity android:name=".FinConQR" />
<activity android:name=".InicioConQR" />
<activity android:name=".Khipu" />
<activity
    android:name=".SettingsActivity"
    android:label="Settings" />
<activity android:name=".Perfil" />
<activity android:name=".Saldo" />
<activity android:name=".Confirmar_reserva" />
<activity android:name=".Vehiculos" />
<activity android:name=".Reserva" />
<activity android:name=".Registro" />

```

Ilustración 91 Activitys

Nota: La Ilustración 91 Activitys es obtenida del manifiesto de la aplicación en Android.

```

<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyByMgvQHg-haXgn5n0S7hIjL0FeGWTcJnA" />

<activity
    android:name=".MapsActivity"
    android:label="Map" />

```

Ilustración 92 Meta data 1

Nota: La Ilustración 92 Meta data 1 es obtenida del manifiesto de la aplicación.

```

<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

```

Ilustración 93 meta data 2

Nota: La Ilustración 93 meta data 2 es obtenida del manifiesto de la aplicación en Android

6.4 Estadísticas de la aplicación

Firebase tiene servicio de analíticas. En este servicio se puede monitorear el uso de la aplicación. Se llama Firebase Analytics y se integra automáticamente cuando se esta

realizando la integración a los servicios de Autenticación, Base de datos y notificaciones, el usuario no tiene que realizar ninguna cambio en el código.

Este servicio te ofrece varias tipos de gráficos que pueden ser útiles para verificar el estado y el rendimiento de la aplicación.

La Ilustración 94 Usuarios Activos es un grafico lineal que muestra la cantidad de usuarios que usan la aplicación por día. También muestra en porcentaje cuantos usuarios nuevos tiene respecto. Este grafico es un grafico dinámico, el usuario puede modificar las entradas según el desee ver las estadísticas.

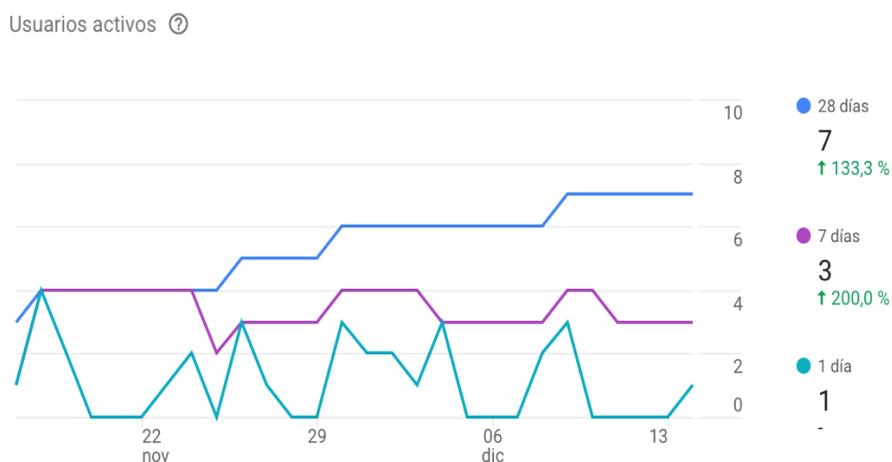


Ilustración 94 Usuarios Activos

Nota: La Ilustración 94 Usuarios Activos es obtenida de la consola de Google, las estadísticas son proyecto.

La Ilustración 95 interacción diaria es un grafico que muestra un resumen general del uso diario de la aplicación. Muestra el tiempo que los usuarios utilizan la aplicación en horas, minutos o segundos. También muestra cuales son las pantallas más vistas de la aplicación.

