

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



Desarrollo de método de pago de parqueo en centros comerciales que sea seguro, accesible, y con buena experiencia de usuario.

Trabajo de graduación en modalidad de Trabajo Profesional presentado por

José Rodrigo Martínez Zúñiga

para optar el grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2021

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería



Desarrollo de método de pago de parqueo en centros comerciales que sea seguro, accesible, y con buena experiencia de usuario.

Trabajo de graduación en modalidad de Trabajo Profesional presentado por

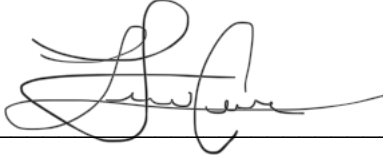
José Rodrigo Martínez Zúñiga

para optar el grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

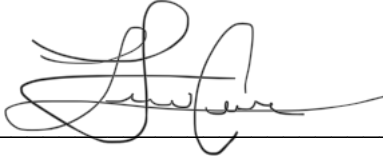
Guatemala,


2021

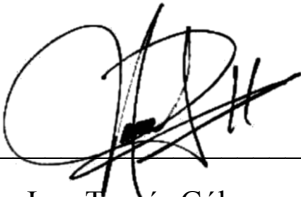
Vo.Bo.:

(f) 
Ludwing Ottoniel Cano Fuentes

Tribunal Examinador:

(f) 
Ludwing Ottoniel Cano Fuentes

(f) 
Ing. Douglas Barrios

(f) 
Ing. Tomás Gálvez

Fecha de aprobación: Guatemala, 3 de diciembre de 2021

PREFACIO

La elaboración del presente trabajo de graduación surge a partir de la identificación que los sistemas de pagos de parqueos de centros comerciales tradicionales son obsoletos, y los sistemas más innovadores son poco accesibles. También surge a partir de la necesidad de poder encontrar un método de pago que evite que las personas se coloquen en situaciones vulnerables contra el virus Covid-19.

El mayor reto en el desarrollo de este proyecto fue elaborar una interfaz de usuario que ayude a mejorar la experiencia de usuario. Esto se debe a que mi área de expertise es la programación y no el diseño gráfico, pero la retroalimentación recibida por los usuarios ayudó crear una interfaz de usuario aceptable.

El alcance de este proyecto es poder llevar el producto a un ambiente de desarrollo, en donde el producto no se encuentra integrado con componentes reales, sin embargo, se busca poder crear un producto mínimo viable con el fin de proveer una idea diferente que permita encontrar empresas interesadas en la solución.

Me gustaría agradecer a las personas que me han acompañado en el camino de mi carrera universitaria, sin ellos no estaría en donde estoy ahora. A mi asesor que siempre me retroalimentó de la mejor manera posible y que me acompañó en esta última fase de la carrera. A la Universidad del Valle de Guatemala por haberme dado herramientas que me ayudaron a crecer en el ámbito profesional. Especialmente quisiera agradecer a mi familia, que siempre me han apoyado y motivado para poder seguir adelante.

ÍNDICE

PREFACIO	v
Lista de cuadros	ix
Lista de figuras.....	x
Resumen.....	xi
I. Introducción.....	1
II. Objetivos	2
III. Alcance	3
VI. Justificación	4
V. Marco teórico	5
A. Métodos de pago existentes	5
B. Hardware.....	5
C. NFC.....	5
D. NFC en dispositivos móviles	6
E. Códigos QR.....	6
F. Internet de las cosas	6
G. Raspberry Pi.....	6
H. Servomotores	7
I. Software	7
J. Sistema operativo.....	7
K. Arquitectura de software.....	7
L. MVC	7
M. Clean Architecture	8
N. Lenguajes de programación	9
O. Frameworks	9
P. Backend	9
Q. API Rest.....	9
R. Python	10
S. Django.....	10
T. Frontend.....	10

U.	Javascript	11
V.	Flutter.....	11
W.	React.....	11
X.	UX/UI	11
Y.	Arquitectura de aplicaciones.....	11
Z.	Bases de datos.....	12
AA.	PostgreSQL.....	13
BB.	Arquitectura en la nube.....	13
CC.	Protocolo HTTP.....	13
DD.	Heroku	13
EE.	Firebase.....	13
FF.	Metodologías de desarrollo	14
GG.	Scrum.....	14
HH.	JIRA.....	15
II.	Sistemas de Versionamiento de Código.....	15
JJ.	GitFlow	15
KK.	Talanquera Amano AGP-1711	16
VI.	Antecedentes.....	18
A.	Máquina de tickets	18
B.	Compass de Credomatic	18
C.	Spectrum App	19
VII.	Metodología.....	20
A.	Encuesta previa	20
B.	Arquitectura del sistema	20
C.	Metodología ágil	20
D.	Frameworks, herramientas y plataformas a utilizar	20
E.	Funcionalidades de la aplicación móvil	21
F.	Funcionalidades del backend	21
G.	Proceso de retroalimentación del usuario	21
H.	Propuesta de integración con talanqueras reales.....	21
VIII.	Resultados	22
A.	Resultados de encuesta previa	22
B.	Resultados del análisis de la arquitectura del sistema.....	26

C.	Resultados del análisis de la metodología ágil a utilizar	29
D.	Resultados del análisis de frameworks, herramientas y plataformas a utilizar utilizadas.....	30
E.	Resultados del análisis de funcionalidades de la aplicación móvil.....	31
F.	Resultados del análisis de las funcionalidades del backend.....	32
G.	Resultados del análisis del proceso de retroalimentación del usuario	33
H.	Resultados del primer prototipo	34
I.	Resultados del segundo prototipo	36
J.	Resultados del tercer prototipo	38
K.	Propuesta de integración con talanqueras reales.....	40
L.	Costos.....	41
IX.	Análisis de resultados	43
A.	Análisis de resultados de la encuesta previa	43
B.	Análisis de resultados del primer prototipo	43
C.	Análisis de resultados del segundo prototipo.....	43
D.	Análisis de resultados del tercer prototipo.....	44
X.	Conclusiones	45
XI.	Recomendaciones	46
XII.	Bibliografía.....	47
XIII.	Anexos	50
A.	Retroalimentación escrita del primer prototipo	50
B.	Retroalimentación escrita del segundo prototipo.....	50
C.	Retroalimentación escrita del tercer prototipo.....	51
D.	Prototipo final	53

Lista de cuadros

Cuadro #1. Cuadro de comparación entre servidores On Premise y en la Nube.....	28
Cuadro #2. Comparación de costos de servidores On Premise y en la Nube.....	28
Cuadro #3. Definición de perfiles que pusieron a prueba el prototipo.....	34
Cuadro #4. Inversión inicial para ambiente de desarrollo.....	41
Cuadro #5. Inversión inicial para ambiente de producción.....	42
Cuadro #6. Retroalimentación del primer prototipo.....	50
Cuadro #7. Retroalimentación del segundo prototipo.....	51
Cuadro #8. Retroalimentación del tercer prototipo.....	52

Lista de figuras

Figura 1. Model Vista Controlador.....	8
Figura 2. Representación de capas de Clean Architecture.....	8
Figura 3. Representación gráfica de un flujo de GitFlow.....	16
Figura 4. Entradas y salidas de la talanquera.....	17
Figura 5. Gráfica de la edad de los encuestados.....	23
Figura 6. Gráfica de cantidad de visitas a centros comerciales durante la pandemia.....	23
Figura 7. Gráfica de método de pago que utilizan normalmente.....	24
Figura 8. Gráfica de satisfacción de experiencia de usuario con Máquina de centro comercial.....	24
Figura 9. Gráfica de satisfacción de experiencia de usuario con Compass de Credomatic.....	25
Figura 10. Gráfica de preferencia de tecnología para nuevo método de pago.....	25
Figura 11. Diagrama de arquitectura del sistema.....	26
Figura 12. Diagrama de red de un sistema en la nube.....	29
Figura 13. Cronograma de actividades del proyecto.....	30
Figura 14. Diagrama de entidad relación de la base de datos.....	32
Figura 15. Gráfica de calificación del User Interface del primer prototipo.....	35
Figura 16. Gráfica de calificación del User Experience del primer prototipo.....	35
Figura 17. Gráfica de calificación del funcionamiento del prototipo 1.....	36
Figura 18. Gráfica de calificación del User Interface del segundo prototipo.....	37
Figura 19. Gráfica de calificación del User Experience del segundo prototipo.....	37
Figura 20. Gráfica de calificación del funcionamiento del segundo prototipo.....	38
Figura 21. Gráfica de calificación del User Interface del tercer prototipo.....	39
Figura 22. Gráfica de calificación del User Experience del tercer prototipo.....	39
Figura 23. Gráfica de calificación del funcionamiento del tercer prototipo.....	40
Figura 24. Pantalla de On Boarding, paso #1.....	53
Figura 25. Pantalla de On Boarding paso #2.....	54
Figura 26. Pantalla de On Boarding paso #3.....	55
Figura 27. Pantalla de On Boarding paso #4.....	56
Figura 28. Pantalla de registro del usuario.....	57
Figura 29. Pantalla de inicio de sesión del usuario.....	58
Figura 30. Pantalla de olvidé mi contraseña.....	59
Figura 31. Pantalla de Home con tutorial.....	60
Figura 32. Pantalla de Home.....	61
Figura 33. Lectura NFC activada en el dispositivo.....	62
Figura 34. Pantalla de historial de transacciones del usuario.....	63
Figura 35. Pantalla de ajustes.....	64
Figura 36. Pantalla de administración de tarjetas de crédito del usuario.....	65
Figura 37. Pantalla para agregar una tarjeta de crédito.....	66

Resumen

Este trabajo describe el proceso de desarrollo de un nuevo método de pago de centros comerciales que sea accesible, seguro, y con buena experiencia de usuario. Utilizando tecnologías innovadoras para poder crear un producto que tenga un valor agregado a diferencia de la competencia. Con la finalidad de poder presentar un producto mínimo viable que en un futuro pueda ser integrado con componentes reales utilizados en los parqueos de centros comerciales. Utilizando una aplicación móvil que se encargará de leer NFC Tags; de esta manera, utilizando la arquitectura de cliente-servidor, poder enviar señales para poder abrir una talanquera. Para mejorar la experiencia de usuario del producto final se recibió retroalimentación de usuarios voluntarios sobre tres aspectos importantes: funcionamiento, experiencia de usuario, e interfaz de usuario. Todos los usuarios que pusieron a prueba el producto desarrollado dieron una calificación satisfactoria a la solución propuesta. Reconociendo esta solución como una posible alternativa para poder pagar parqueos de centros comerciales.

I. Introducción

En los últimos años se ha tratado de atacar el problema de la experiencia de usuario en los parqueos de centros comerciales; esto se debe a que los métodos de pago tradicionales se consideran como obsoletos y los más innovadores son poco accesibles. Hoy en día existen varios métodos de pago, pero cada uno tiene sus propias limitaciones. El problema en general es que no existe un método de pago que tenga las siguientes características: 1) Completamente seguro (financieramente y/o sanitariamente), 2) Buena experiencia de usuario, y 3) Accesible para todos (no estar asociado a un cobro directo con un banco).