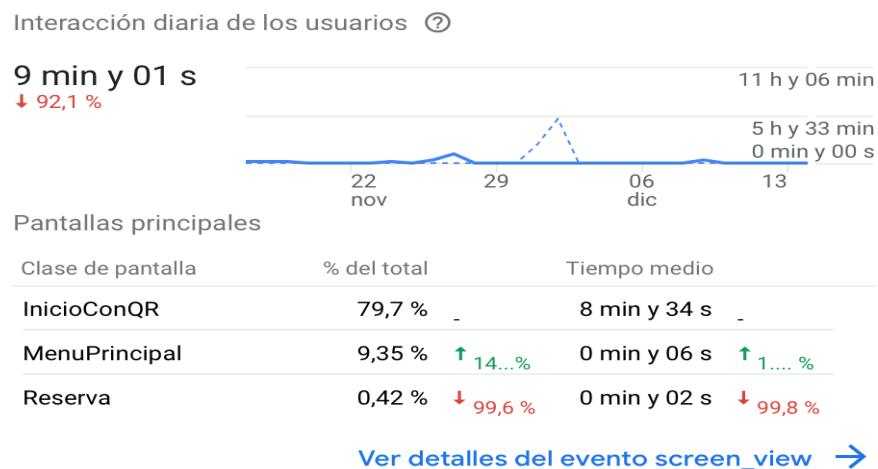


Ilustración 95 interacción diaria

Nota: La *Ilustración 95 interacción diaria* es obtenida de la consola de Google, las estadísticas de uso son las del proyecto.

El gráfico “*Ilustración 96 Dispositivos*” muestra los modelos y los sistemas operativos de los dispositivos usados por los clientes. Se muestra en porcentaje según el 100%.

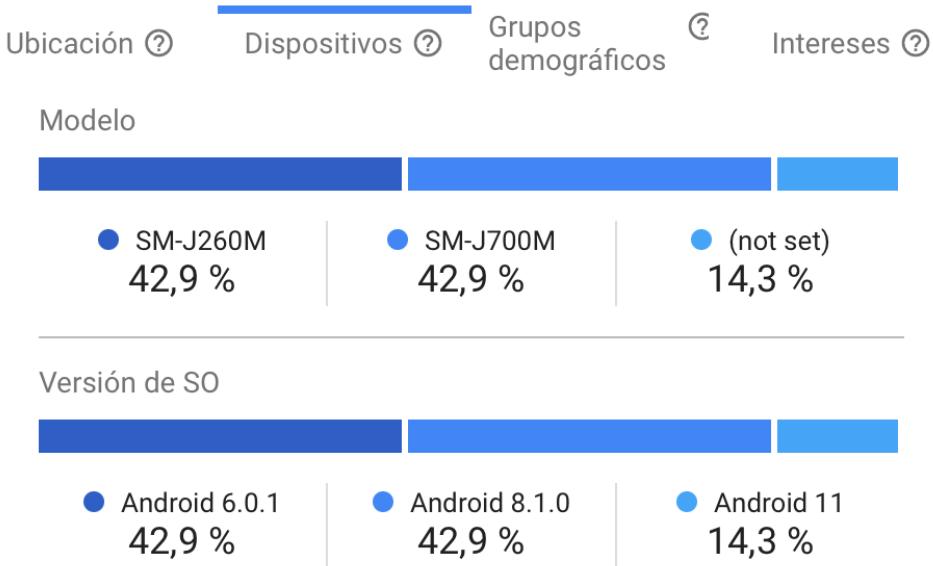


Ilustración 96 Dispositivos

Nota: La Ilustración 96 Dispositivos es obtenida de la consola de Google, las estadísticas de uso son las del proyecto.

7 Conclusiones y recomendaciones

7.1 Conclusiones

Se desarrolló un sistema de información móvil para poder implementarlo en la ciudad de La Paz. Este sistema podría permitir a los habitantes de esta ciudad descargar la aplicación y usarla para realizar la búsqueda, reserva y pago del estacionamiento que seleccionó. La aplicación utiliza una plataforma de pago electrónico proporcionada por Red Enlace que permite realizar los pagos vía internet. Esto permite a los funcionarios del municipio reducir el uso de dinero en efectivo y a los usuarios una más fácil administración de su dinero y tiempo.

El sistema que actualmente utiliza el municipio paceño es obsoleto e ineficiente. Su alto costo ha provocado grandes pérdidas, especialmente porque se debe cancelar los salarios a los controladores y cobradores en cuadras en las que muy pocos vehículos

se estacionan. Recordemos que por cada cuadra hay un cobrador y un grupo de controladores hacen rondas para inmovilizar a los vehículos que no pagan el estacionamiento.