Este trabajo consiste en el desarrollo de un nuevo método de pago que tenga las características previamente mencionadas. Estudiando las soluciones que existen hoy en día para poder identificar sus funcionalidades efectivas y deficientes. Actualmente existen tres soluciones principales: pago con tarjeta de papel y máquina de pago, Compass de Credomatic, y la Spectrum App.

La solución desarrollada está conformada por un sistema cliente-servidor en donde un dispositivo móvil lee un NFC Tag para poder indicarle al servidor la talanquera en la que se encuentra el usuario. Al recibir esta solicitud, el servidor se comunica con una Raspberry Pi por medio del Firestore de Firebase para que un servomotor se mueva *simulado* la apertura de una talanquera. Dependiendo si el usuario se encuentra en una talanquera de entrada o salida se abre o se cierra una transacción respectivamente. Por medio de la aplicación móvil el usuario puede ver su historial de transacciones y también puede manejar sus tarjetas de crédito con el que se hará el cobro del parqueo.

II. Objetivos

A. Generales:

- Desarrollar un método de pago para parqueos de centros comerciales que sea: completamente seguro, accesible para todos, y que tenga una buena experiencia de usuario.

B. Específicos:

- Desarrollar una aplicación móvil que utilice tecnología NFC para ingresar, pagar y salir de un parqueo.
- Desarrollar un Backend que maneje solicitudes de ingreso, pago y salida de un parqueo de centro comercial.
- Desarrollar un simulador de talanquera de un centro comercial utilizando una Raspberry Pi.

III. Alcance

El proyecto realizado es una propuesta de un nuevo método de pago para parqueos de centros comerciales. Limitado para un ambiente de desarrollo, es decir, no para llevarlo a un ambiente de producción. Esto se debe a que para llevarlo a un ambiente de producción es necesario pasar por un proceso de certificación con Visa y también ponerse en contacto con los gerentes de los centros comerciales para poder integrar la solución en sus parqueos. Los últimos dos puntos son procesos que pueden tomar varios meses, estos objetivos no pueden ser completados con el tiempo de desarrollo del proyecto establecido.

VI. Justificación

Desde el comienzo de la pandemia muchos negocios fueron afectados, nuevos protocolos fueron creados y nuevas medidas fueron tomadas en cuenta. En el caso de los centros comerciales muchas medidas se llevaron a cabo, desde limitar la capacidad de personas visitantes hasta ofrecer gel antibacterial en cada tienda. Sin embargo, muchos centros comerciales no fueron capaces de cumplir con todas las medidas preventivas por el simple hecho de tener una falta de conocimiento y preparación. En diciembre del año 2020 cuatro centros comerciales del área capitalina recibieron avisos por parte del Ministerio de Salud Pública y Asistencia Social por *incumplimiento en el aforo del parqueo*. Los centros comerciales sobre pasaron la cantidad de personas permitidas en los parqueos, lo que significa que muchas personas tocaron el botón de la talanquera para recibir su tarjeta de parqueo y muchas personas tocaron las máquinas de pago, lo cual crea una **vulnerabilidad sanitaria** para los visitantes.

Hoy en día existen diferentes soluciones para el pago de parqueo *sin contacto*, como el *Compass* de Credomatic o la *Spectrum App*, pero cada una tiene sus limitaciones. El *Compass* es una solución innovadora, ya que utiliza un sensor para poder registrar el ingreso y salida de un visitante, pero lamentablemente solo se encuentra disponible para usuarios exclusivos que cuentan con una tarjeta de crédito con el Banco Credomatic, **no está disponible para todos los usuarios**. Además, el sensor viene en una calcomanía para colocar en la ventana frontal del automóvil, lo que significa que solo se puede utilizar un sensor por automóvil.

Por otro lado, se encuentra la *Spectrum App* la cual cuenta con varias funcionalidades muy interesantes, como poder localizar el automóvil de un visitante en el parqueo del centro comercial. Otra de las funcionalidades que tiene esta aplicación móvil es poder pagar el parqueo directamente utilizando cualquier tarjeta de crédito, pero según los usuarios esta funcionalidad **no funciona correctamente**. Una desventaja que comparten las soluciones de *Spectrum App* y el *Compass* de Credomatic es que no se encuentran disponibles en todos los centros comerciales de la ciudad.

V. Marco teórico

Actualmente existen dos diferentes soluciones que se relacionan con el problema. El Compass de Credomatic que utiliza un chip NFC para poder ingresar y salir de un parqueo de centro comercial, y la Spectrum App que utiliza la lectura de placas para realizar la misma acción. Dentro de la Ciudad de Guatemala existen por lo menos 25 centros comerciales que utilizan talanquera para su ingreso y salida, entre ellos se encuentran:

- Paseo Cayalá
- Oakland Mall
- Plaza Fontabella
- Pradera Concepción
- Pradera Zona 10

Dado a que el trabajo se centra en crear un nuevo método de pago para parqueos de centros comerciales que tenga una mejor experiencia de usuario es necesario definir conceptos y metodologías que sirvan para entender mejor el trabajo elaborado.

A. Métodos de pago existentes

En la actualidad existen dos métodos de pago tradicionales para los parqueos de centro comercial:

- Efectivo: El diccionario de la Real Academia Española lo define como “*dicho del dinero: en monedas o billetes*”.
- Tarjetas de crédito: es un documento de material plástico o metal emitido por un banco o institución especializada a nombre de una persona, que podrá utilizarla para efectuar compras sin tener que pagar en efectivo

B. Hardware

El hardware se define como todos los componentes materiales y físicos de un dispositivo. Una computadora se conforma por varios componentes de hardware que funcionan en conjunto. Algunos de los componentes de hardware conocidos son: Procesador, Tarjeta Madre, RAM, Tarjeta Gráfica, etc. Los dispositivos móviles también se conforman por componentes de hardware, los más recientes tienden a tener más componentes modernos. Uno de los componentes modernos que contienen los dispositivos móviles es el chip NFC.

C. NFC

El término NFC significa Near Field Communication. Es una tecnología inalámbrica de alta frecuencia que funciona en la banda de los 13.56 MHz, y que deriva de las etiquetas RFID, las cuales ayudan a que un objeto pueda ser detectable de forma inalámbrica.

D. NFC en dispositivos móviles

Los dispositivos móviles más modernos contienen chips NFC. Esta tecnología tiene diferentes usos:

- **Identificación:** Puede utilizarse para tener acceso a lugares en donde se necesita tener una identificación.
- **Intercambio de datos:** Utilizando las etiquetas RFID se pueden recibir datos de manera instantánea.
- **Sincronización instantánea de dispositivos:** Algunos fabricantes utilizan esta tecnología para poder sincronizar los dispositivos móviles con sus dispositivos.
- **Automatización de acciones:** Algunas de las etiquetas RFID se pueden configurar para que, cuando se pase el dispositivo móvil sobre ellas, realicen configuraciones automáticas.
- **Pago con el dispositivo móvil:** En algunos dispositivos móviles se puede vincular una tarjeta de crédito para poder realizar pagos.

E. Códigos QR

Los códigos QR (en inglés “Quick Response”, “respuesta rápida”) son códigos de barras bidimensionales, que fueron desarrollados por la compañía japonesa Denso Wave, en 1994. En un código QR se almacenan datos codificados. La mayoría del tiempo los datos es un enlace a un sitio web (URL). La matriz de puntos en la que se guardan los datos no es legible para el ojo humano. Se debe leer con un teléfono móvil o con un dispositivo que disponga de la aplicación correspondiente (un lector de códigos QR). La lectura del código se lleva a cabo en cuestión de segundos. Además, gracias a la corrección de errores, la lectura también funciona si falta alguna pieza en el código.

F. Internet de las cosas

El Internet de las cosas (IoT) es el proceso que conecta elementos físicos cotidianos al Internet: desde objetos domésticos, como las bombillas de luz, hasta recursos para la atención de la salud, como los dispositivos médicos. En el desarrollo de proyectos es muy común utilizar dispositivos para poder realizar la conexión entre los elementos físicos con el internet, unos de estos dispositivos es la Raspberry Pi.

G. Raspberry Pi

Una Raspberry Pi es una computadora del tamaño de una tarjeta de crédito. Su placa tiene la capacidad de tener componentes que utiliza una computadora normal. Este puede ser utilizado para muchas de las cosas que una PC de escritorio hace, como hojas de cálculo, procesadores de texto, juegos, y los más importante crear programas de código. También cuenta con un set de pines digitales y analógicos con el que se pueden conectar y controlar componentes de electrónica.

H. Servomotores

Un servomotor es un dispositivo eléctrico autónomo que gira partes de una máquina con alta eficiencia y con gran precisión. El eje de salida de este motor se puede mover a un ángulo, posición y velocidad particulares que un motor normal no tiene.

I. Software

El software es un programa o conjunto de programas de cómputo. La interacción entre el software y el hardware hace operativo una computadora. Comúnmente se utiliza este término para referirse de una forma muy genérica a los programas de un dispositivo informático.

J. Sistema operativo

Un sistema operativo es el software que maneja el hardware. Comprende un conjunto de programas que controla el funcionamiento del componente físico, facilitando al usuario el uso de la computadora u otro equipo. El sistema operativo administra los recursos del hardware y actúa como un intermediario entre la computadora y su usuario. Además, proporciona un ambiente en donde el usuario pueda ejecutar programas en una forma conveniente y eficiente.

K. Arquitectura de software

La arquitectura de software es la estructuración un sistema que se crea en etapas tempranas del desarrollo. Esta estructuración representa un diseño de alto nivel del sistema que tiene dos propósitos primarios: satisfacer los atributos de calidad (desempeño, seguridad, adaptabilidad a cambios), y servir como guía en el desarrollo. Sin una arquitectura puede llegar a ser muy complicado satisfacer estos propósitos. Los diseños arquitectónicos que se crean en una organización pueden ser reutilizados para crear sistemas distintos. Esto permite reducir costos y aumentar la calidad, sobre todo si dichos diseños han resultado previamente en sistemas exitosos. Entre las arquitecturas de software más famosas se encuentran: MVC (Modelo-Vista-Controlador) y Clean Architecture.

L. MVC

MVC (Modelo-Vista-Controlador) es una arquitectura de software comúnmente utilizado para implementar interfaces de usuario, datos y lógica de control. Enfatiza una separación entre la lógica de negocios y su visualización. Esta separación ayuda mucho a mejorar la división del trabajo y el mantenimiento del código. Las tres partes de la arquitectura se pueden describir de la siguiente manera:

- Modelo: Son las clases que definen los datos de nuestra aplicación.
- Vista: Es la interfaz propiamente dicha en definitiva el punto de entrada del usuario a nuestro producto.

- Controlador: Son las clases en las que se van a realizar la captura de los eventos que suceden en la interfaz actuando en consecuencia. Además, de realizarse las interacciones con el modelo de la aplicación.

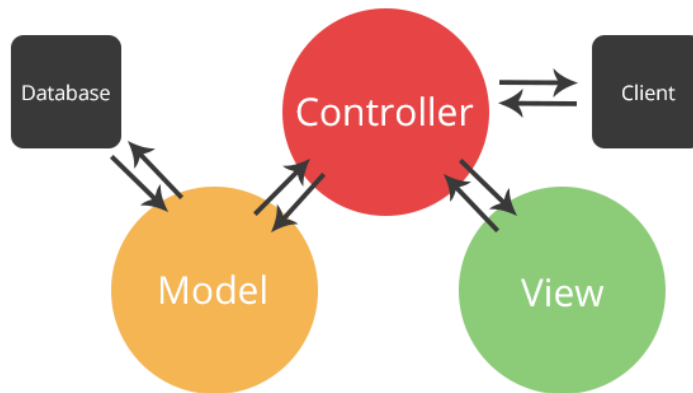


Figura 1. Model Vista Controlador

M. Clean Architecture

Clean Architecture es una arquitectura que se basa en la premisa de estructurar el código en capas contiguas, es decir, que solo tienen comunicación con las capas que están inmediatamente a sus lados. Cada capa debe realizar sus propias tareas y se comunica únicamente con sus niveles inmediatamente contiguos. Las capas que conforman Clean Architecture son las siguientes:

- Interfaz de usuario: Es la capa con la que el usuario interactúa directamente.
- Presentadores: Son las clases que se subscriben a los eventos generados por la interfaz de usuario y que responden en consecuencia.
- Casos de uso: Clases que representan la evaluación de reglas de negocio y toma de decisiones.
- Entidades: Son los modelos de datos de la aplicación.

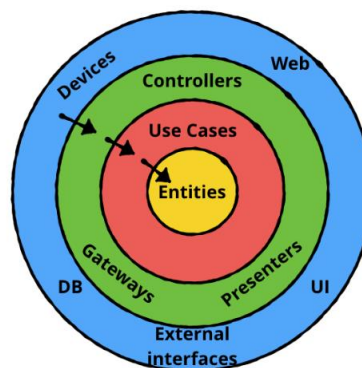


Figura 2. Representación de capas de Clean Architecture

N. Lenguajes de programación

Un lenguaje de programación es un lenguaje formal que, mediante una serie de instrucciones le permite a un programador escribir un conjunto de órdenes, acciones consecutivas, datos y algoritmos para crear programas que controlen el comportamiento físico y lógico de una computadora.

O. Frameworks

Un framework es un entorno de trabajo que tiene como objetivo facilitar la labor de programación ofreciendo una serie de características y funciones que aceleran el proceso, reducen los errores, favorecen el trabajo colaborativo y consiguen obtener un producto de mayor calidad. Los framework ofrecen una estructura para el desarrollo y no tienen que estar sujetos a un único lenguaje de programación.

P. Backend

El backend se encarga de todos los procesos necesarios para que una aplicación móvil funcione de forma correcta. Estos procesos o funciones no son visibles, pero tienen mucha importancia en el buen funcionamiento de una aplicación móvil. Una de las acciones que controla el backend es la conexión con la base de datos. Actualmente existen muchos lenguajes de programación y frameworks que se pueden utilizar para construir un backend, entre los lenguajes más comunes se encuentran:

- Python
- C#

Algunos de los frameworks conocidos son los siguientes:

- Django: Este es un framework que utiliza el lenguaje de programación Python, comúnmente es utilizado para construir páginas web, pero también es utilizado para desarrollar un API Rest.
- .NET Core: Es la plataforma de desarrollo de Microsoft más moderna y depende totalmente del sistema operativo de Windows. Es un framework que permite construir todo tipo de aplicaciones, incluyendo un API Rest.

Q. API Rest

Es un conjunto de solicitudes que permite la comunicación de datos entre aplicaciones utilizando el protocolo HTTP. Las principales solicitudes que se utilizan son:

- POST: Utilizada para crear datos en el servidor.

- GET: Utilizada para la lectura de datos en el servidor.
- DELETE: Utilizada para borrar datos en el servidor.
- PUT: Utilizada para actualizar datos en el servidor.

R. Python

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. Este es un lenguaje interpretado, lo cual significa que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador.

S. Django

Django es un framework de aplicaciones web gratuito y de código abierto (open source) escrito en Python. Este framework es muy conocido porque ayuda a desarrollar proyectos de manera rápida y segura. Django se puede complementar utilizando librerías de terceros, una de las más populares es la librería de Rest Framework el cuál ayuda a crear API Rest dentro de la aplicación de una manera sencilla.

T. Frontend

Frontend es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo móvil que corren en el dispositivo y que se encargan de la interactividad con los usuarios.

Actualmente existen dos tipos de aplicaciones móviles: Nativas e Híbridas. La principal diferencia entre ambos es que para las aplicaciones nativas hay que programar una aplicación por sistema operativo (Android e iOS), y con las aplicaciones híbridas solo hay que programar una aplicación para ambos sistemas operativos.

Existen diferentes frameworks para poder desarrollar aplicaciones híbridas, entre los más populares se encuentran:

- Flutter: Este framework fue desarrollado por Google, utiliza el lenguaje de Dart. Su rendimiento es casi nativo.
- Ionic: Utiliza tecnologías web (HTML, CSS, JS) para poder desarrollar aplicaciones móviles. Este framework es basado en un Web View, por lo que su rendimiento no se acerca a un rendimiento nativo.
- React Native: Este framework fue desarrollado por Facebook, utiliza el lenguaje de Javascript. A pesar de utilizar una tecnología web el rendimiento de una aplicación desarrollada con este framework es casi nativo, esto se debe a que en la compilación todos los componentes visuales programados en Javascript se transforman en componentes nativos.

U. Javascript

Javascript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web. Es el único lenguaje de programación que funciona en los navegadores de forma nativa.

V. Flutter

Flutter es un framework de desarrollo de aplicaciones híbridas, es decir, se escribe un código para dos plataformas (Android e iOS). Este es desarrollado por Google y fue lanzado en el año 2018. A pesar de que es un framework relativamente nuevo su comunidad ha crecido exponencialmente, superando a su más grande competidor (React Native). El lenguaje que utiliza Flutter es Dart, un lenguaje desarrollado por Google específicamente para ser utilizado en este framework. Flutter se destaca por tres características principales:

- Desarrollo rápido: Utilizando la funcionalidad de Hot Reload en el desarrollo; una aplicación se puede actualizar en cuestión de milisegundos sin perder el estado actual.
- UI Expresiva y Flexible: La arquitectura en capas permite una completa personalización, que resultan en un renderizado increíblemente rápido y diseños expresivos y flexibles.
- Rendimiento nativo: El proceso de compilación de Flutter incluye poder traducir los widgets programados a componentes nativos; esto ayuda a tener un rendimiento casi nativo.

W. React

React es una librería open source de JavaScript para desarrollar interfaces de usuario en plataformas web. Esta fue desarrollada por Facebook en el año 2013. Se considera como solamente la “V” del Modelo Vista Controlador, es por esa razón que React necesita ser acompañado de otras librerías para completar las necesidades de desarrollo de nuestra aplicación web.

X. UX/UI

UX, también conocido como User Experience o Experiencia de Usuario, es un concepto que se enfoca en cómo un usuario se siente mientras utiliza un producto o servicio digital. UI, también conocido como User Interface o Interfaz del Usuario, se enfoca en encontrar la mejor manera de guiar al usuario dentro de la aplicación. A pesar de que ambos son diferentes, están íntimamente conectados. Es recomendado utilizar ambas en una aplicación para poder obtener los mejores resultados.

Y. Arquitectura de aplicaciones

Una arquitectura de aplicaciones describe los patrones y las técnicas que se utilizan para diseñar y desarrollar aplicaciones. La arquitectura le proporciona un plan y las prácticas recomendadas que debe seguir al momento de diseñar una aplicación, de modo que obtenga una aplicación bien estructurada. En una arquitectura de aplicaciones, habrá servicios de frontend y de backend. La arquitectura es el punto de partida o el roadmap para diseñar una aplicación.

Una arquitectura muy popular es Cliente-Servidor, en donde el cliente se encarga de solicitar información y el servidor se encarga de disponer información. En esta arquitectura se definen tipos de agentes dependiendo de lo que se haga con la información; los tipos de agentes son los siguientes:

- Activo: El Agente realiza un procesado activo de la información.
- Pasivo: El agente se limita a manejar información en bruto o preprocesada.

La arquitectura cliente-servidor tiene diferentes tipos:

- Cliente activo, Servidor pasivo
- Cliente pasivo, Servidor pasivo
- Cliente pasivo, Servidor activo
- Cliente activo, Servidor activo

En este proyecto se utilizará el tipo de Cliente pasivo, Servidor activo. En este tipo el cliente solo se encarga de solicitar y obtener datos, mientras que el servidor se encarga de procesar los datos.

Z. Bases de datos

Una base de datos es una colección organizada de información estructurada, o datos, típicamente almacenados electrónicamente en un sistema de computadora. Existen diferentes tipos de bases de datos:

- Relacional: Los elementos de una base de datos relacional se organizan como un conjunto de tablas con columnas y filas. La tecnología de base de datos relacional proporciona la manera más eficiente y flexible de acceder a información estructurada.
- Documental: Diseñadas para almacenar, recuperar y administrar información orientada a documentos, las bases de datos documentales son una forma moderna de almacenar datos en formato JSON en lugar de filas y columnas.

Existe una variedad de gestores de bases de datos para tipo relacional, entre las más utilizadas se encuentran:

- MySQL
- PostgreSQL
- SQLite
- SQL Server

En este proyecto se utilizará el gestor de PostgreSQL.

AA. PostgreSQL

Es un sistema de código abierto de administración de bases de datos del tipo relacional, aunque también es posible ejecutar consultas que sean no relaciones. En este sistema, las consultas relacionales se basan en SQL, mientras que las no relacionales hacen uso de JSON. Dos detalles para destacar de PostgreSQL es que posee tipos de datos avanzados y permite ejecutar optimizaciones de rendimiento avanzadas, que son características que por lo general solo se ven en sistemas de bases de datos comerciales.

BB. Arquitectura en la nube

La nube es una red mundial de servidores, cada uno con una función única. La nube no es una entidad física, sino una red enorme de servidores remotos de todo el mundo que están conectados para funcionar como un único ecosistema. Estos servidores están diseñados para almacenar y administrar datos, ejecutar aplicaciones o entregar contenido o servicios, como streaming de vídeos, correo web, software de ofimática o medios sociales. Existen cuatro tipos de nube:

- Nube pública: Este tipo de nube ofrece sus servicios a cualquier usuario de internet.
- Nube privada: Estas nubes ofrecen sus servicios a un número limitado de usuarios a través de una red de una empresa.
- Nube híbrida: Este tipo de nube es fruto de una combinación de las dos anteriores.
- Multicloud: Se trata de una opción que está siendo adoptada por muchas entidades y que consiste en varias nubes entre las que se desplazan los distintos servicios y que pueden trabajar simultáneamente.

CC. Protocolo HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo de comunicación el cual permite realizar una petición de datos y recursos. Es la base de cualquier intercambio de datos en la Web, y un protocolo de estructura cliente-servidor, esto quiere decir que una petición de datos es iniciada por el elemento que recibirá los datos (el cliente), normalmente un navegador Web.