La aplicación desarrollada está pensada en el usuario lo que la convierte en una herramienta sumamente útil y de fácil uso. No se precisan grandes explicaciones para capacitar a los usuarios en su uso. Un manual que se descargará en línea es suficiente.

Al mismo tiempo, se ha demostrado que a través de esta aplicación no sólo se mejora la administración financiera de los estacionamientos reduciendo el número de funcionarios y sistematizando el flujo de efectivo, sino que también se sientan las bases para una mejora en las condiciones laborales de quienes están a cargo del control y cobro de los estacionamientos. Esto quiere decir que, al disponer de un mayor número de personal disponible, se puede programar turnos más cortos haciendo que el trabajo en la calle no sea tan pesado.

La revisión de los logros alcanzados en torno a los objetivos específicos arroja los siguientes resultados:

Se desarrolló una aplicación móvil donde los usuarios pueden buscar estacionamientos disponibles en un mapa, reservarlos y pagarlos según un tarifario establecido. Esto presenta las siguientes ventajas:

1. Reduce el costo psicológico del usuario al informarle con precisión la disponibilidad del estacionamiento más cercano.
2. Reduce el costo en combustible pues el usuario no deberá dar vueltas para ubicar su estacionamiento.
3. Reduce el uso de efectivo haciendo que la transacción se concrete en tiempo record sin el uso de dinero en efectivo.
4. Reduce la posibilidad de pérdida de dinero ya que el usuario pagará a través de una tarjeta y el funcionario no deberá recibir dinero ni entregar un cambio.

Hay que señalar que prácticamente que la mayoría de los habitantes de la ciudad que poseen un automóvil tienen una cuenta bancaria y, en consecuencia, una tarjeta de débito.

Se desarrolló una aplicación móvil que permitirá a los funcionarios de la alcaldía escanear los códigos QR que los usuarios obtienen al realizar su reserva. Con esto se pueden realizar los controles correspondientes. Es decir, el momento en el que comienza a correr el tiempo pagado por el usuario, así como el momento en que el mismo ha concluido.

Con esta aplicación, los funcionarios también pueden realizar una reserva para usuarios que están en el estacionamiento pero que no disponen de la aplicación o de un dispositivo.

Se integro la aplicación de los usuarios a los servicios de Firebase. Se utiliza los servicios de base de datos en tiempo real, autenticación y notificaciones. La base de datos se utiliza para almacenar las coordenadas y la información de los estacionamientos y de los usuarios (registro de la placa, modelo y marca del automóvil, los datos del propietario, así como de quien esta al volante). Esta aplicación tiene los correspondientes permisos de escritura y lectura. Al ser una base de datos en tiempo real, el usuario no necesita actualizar su teléfono constantemente. Esto quiere decir que se actualiza con cada transacción. Los datos están siempre en perfecta sincronía.

Esta aplicación se integro a la Api de Google Maps y opera de manera adecuada.

La aplicación para los controladores esta integrada a la base de datos en tiempo real de Firebase. El controlador va a poder leer y escribir en la base de datos. La lectura de la base de datos se logra cuando se escanea el código QR y se verifica si el usuario realiza la reserva. También sirve para verificar si los autos estacionados en los cajones corresponden a aquellos que hicieron su reserva y cancelaron el costo.

Se Integro el sistema a la pasarela de pagos Khipu. Desde esta pasarela los usuarios pueden realizar recargas a su teléfono como si fuera una billetera móvil. De esta manera los usuarios puedan administrar su dinero fácilmente y no tienen que estar ingresando los datos de su tarjeta de crédito cada vez que desean hacer uso de un estacionamiento. Las recargas se pueden realizar cada mes o semana. Lamentablemente a partir de Diciembre del 2020 la empresa Khipu

dejo de funcionar en Bolivia por este motivo se cambio el método de pago a la plataforma de PayPal.