DD. Heroku

Es una plataforma en la nube que permite a las empresas construir, entregar, supervisar aplicaciones y alojarlas en la nube. Esta herramienta se reconoce como un Platform as a Service, es uno de los más utilizados en la actualidad en entornos empresariales por su fuerte enfoque en resolver el despliegue de una aplicación.

EE. Firebase

Es una plataforma en la nube para el desarrollo de aplicaciones web y móvil, fue desarrollada por Google en el 2014. Su principal función es proporcionar bases de datos a tiempo

real. Estas se alojan en la nube, son No SQL y almacenan los datos como JSON. Permiten alojar y disponer de los datos e información de la aplicación en tiempo real, manteniéndolos actualizados, aunque el usuario no realice ninguna acción. Firebase envía automáticamente eventos a las aplicaciones cuando los datos cambian, almacenando los datos nuevos en el disco.

FF. Metodologías de desarrollo

Las metodologías de desarrollo de software son un conjunto de técnicas y métodos organizativos que se aplican para diseñar soluciones de software informático. El objetivo de las distintas metodologías es el de intentar organizar los equipos de trabajo para que estos desarrollen las funciones de un programa de la mejor manera posible. Existen dos categorías de metodologías: la tradicional y la ágil.

Las metodologías de desarrollo de software tradicionales se caracterizan por definir total y rígidamente los requisitos al inicio de los proyectos de ingeniería de software. Los ciclos de desarrollo son poco flexibles y no permiten realizar cambios, al contrario que las metodologías ágiles; lo que ha propiciado el incremento en el uso de las segundas. Las principales metodologías tradicionales son:

- Cascada
- Prototipado
- Espiral
- Incremental

Las metodologías ágiles de desarrollo de software son las más utilizadas hoy en día debido a su alta flexibilidad y agilidad. Los equipos de trabajo que las utilizan son mucho más productivos y eficientes, ya que saben lo que tienen que hacer en cada momento. Además, la metodología permite adaptar el software a las necesidades que van surgiendo por el camino, lo que facilita construir aplicaciones más funcionales. Las principales metodologías ágiles son:

- Scrum
- Kanban
- Lean

GG. Scrum

La metodología Scrum es una metodología de trabajo ágil que tiene como finalidad la entrega de valor en períodos cortos de tiempo y para ello se basa en tres pilares: la transparencia, inspección y adaptación. En Scrum existen tres roles muy importantes: Product Owner, Scrum Master y Equipo de desarrollo. El desarrollo de Scrum es iterativo, llevado a cabo por medio de varios Sprints, en donde cada uno contiene:

- Sprint Planning
- Daily Meeting
- Sprint Review

- Sprint Retrospective

HH. JIRA

Existen muchas herramientas que ayudan a facilitar la gestión de Scrum, una de estas herramientas es Jira. Jira es una herramienta que sirve para la gestión de trabajo, desde la planificación del trabajo a realizar, hasta la propia gestión de requisitos y casos de prueba. Se suele utilizar para la gestión del desarrollo de software en equipos que trabajan con metodologías ágiles. Esta herramienta fue desarrollada por una empresa australiana llamada Atlassian. Jira ha logrado adaptarse las empresas y equipos que realmente quieren trabajar con Agile, ya sea utilizando la metodología Scrum, Kanban u otras. Además, puede adaptarse a empresas que no tienen nada que ver con el mundo del desarrollo de software, ya que su flujo de procesos también permite adecuarse a diferentes industrias y necesidades.

II. Sistemas de Versionamiento de Código

Un Sistema de Versionamiento de Código (SVC) permite compartir el código fuente de nuestros desarrollos y a la vez mantener un registro de los cambios por los que va pasando. Existen dos tipos de sistemas:

- Sistemas Centralizados: Son los más "tradicionales"; por ejemplo, SVN, CVS, etc.
- Sistemas Distribuidos (o descentralizados): son los que están en auge actualmente como: Git, Mercurial, Bazaar, etc.

Actualmente existen varios servicios que funcionan con Git, por ejemplo: GitHub, GitLab, BitBucket, entre otros. Estos sistemas tienen la capacidad de tener diferentes ramas dentro de un repositorio, de esta manera agilizar el desarrollo dentro del equipo de trabajo.

JJ. GitFlow

Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales. Este modelo es una especie de idea abstracta de un flujo de trabajo de Git. Esto quiere decir que ordena qué tipo de ramas se deben configurar y cómo fusionarlas. Dentro de este flujo se tienen dos ramas principales: Main y Develop. Todo lo que debe de estar en estado de producción debe de estar en la rama Main, y todo lo que se encuentre en un estado de pruebas debe de estar en la rama Develop. Por otro lado, existen tipos de rama que ayudan al desarrollo del proyecto:

- Feature: Este tipo de rama se utiliza para agregar una nueva funcionalidad al proyecto. Siempre debe de crearse a partir de la rama Develop.
- Bugfix: Este tipo de rama se utiliza para realizar correcciones de un desarrollo. Siempre debe de crearse a partir de la rama Develop.

- Release: Esta rama funciona como un intermediario entre la rama Develop y Main. Comúnmente se utiliza para colocar los cambios de la rama Develop hacia la rama Main.
- Hotfix: Este tipo de rama para realizar correcciones de lo que se encuentra en producción. Siempre debe de crearse a partir de la rama Main. Es recomendable utilizar este tipo de rama si la corrección a realizarse es urgente y pequeña, de lo contrario es recomendable realizar la corrección utilizando una rama de tipo Bugfix.



Figura 3. Representación gráfica de un flujo de GitFlow.

KK. Talanquera Amano AGP-1711

En el mercado existen muchos modelos de talanqueras que los centros comerciales pueden instalar en sus entradas y salidas de sus parqueos. En el centro comercial de Paseo Cayalá se utiliza el modelo Amano AGP-1711. Este es un modelo con un diseño compacto, detector de doble canal integrado a un microprocesador lógico y funcional. Esta talanquera cuenta con un circuito con entradas y salidas que pueden ser utilizadas para poder controlar la talanquera por un dispositivo externo.

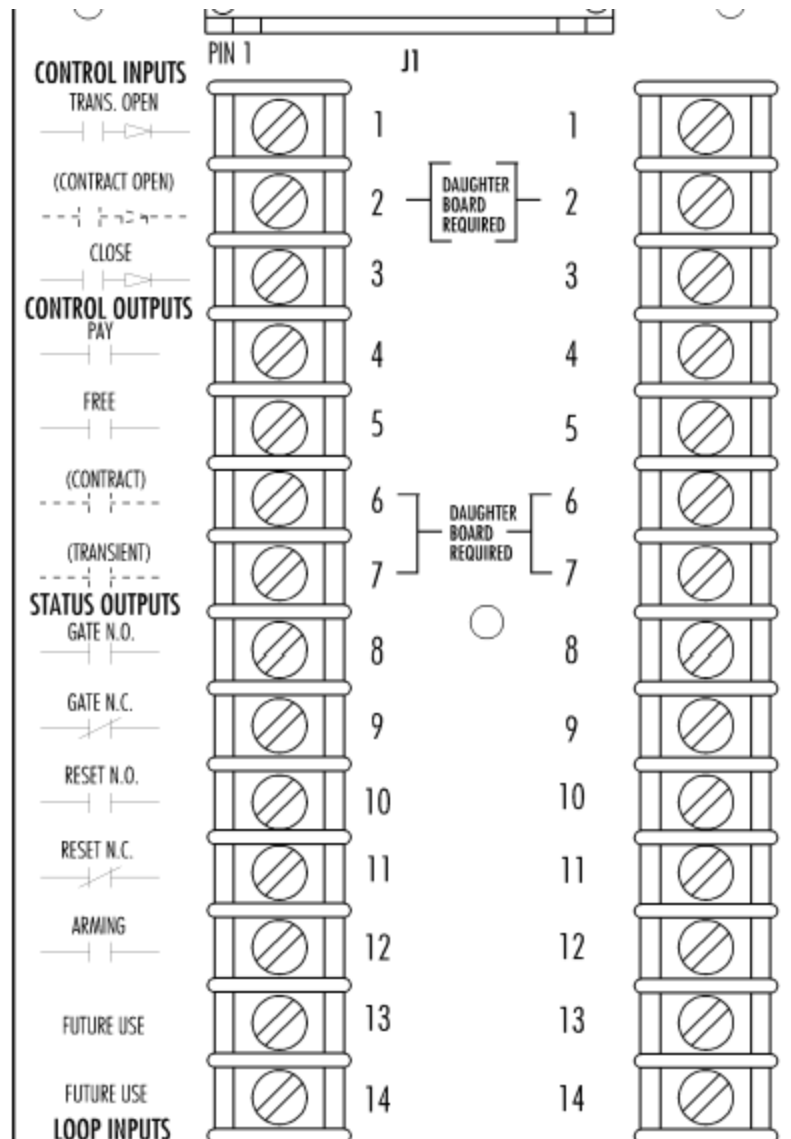


Figura 4. Entradas y salidas de la talanquera.

Esta talanquera puede configurarse en diferentes modos de uso:

- One Way Pay Direction
- Free Direction Only
- Two Way Operation

Cada uno de estos modos de uso representan la manera en la que el vehículo interactúa con la talanquera. El modo que se utilizará en este proyecto es el One Way Pay Direction. En este modo se le puede integrar un dispositivo tercero para poder controlar la talanquera; este dispositivo puede ser un dispensador de tickets, un lector de tarjetas, o en el caso de este proyecto una Raspberry Pi.

VI. Antecedentes

En algunas partes del mundo se han llegado a topar con el problema de que no existe un método de pago que sea eficiente y con buena experiencia de usuario. Algunos desarrolladores en Hawaii han creado Parklinq, esta aplicación permite que las empresas y las personas registren sus espacios de estacionamiento privados y entradas para vehículos y los alquilen durante períodos en los que, de otro modo, no se utilizarían. Para usar el servicio, los automovilistas buscan y reservan el espacio de estacionamiento registrado en Parklinq más cercano a su destino que esté disponible para el tiempo que desean estacionar. Después de que terminan de utilizar el espacio reservado los usuarios deben pagar por su lugar tocando una 'estación inteligente' sin contacto con su teléfono inteligente NFC.

En Guatemala existen diferentes métodos de pago de parqueos de centros comerciales. Entre los más comunes se encuentran:

A. Máquina de tickets

Este es un método de pago muy común en los centros comerciales, ya que lo único que necesita el usuario para utilizarlo es dinero en efectivo o tarjetas de crédito. El proceso para utilizarlo es el siguiente:

1. El usuario llega al centro comercial con su vehículo y presiona un botón en la talanquera para poder obtener un ticket.
2. El usuario permanece en el centro comercial el tiempo que éste desee.
3. Cuando el usuario decida retirarse éste debe dirigirse a una máquina de pago.
4. En la máquina de pago el usuario ingresa el ticket para poder verificar el monto a pagar por parquarse dentro del centro comercial.
5. El usuario realiza el pago utilizando su tarjeta de crédito o ingresando dinero en efectivo dentro de la máquina.
6. El usuario se dirige a una talanquera de salida con su vehículo e ingresa el ticket dentro de otra máquina.
7. El usuario sale del centro comercial.