En caso de que el teléfono móvil del usuario o el del controlador no tengan una conexión estable o estén totalmente desconectados de Internet, la Api de Firebase permite que se realicen ciertas operaciones sin Internet. Una vez que los dispositivos se re conectan, Firebase actualiza las transacciones de manera inmediata. De esta manera, no existen inconsistencias en la base de datos. Además, el sistema permite que los usuarios puedan realizar las reservas con los controladores si es que su dispositivo esta fallando.

Se desarrolló una aplicación web que permitirá a los funcionarios de la alcaldía obtener informes empresariales sobre el uso de los estacionamientos. Esta aplicación web permite a los usuarios ver las estadísticas de uso de todos los estacionamientos de la ciudad de la paz, también permite ver los ingresos del municipio. Esta aplicación web desarrollada en Angular permitirá al municipio tener un mejor conocimiento del estado y el uso de los estacionamientos públicos.

7.2 Recomendaciones

De acuerdo a las observaciones y al análisis que se realizó durante la formulación de este proyecto, se puede realizar las siguientes recomendaciones:

1. La aplicación para el uso de los controladores es muy fácil e intuitiva de usar. Sin embargo, se recomienda que los controladores sean capacitados regularmente para que estén bien familiarizados con el uso de la aplicación y puedan superar cualquier emergencia relacionada con la misma.
2. La aplicación para los clientes es muy fácil e intuitiva de usar. Sin embargo, se recomienda que se diseñe un manual de usuario para que los usuarios puedan usar la aplicación de la mejor manera posible y tengan una mejor experiencia de usuario.
3. La base de datos de Firebase es una base de datos muy segura. Sin embargo, se recomienda que las reglas de escritura y lectura estén abiertas únicamente para los usuarios autenticados en el sistema de autenticación de Firebase para que cuando

se realice una modificación en el registro de movimientos se identifique el usuario que los realizo. Además, para que exista una mayor consistencia en los datos.

4. Actualmente se esta utilizando la versión gratuita de Firebase. Sin embargo, se recomienda actualizar a la versión paga. La versión gratuita tiene bastantes limitaciones, por lo que se recomienda cambiar a la versión de pago para que los usuarios no tengan inconvenientes en el uso de la aplicación.
5. Se recomienda no realizar alteraciones a la base de datos en ningún momento para que no exista la inconsistencia en la base de datos.
6. Se recomienda realizar constantes actualizaciones de las aplicaciones previendo posibles errores para que los usuarios tengan la mejor experiencia de usuario posible y para que la aplicación no presente fallas constantes.
7. Se recomienda que en las actualizaciones realizadas se actualicen todas las librerías utilizadas en la aplicación, para que las aplicaciones funcionen de la mejor manera.
8. Se recomienda realizar controles y estar verificando constantemente la pasarela de pagos para que este siempre disponible y no exista una incongruencia con los montos de dinero manejado en cuentas y en la base de datos.
9. Se recomienda poder utilizar más plataformas de pagos, para que los usuarios tengan otras opciones para realizar los pagos.
10. El sistema puede tiene ciertas funcionalidades sin conexión a internet. Sin embargo, para que el sistema funcione de la mejor manera se recomienda que las aplicaciones del controlador estén conectadas el mayor tiempo posible a Internet por lo que es necesario contar con el servicio de algún proveedor boliviano.
11. Algunos dispositivos Android no tienen la capacidad de lectura de archivos PDF por lo que se recomienda que los usuarios tengan una aplicación para leer archivos PDF. De esta manera podrá manejar sus facturas mas fácilmente.
12. Se recomienda que la aplicación web pueda generar más informes con diferentes tipos de gráficos y estos puedan ser plasmados en una archivo PDF.