B. Compass de Credomatic

Compass es un método de pago más sencillo, ya que requiere menos interacción de parte del usuario para poder pagar el parqueo de un centro comercial:

1. El usuario llega al centro comercial con su vehículo y se pone enfrente de la talanquera.
2. La talanquera detecta el chip de Compass y se abre.
3. El usuario permanece en el centro comercial el tiempo que éste desee.
4. Cuando el usuario decida retirarse, éste ingresa a su carro y se dirige a una talanquera de salida.
5. La talanquera detecta el chip de Compass, realiza el cobro y se abre.

C. Spectrum App

Spectrum es un método de pago muy parecido al de Compass, la única diferencia es que en lugar de utilizar un chip NFC este utiliza una cámara para poder detectar el número de placa del vehículo. Esta solución está acompañada por una aplicación móvil en donde el usuario puede revisar sus facturas e ingresar sus tarjetas de crédito. Los pasos que tiene que realizar el usuario para poder ingresar y salir de un parqueo de centro comercial son los siguientes:

1. El usuario llega al centro comercial con su vehículo y se pone enfrente de la talanquera.
2. La talanquera lee el número de placa del vehículo utilizando una cámara de video e inteligencia artificial.
3. Al detectar al usuario la talanquera se abre.
4. El usuario permanece en el centro comercial el tiempo que éste desee.
5. Cuando el usuario decida retirarse, este ingresa a su carro y se dirige a una talanquera de salida.
6. La talanquera vuelve a leer el número de placa para poder realizar el cobro. Después de detectar al usuario y de realizar el cobro esta se abre.

VII. Metodología

Debido a la complejidad del proyecto, la metodología se divide en diferentes secciones en donde el orden representa el flujo de la toma de decisiones y del desarrollo del proyecto.

A. Encuesta previa

En la planificación del proyecto se determinó que existen dos métodos para poder identificar al usuario que se encuentre enfrente de una talanquera: Utilizando código QR o chip NFC integrado en el smartphone. Debido a que este proyecto está enfocado en mejorar la experiencia de usuario se debe de crear una encuesta para poder determinar lo que el usuario prefiere utilizar. Esta encuesta debe de ser respondida por los usuarios finales.

B. Arquitectura del sistema

Para este proyecto se debe de crear una arquitectura que sea de tipo cliente-servidor. En la definición de la arquitectura es importante determinar los componentes a utilizar y la función que tiene cada uno. Para una mejor visualización de la arquitectura es importante crear un diagrama que describa las relaciones de los componentes. La definición de esta arquitectura ayudará a que el desarrollo del proyecto sea más ordenado y eficiente.

C. Metodología ágil

Se debe de seleccionar un tipo de metodología ágil para el desarrollo del proyecto. Para esto se debe de realizar un análisis de las diferentes metodologías ágiles que existen y seleccionar la que más conviene para el proyecto. Al tener la metodología ágil se debe determinar las tareas a realizar para poder desarrollar el proyecto y crear un cronograma de actividades para poder definir las fechas de entregas.

D. Frameworks, herramientas y plataformas a utilizar

En este tipo de proyectos es recomendable utilizar frameworks para el desarrollo, ya que hace que este sea más rápido y ágil. Debido a que la arquitectura a utilizar será una de tipo Cliente-Servidor entonces se debe definir los frameworks que se utilizarán para el frontend y para el backend. Para esto se debe de realizar un análisis de los diferentes tipos de frameworks que se puedan utilizar para el desarrollo de aplicaciones móviles y de backend para poder seleccionar los que mejor se adapten a las necesidades del proyecto. De la misma manera se debe de realizar un análisis de las plataformas en la nube disponibles en donde se pueda establecer el backend de manera pública.

E. Funcionalidades de la aplicación móvil

Independientemente del framework que se utilizará para el desarrollo de la aplicación móvil es importante definir las funcionalidades que tendrá la misma. Para esto se deben de determinar y describir los diferentes módulos que tendrá la aplicación móvil. Estos módulos deben de definirse de una manera que tenga congruencia con la arquitectura del sistema previamente definida.

F. Funcionalidades del backend

De la misma manera con la aplicación móvil, independientemente del framework que se seleccione para desarrollar el backend se deben de determinar las funcionalidades que este tendrá. Para esto es importante determinar y describir las API's que se utilizaran. Estas API's deben de definirse de una manera que tenga congruencia con la arquitectura del sistema previamente definida. Es importante también definir un diagrama de entidad-relación que se utilizará en la base de datos.

G. Proceso de retroalimentación del usuario

Debido a que uno de los objetivos de este proyecto es poder crear una solución que tenga una buena experiencia de usuario, es importante poder recibir una retroalimentación constante de parte de los usuarios finales. Para esto se debe crear un proceso en el que los usuarios puedan poner a prueba los entregables y al mismo tiempo dar su retroalimentación de esta.

H. Propuesta de integración con talanqueras reales

A pesar de que el alcance del prototipo de este proyecto es llevarlo a un ambiente de desarrollo, es importante encontrar la manera para poder llevar esta solución a un ambiente de producción. Por esta razón se necesita una propuesta para poder integrar esta solución con talanqueras reales.

VIII. Resultados

A. Resultados de encuesta previa

La encuesta previa se utilizó por dos razones principales, la primera era para poder encontrar usuarios beta que quisieran probar los prototipos. La segunda razón era para poder determinar si utilizar tecnología NFC o QR para identificar a los usuarios en las talanqueras. En esta encuesta se presentaron las siguientes preguntas y opciones:

1. ¿Cuántos años tienes?
 - a. 0 a 18
 - b. 19 a 30
 - c. 31 a 50
 - d. 51 o más
2. Aproximadamente, ¿cuántas veces has visitado un centro comercial durante la pandemia?
 - a. Ni una vez
 - b. Entre 1 y 5 veces
 - c. Entre 5 y 10 veces
 - d. Más de 10 veces
3. ¿Qué método de pago de parqueo utilizas normalmente?
 - a. Máquina de centro comercial
 - b. Compass de Credomatic
 - c. Spectrum App
 - d. Otro (especifique)
4. ¿Cómo calificarías la experiencia de usuario de tu selección anterior?
 - a. Muy malo
 - b. Malo
 - c. Regular
 - d. Bien
 - e. Excelente
5. Si pudieras implementar una nueva manera para entrar y salir de un parqueo de centro comercial, ¿cuál elegirías?
 - a. Código QR
 - b. Acercando tu teléfono a un sensor
 - c. Ambas
 - d. Otro (especifique)
6. ¿Te interesaría probar un prototipo de una nueva manera para entrar u salir de un parqueo de centro comercial?
 - a. Sí
 - b. No
7. Si tu respuesta anterior fue “Sí”, agrega tu contacto.

Esta encuesta se colocó a la plataforma de MonkeySurvey y el link se le compartió a un total de 58 personas. Los resultados de los encuestados fueron los siguientes:

1. ¿Cuántos años tienes?

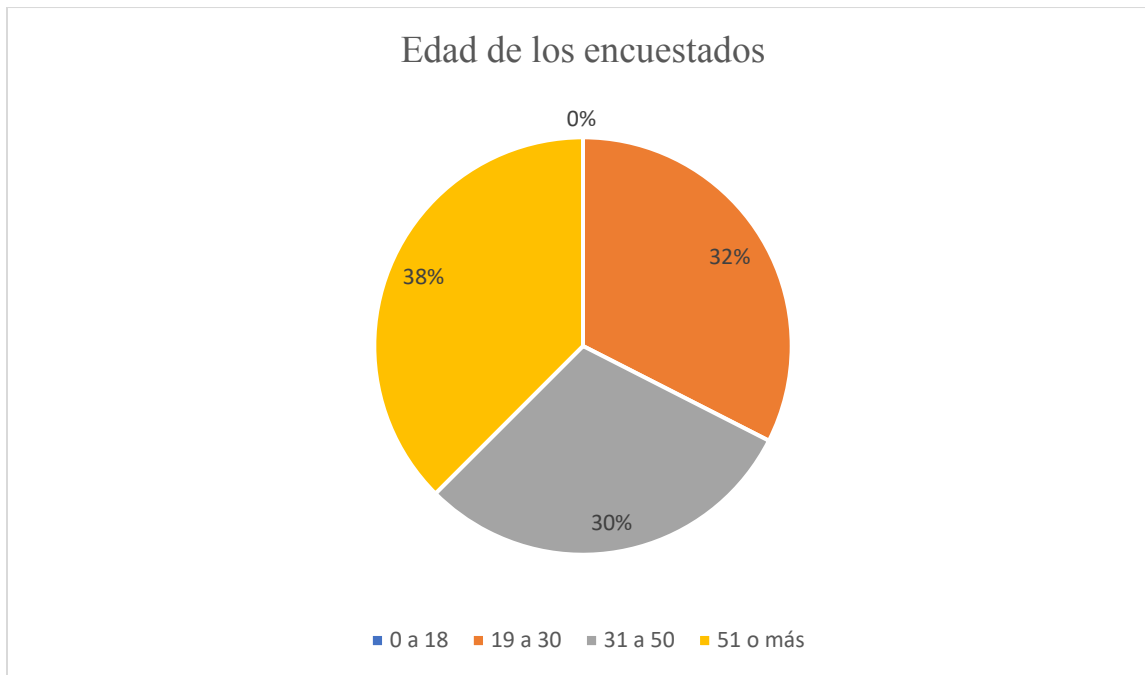


Figura 5. Gráfica de la edad de los encuestados.

2. Aproximadamente, ¿cuántas veces has visitado un centro comercial durante la pandemia?

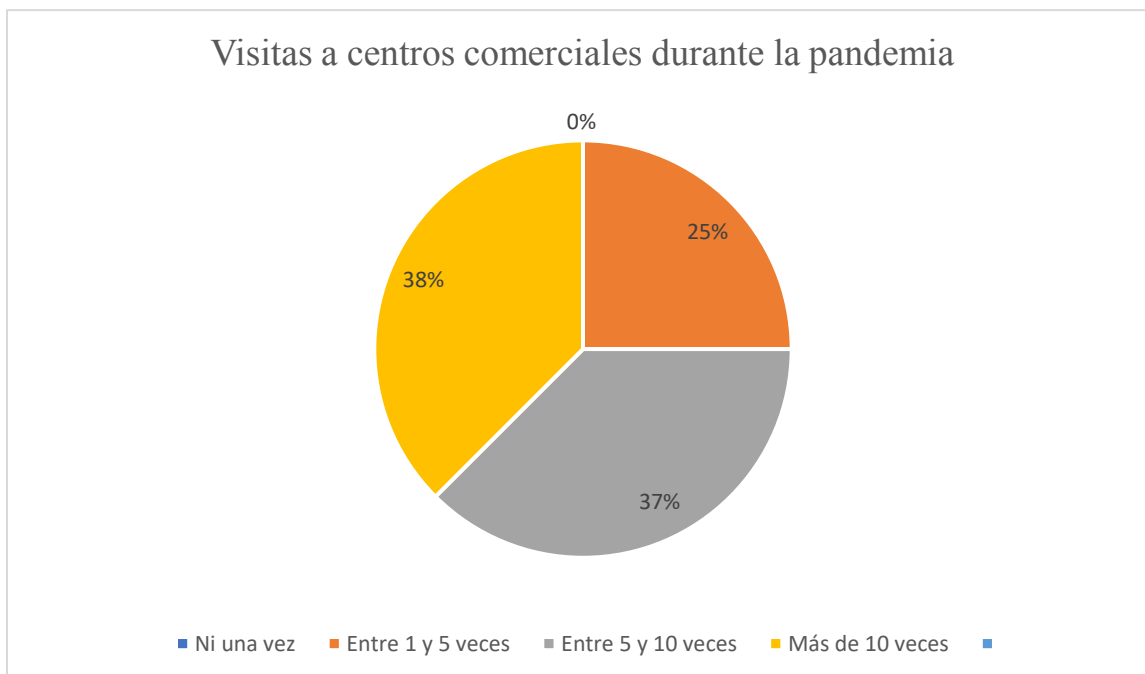


Figura 6. Gráfica de cantidad de visitas a centros comerciales durante la pandemia.

3. ¿Qué método de pago de parqueo utilizas normalmente?

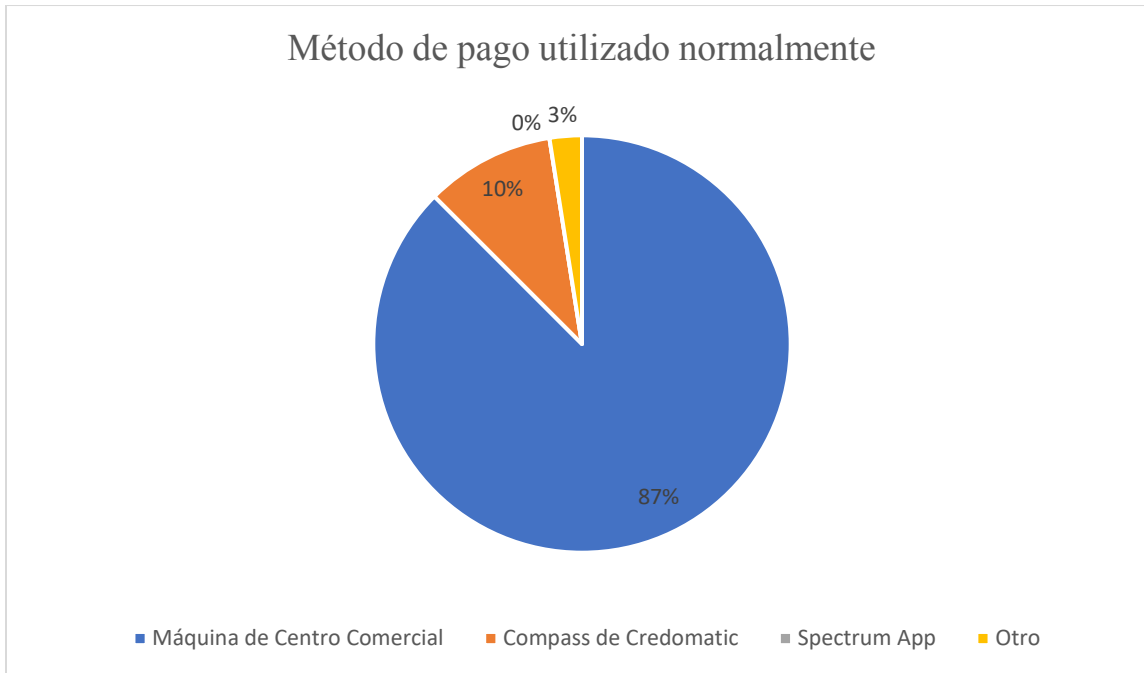


Figura 7. Gráfica de método de pago que utilizan normalmente.

4. ¿Cómo calificarías la experiencia de usuario de tu selección anterior?

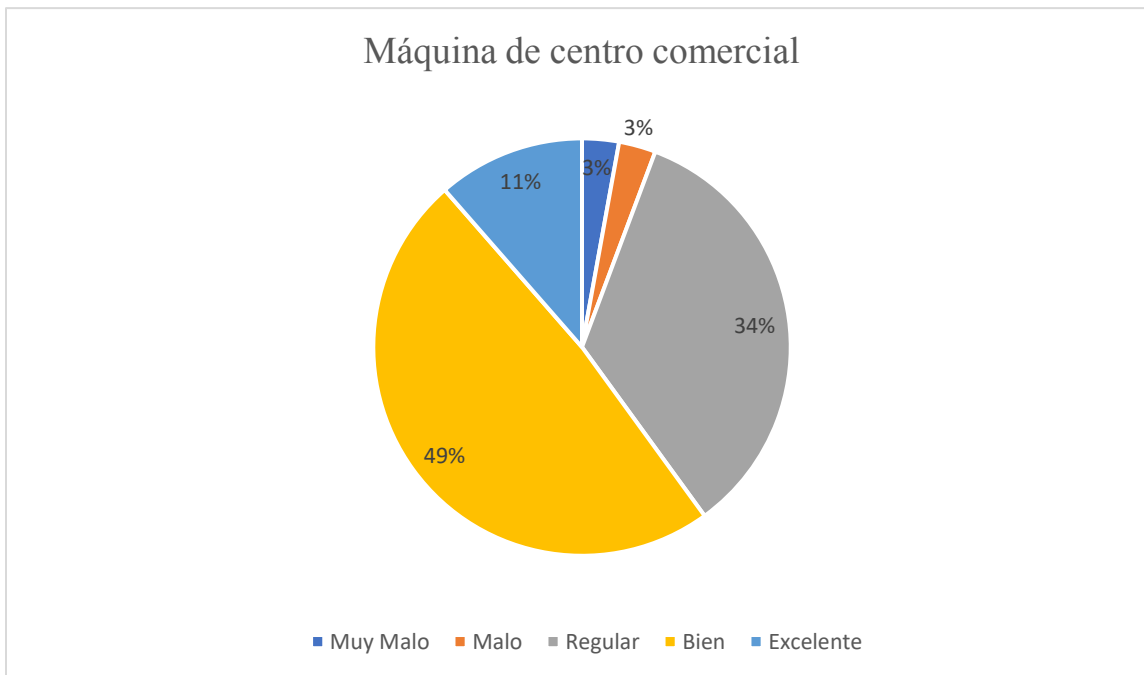


Figura 8. Gráfica de satisfacción de experiencia de usuario con máquina de centro comercial.

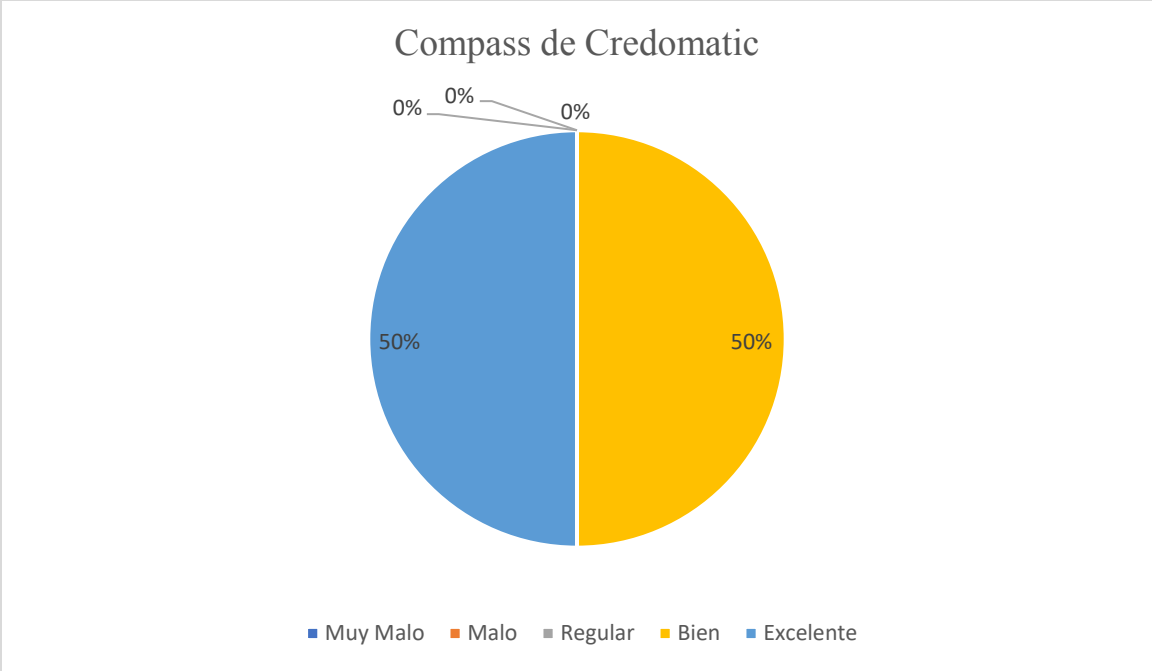


Figura 9. Gráfica de satisfacción de experiencia de usuario con Compass de Credomatic.

5. Si pudieras implementar una nueva manera para entrar y salir de un parqueo de centro comercial, ¿cuál elegirías?

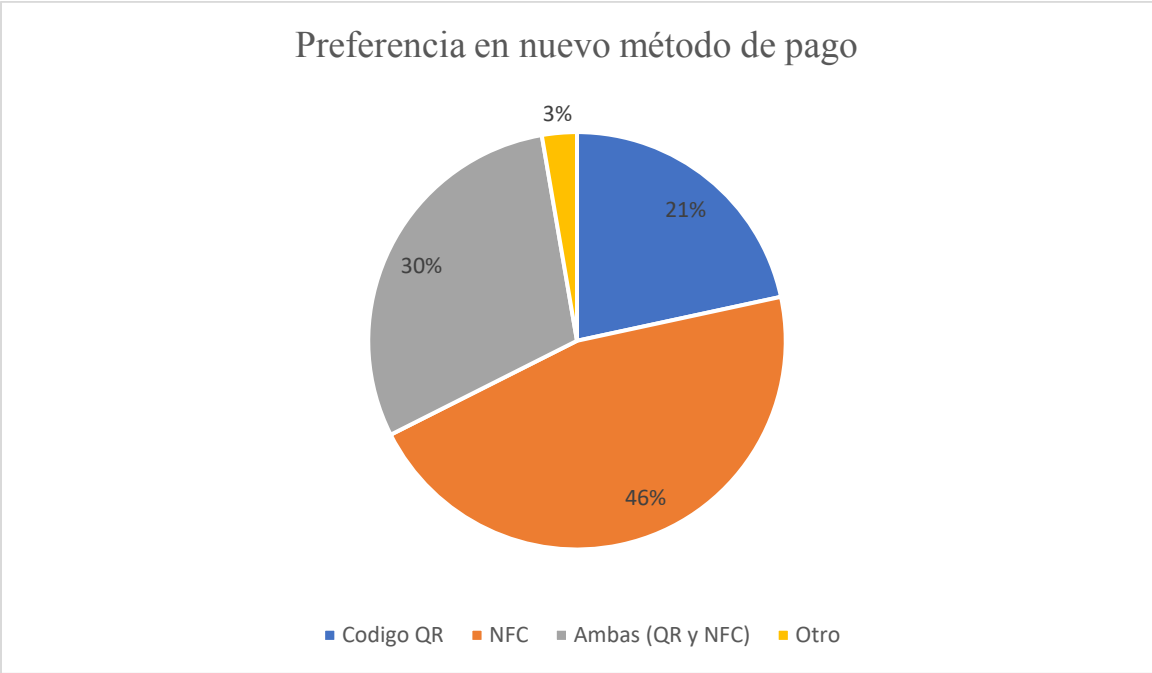


Figura 10. Gráfica de preferencia de tecnología para nuevo método de pago.