8 Referencias

8.1 Bibliografía

- Agencia de noticias Fides. (4 de 10 de 2018). *Según datos de la Alcaldía, el sistema de parqueos genera déficit de Bs 428.500 al mes.* Obtenido de <https://www.noticiasfides.com/la-paz/segun-datos-de-la-alcaldia-el-sistema-de-parqueos-genera-un-deficit-de-bs-428500-al-mes-391776>
- Android . (2018). *Android Source.* Obtenido de <https://source.android.com/>
- Android developers. (2020). *Android developers.* Obtenido de <https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419&authuser=1>
- CGFAprende libre. (2017). *¿Qué es una aplicación móvil?* Obtenido de <https://edu.gcfglobal.org/es/informatica-basica/que-es-una-aplicacion-movil/1/>
- Coleman, J. (2011). *Google academics.* Obtenido de https://www.researchgate.net/profile/Jason_Coleman5/publication/266354609_QR_Codes_What_Are_They_and_Why_Should_You_Care/links/5564780908ae9963a12020fa.pdf
- Coplien, J. O. (2019). *A Scrum Book: The Spirit of the Game.* the pragmatic programmers.
- Delaney, J. (2018). *The Angular Firebase Survival Guide: Build Angular Apps on a Solid Foundation with Firebase.* leanpub.
- FRANCO, Y. J. (2012). *SISTEMA OPERATIVO ANDROID: CARACTERÍSTICAS Y FUNCIONALIDAD PARA DISPOSITIVOS MÓVILES.* Colombia.
- Facultad de Informática, Universidad de Murcia. (2019). *Visualización de Esquemas en Bases de Datos NoSQL basadas en documentos.* Obtenido de https://biblioteca.sistedes.es/submissions/uploaded-files/JISBD_2017_paper_71.pdf
- Firebase. (2020). *Firebase Realtime Database.* Obtenido de <https://firebase.google.com/products realtime-database?authuser=0>
- Firebase. (2020). *Firebase.* Obtenido de Firebase: <https://firebase.google.com/>

Horton, j. (2018). *Android Programming for Beginners: Build in-depth, full-featured Android 9 Pie apps starting from zero programming experience, 2nd Edition*. pakt.

IEBSChool. (2016). *Firebase, qué es y para qué sirve la plataforma de Google*. Obtenido de <https://www.iebschool.com/blog/firebase-que-es-para-que-sirve-la-plataforma-desarrolladores-google-seo-sem/>

Joshua J. Drake, Z. L. (2014). *Android Hacker's Handbook*. Willey.

Khipu. (2015). *khipu*. Obtenido de https://khipu.com/es_BO/page/bolivia-quienes-somos

La Razon. (2014). *La Razon*. Obtenido de https://m.la-razon.com/suplementos/financiero/Empresas-trabajan-moviles-encuentran-desventaja_0_2183181779.html

Pagina 7. (2017). Alcaldía planifica 928 parqueos en San Miguel y el Parque urbano. *Pagina 7*. Obtenido de <https://www.paginasiete.bo/sociedad/2017/8/26/alcaldia-planifica-parqueos-miguel-parque-urbano-149767.html>

Página Siete. (09 de 09 de 2018). *Las 19 calles donde se cobrará por estacionar*. Obtenido de <https://www.paginasiete.bo/sociedad/2017/9/9/calles-donde-cobrara-estacionar-151428.html>

Scrum. (2017). *What Is Scrum?* Recuperado el octubre de 2017, de Scrum.org: <https://www.scrum.org/resources/what-is-scrum>

Trebilcox-Ruiz, P. (2016). *Android Desde Cero: Comprendiendo Vistas y Grupo de Vistas*. Obtenido de <https://code.tutsplus.com/es/tutorials/android-from-scratch-understanding-views-and-view-groups--cms-26043>

Universidad Alicante. (2015). *Modelo vista controlador (MVC)*. Obtenido de <https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>

Univesidad Carlos III Madrid. (2018). *Arquitectura Android*. Obtenido de <https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android>

Urgente.bo. (9 de 6 de 2018). *El número de celulares en Bolivia supera al de su población, con el liderazgo de La Paz.* Obtenido de <https://urgente.bo/noticia/el-n%C3%BAmero-de-celulares-en-bolivia-supera-al-de-su-poblaci%C3%B3n-con-el-liderazgo-de-la-paz>

Weber, J. (2015). *Practical Google Analytics and Google Tag Manager for Developers.* Apress.