B. Resultados del análisis de la arquitectura del sistema

El diagrama de la arquitectura de componentes se construyó con base en las necesidades del proyecto y de las respuestas recibidas en la encuesta previa. El resultado es el siguiente:

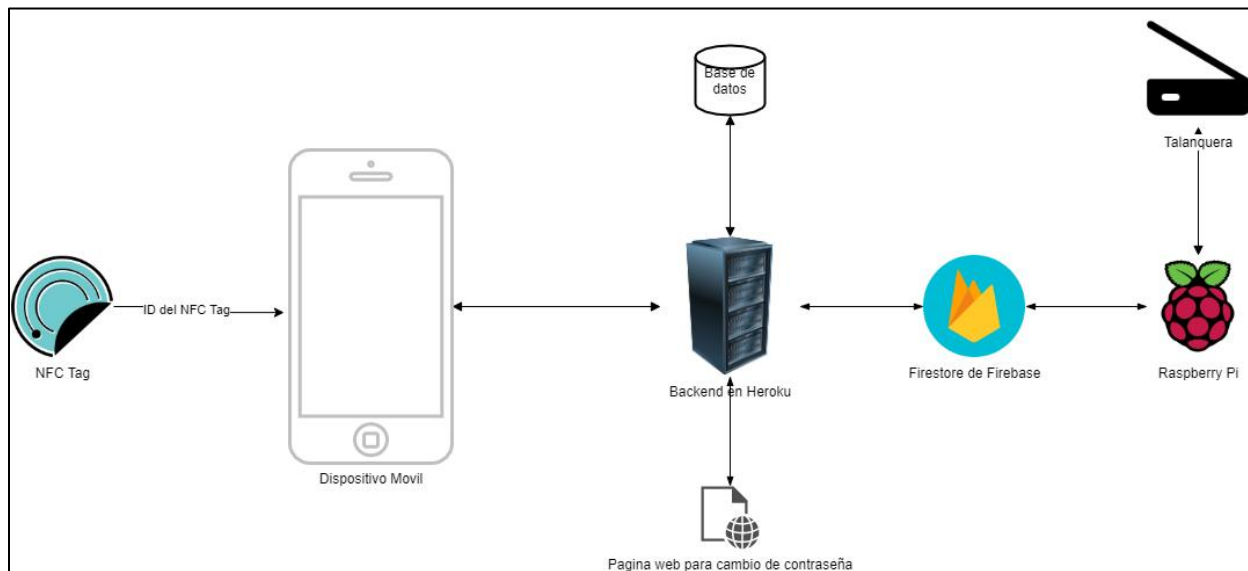


Figura 11. Diagrama de arquitectura del sistema.

Los componentes utilizados en la arquitectura son los siguientes:

- NFC Tag: En la sección de resultados se puede ver que la mayoría de las personas respondieron en la pregunta número 5 de la encuesta previa que prefieren utilizar la tecnología NFC. Es por esta razón que el proyecto se siguió desarrollando utilizando esta tecnología. Cada tag tiene un identificador único que se puede utilizar para identificar en qué talanquera se encuentra el usuario.
- Dispositivo móvil: Se optó por un dispositivo móvil para canal de interacción del usuario debido a que en el año 2021 es un dispositivo al que casi todos pueden tener acceso. Otra de las razones es que la tecnología NFC se está integrando en casi todos los dispositivos móviles que se encuentran en el mercado. En la arquitectura Cliente-Servidor, a éste se le considera como un agente Pasivo debido que solo se encarga de solicitar y presentar datos.
- Servidor de backend: El servidor de backend funciona como un ente que procesa todos los datos, y también como un puente entre el dispositivo móvil y la talanquera. En la arquitectura cliente-servidor, al servidor de backend se le considera como un agente activo, debido a que se encarga de procesar y presentar datos al cliente. Este servidor no solo se conecta con el dispositivo móvil, sino que también con la base de datos y con Firestore de Firebase.

- Base de datos de PostgreSQL: La base de datos se encarga de almacenar datos importantes para el proyecto. El gestor de base de datos que se utilizó para este proyecto es el de PostgreSQL debido a que:
 - Su disponibilidad es multiplataforma. Si en algún momento se desea migrar la base de datos a otra plataforma no deberían existir complicaciones que comprometan la integridad de los datos almacenados.
 - Es de tipo relacional. La complejidad del proyecto exige que se utilice una base de datos de tipo relacional. Funciona con el estándar SQL.
 - Soporta la concurrencia multiversión (MVCC), la cual permite añadir a las transacciones una imagen del estado de la base de datos. De esta manera las transacciones con más consistentes, repercutiendo de forma positiva en el rendimiento del programa.
- Firestore de Firebase: Firestore es una base de datos de tipo no relacional la cual permite tener los datos más actualizados en tiempo real con las plataformas que se encuentran conectadas a él. En este proyecto esta base de datos mantiene el estado de cada talanquera registrada, es decir, si se encuentra abierta o cerrada.
- Raspberry Pi: Este componente se utilizará para simular la apertura y cierre de una talanquera de un centro comercial. El componente que utilizará la Raspberry Pi para simular la apertura y cierre de una talanquera es un servomotor que se moverá cuando el Firestore de Firebase le diga que la talanquera se tiene que abrir.

Un aspecto importante que se tiene que considerar para la implementación es el tipo de servidores que se utilizarán: On Premise o en la Nube. La diferencia principal entre ambos es que los servidores On Premise se encuentran en la infraestructura local de la empresa, y en la Nube los servidores se encuentran en la infraestructura de una empresa tercera a los cuales se pueden acceder por medio del internet. Para poder seleccionar alguna de estas dos opciones se tienen que comparar las ventajas y desventajas de cada una:

	Ventajas	Desventajas
On Premise	<ul style="list-style-type: none"> • El dueño tiene control de hacer lo que quiera con el servidor. • Los datos sensibles pueden permanecer en el sistema y no se traspasan a terceros. • Siempre se tiene acceso dentro de la infraestructura de la empresa. 	<ul style="list-style-type: none"> • El hardware debe de tener mantenimiento constante. • Los dispositivos de hardware deben de ser los adecuados para el software. • Las licencias agregan un mayor costo y deben de adquirirse en plazos largos.
Nube	<ul style="list-style-type: none"> • Las actualizaciones y el mantenimiento los realiza el proveedor y sin costo adicional. 	<ul style="list-style-type: none"> • Se debe contar con una conexión a internet para poder tener acceso a los servidores.

	Ventajas	Desventajas
	<ul style="list-style-type: none"> • Tiene una alta escalabilidad porque se pueden añadir o eliminar funciones con relativa rapidez. • El costo puede ser menor porque únicamente se paga por los recursos que se utilizan. 	<ul style="list-style-type: none"> • El proveedor es que maneja los datos. • Si el proveedor deja de operar entonces no se podrá seguir utilizando el software.

Cuadro #1. Cuadro de comparación entre servidores On Premise y en la Nube.

(Nephos IT, 2022)

Uno de los aspectos mas importantes a considerar es el costo que genera cada una de estas opciones. La empresa SherWeb ofrece una herramienta para poder calcular los costos de servidores On Premise y en la Nube utilizando los mismos componentes de hardware. La comparación se realizó utilizando los siguientes componentes:

- 2 CPUs
- 16 GBs de RAM
- 500 GBs de disco duro
- 48 meses de ciclo para actualización de hardware

Los resultados de esta comparación fueron los siguientes:

	On-premises	Nube
Año 1	\$39,269.18	\$4,667.33
Año 2	\$9,051.49	\$4,667.33
Año 3	\$9,051.49	\$4,667.33
Año 4	\$9,051.49	\$4,667.33
Año 5	\$39,269.18	\$4,667.33
Año 6	\$9,051.49	\$4,667.33
Año 7	\$9,051.49	\$4,667.33
Total:	\$123,795.81	\$32,671.30

Cuadro #2. Comparación de costos de servidores On Premise y en la Nube.

El costo utilizando la Nube es considerablemente bajo comparando con On Premise, por este motivo se decidió utilizar la nube como host de los servidores. La comunicación interna en la nube se conforma de la siguiente manera:

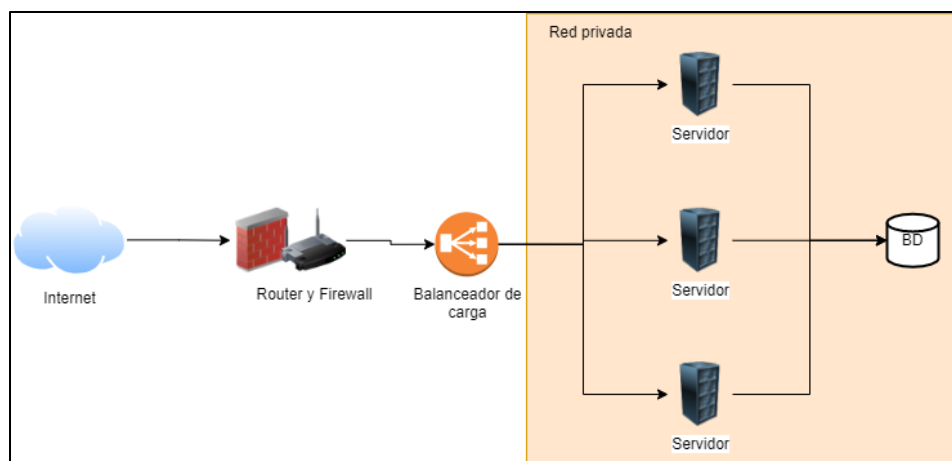


Figura 12. Diagrama de red de un sistema en la nube.

Todas las solicitudes vienen desde el internet que después pasan al firewall y router. Después estas solicitudes pasan por un balanceador de carga el cual decide a que servidor o instancia pasar la solicitud, esto sirve para ser poder manejar varias solicitudes simultáneas. Después de que la solicitud se pasa a alguna instancia esta realiza el proceso que tenga que realizar para poder responder la solicitud. Es importante mencionar que todas las instancias de la red privada tienen acceso a una base de datos centralizada.

C. Resultados del análisis de la metodología ágil a utilizar

Según el análisis realizado, la metodología que se utilizó en el desarrollo del proyecto es la metodología ágil Scrum. Debido a que el equipo de trabajo de este proyecto está conformado por un solo miembro entonces los roles de *Scrum Master*, *Scrum Team Members*, y *Product Owner* se le asignaron a este miembro. El rol de *Stakeholders* puede ser asignado a futuros usuarios o centros comerciales que se encuentren interesados en la solución propuesta.

El tiempo establecido de desarrollo de cada sprint fue de dos semanas, en donde la asignación de tareas se basó en el cronograma de actividades previamente establecido. Al finalizar cada sprint se realizó una reunión de Retrospectiva del sprint con el asesor de este proyecto para verificar los avances. Cada dos sprints transcurridos se realizaron pruebas de experiencia de usuario con algunas de las personas que respondieron la encuesta previa.

La plataforma utilizada para organizar el desarrollo del proyecto fue Jira. En este proyecto se crearon tareas épicas que representan cada módulo del sistema. Cada épica representa el desarrollo del módulo en la aplicación móvil, el servidor backend y en la Raspberry Pi (en donde aplique). Las tareas épicas que se asignaron son las siguientes:

1. Prototipo funcional
2. Entrevistas
3. Diseño de pantallas
4. Registro de cuenta
5. Inicio de sesión

6. Cambio de contraseña
7. Manejo de tarjetas de crédito
8. Lectura NFC y transacciones

El cronograma de actividades se construyó a partir de las épicas previamente descritas, el cual quedó de la siguiente manera:



Figura 13. Cronograma de actividades del proyecto.

D. Resultados del análisis de frameworks, herramientas y plataformas a utilizar utilizadas

Para este proyecto se tuvo que determinar los frameworks a utilizar para el frontend y el backend, y también la plataforma en la nube en la que el backend estaría ejecutándose. Con base en los análisis realizados se utilizaron los siguientes:

Para la aplicación móvil (el frontend) se escogió utilizar el framework de Flutter. Después de un análisis entre este y otros frameworks se determinó que Flutter tiene muchas ventajas que apoyan al desarrollo del proyecto. La primera ventaja que se encontró fue que es un framework de aplicaciones híbridas, esto quiere decir que solamente se escribe un código y este código se puede compilar para las plataformas de Android y iOS. Otra ventaja que se encontró es que a pesar de que sea un framework híbrido el rendimiento de la aplicación en los dispositivos móviles es casi nativo, lo cual favorece al objetivo de tener una buena experiencia de usuario.

Para las funcionalidades que necesitaron de una página web se utilizó el framework de React, el cual se montó en la plataforma de Firebase. Se escogió React porque se basa en Node JS, el cual se utiliza mucho para poder desarrollar aplicaciones web. Una de las ventajas de React es que la comunidad es grande, y gracias a esto existen muchas librerías que se pueden utilizar en el desarrollo. La razón principal por la cual se escogió este framework es por la compatibilidad que se tiene con Firebase para poder montar en la nube la aplicación web.

El framework que se escogió para desarrollar el backend es Django. Una gran ventaja de Django es que es un framework que utiliza el lenguaje de Python, el cual es un buen lenguaje para

poder analizar y procesar datos. Este framework es muy bueno para desarrollar proyectos de pequeña escala debido a la facilidad de crear modelos de bases de datos y API Rest en cuestión de minutos. Otra gran ventaja que tiene Django es la facilidad de integrar librerías de terceros que pueden ser muy útiles para las funcionalidades del proyecto.

Por último, la plataforma en la nube que se escogió para este proyecto fue Heroku. La razón principal de esta selección fue el precio de utilidad, ya que esta plataforma ofrece un plan gratuito para los proyectos de pequeña escala. Heroku ofrece compatibilidad para proyectos desarrollados con Django. Otra de las ventajas de esta plataforma es que ofrece un plan gratuito para utilizar una base de datos de PostgreSQL, la cual tiene un número limitado de registros a escribir, pero para proyectos pequeños funciona bien.

Para el versionamiento del código se utilizó GitHub debido a que es un sistema de versionamiento seguro para repositorios privados. Y para el manejo de ramas se utilizó GitFlow ya que mantiene bien separado los ambientes de trabajo para cada proyecto, y también porque ofrece un flujo congruente para agregar funcionalidades y para realizar correcciones.

E. Resultados del análisis de funcionalidades de la aplicación móvil

Previamente se mencionó que la aplicación móvil se desarrolló con Flutter. Según el análisis realizado, esta aplicación tendrá 6 módulos básicos:

1. **On Boarding (Bienvenida):** Este módulo se encargará de darle una bienvenida al usuario, en donde se describirán las funcionalidades la aplicación y ventajas de estas.
2. **Registro de usuario:** En este módulo el usuario podrá registrarse con sus datos personales y crear una cuenta. Los datos que se le solicitarán al usuario para su registro son: nombre completo, correo electrónico y número de teléfono.
3. **Inicio de sesión:** Con las credenciales previamente creadas en el módulo de Registro el usuario podrá iniciar sesión y utilizar las funcionalidades principales de la aplicación.
4. **Cambio de contraseña:** En este módulo el usuario podrá solicitar un cambio para su contraseña utilizando el correo electrónico con el que se registró. Cuando un usuario solicita un cambio de contraseña se envía un correo electrónico que contiene un link a una página web en donde éste podrá cambiar su contraseña.
5. **Gestión de tarjetas de crédito:** En este módulo el usuario podrá gestionar sus tarjetas de crédito (agregar, visualizar, actualizar y eliminar) utilizando los estándares de Visa.
6. **Lectura de NFC y transacciones:** Este módulo tendrá la funcionalidad de leer los Tags de NFC para poder ingresar y salir de un centro comercial; el pago del parqueo se realizará de manera automática al salir. En este módulo también se podrá visualizar el historial de transacciones del usuario.

La arquitectura de software que se seleccionó para el desarrollo de la aplicación móvil es la de Clean Architecture. Principalmente seleccionó esta arquitectura debido a que ofrece tener escalabilidad en el código y es amigable con los cambios que se tengan que realizar.

F. Resultados del análisis de las funcionalidades del backend

La arquitectura de software utilizada en el backend fue la de Modelo-Vista-Controlador (MVC). Se utilizó esta arquitectura porque es muy útil para desarrollos pequeños y para frameworks como Django.

Para el backend se utilizó el gestor de base de datos de PostgreSQL, el cual utiliza bases de datos de tipo relacional. Por lo tanto, se diseñó un diagrama de entidad-relación para poder determinar los modelos y las relaciones entre ellos. El diagrama utilizado en el proyecto es el siguiente:

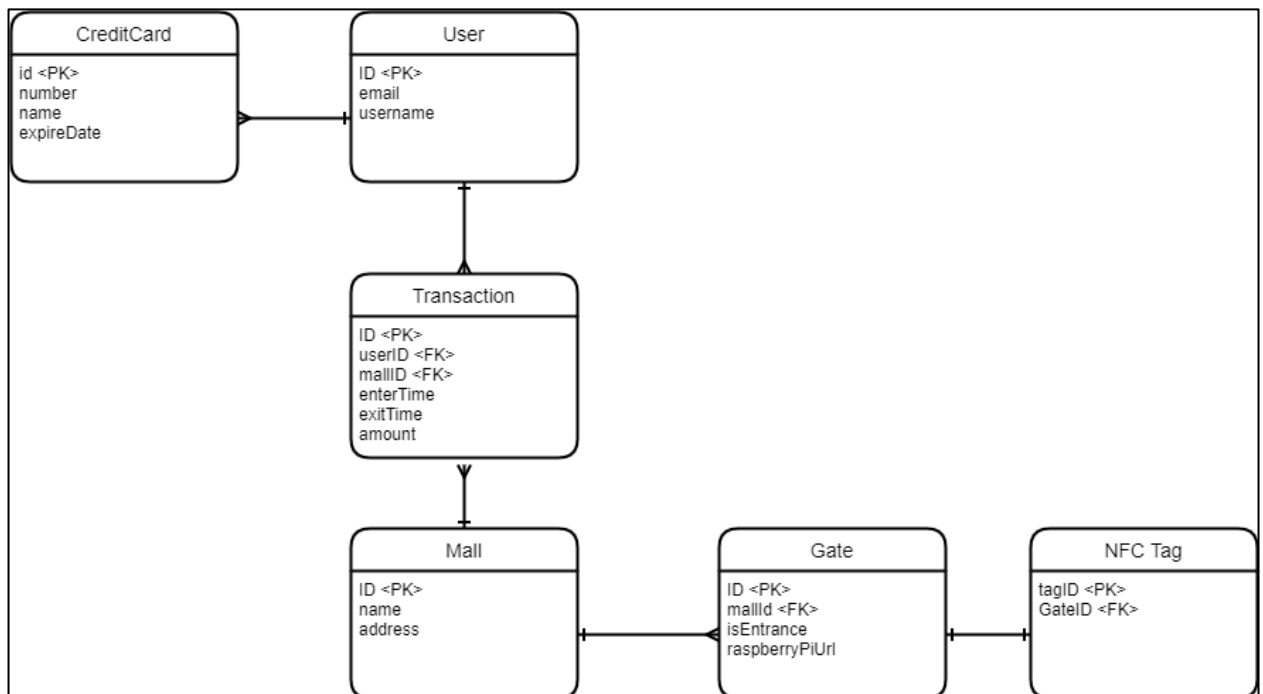


Figura 14. Diagrama de entidad relación de la base de datos.

Previamente se mencionó que el backend se desarrolló con Django, con el cual se crearon las siguientes API's Rest:

- /auth/login: Esta es una solicitud de tipo POST en el cual se inicia sesión.
- /auth/password_reset: Esta es una solicitud de tipo POST en el cual se solicita un correo para poder reiniciar la contraseña de un usuario.
- /auth/password_reset/confirm: Esta es una solicitud de tipo POST en el cual se reinicia la contraseña de un usuario.
- /auth/register: Esta es una solicitud de tipo POST en el cual el usuario puede crear su cuenta en la plataforma.
- /auth/token/refresh: Esta solicitud es de tipo POST en donde el usuario puede solicitar un nuevo token de autenticación para que pueda seguir realizando solicitudes que la necesiten.

- /credit-card: Con esta URL se pueden realizar dos tipos de solicitudes: POST y GET. Con la solicitud GET el usuario puede solicitar la lista de tarjetas de crédito que ha agregado, y con la solicitud POST el usuario puede agregar una tarjeta de crédito a su listado de tarjetas de crédito.
- /credit-card/{id}: Con esta URL se pueden realizar tres tipos de solicitudes: GET, PUT Y DELETE. Con la solicitud GET el usuario puede solicitar la información de una tarjeta de crédito en específico, con la solicitud de tipo PUT el usuario puede actualizar la información de una tarjeta de crédito, y con la solicitud de tipo DELETE el usuario puede eliminar una tarjeta en específico de su listado de tarjetas de crédito.
- /transactions: Esta solicitud es de tipo GET en donde el usuario puede solicitar su listado de transacciones realizadas.
- /transactions/open-mall-gate: Esta solicitud es de tipo POST en donde el usuario puede solicitar abrir una talanquera en específico.

G. Resultados del análisis del proceso de retroalimentación del usuario

Previamente se mencionó que cada dos sprints se realizaron pruebas de experiencia de usuario con algunas de las personas que respondieron la encuesta previa. Las pruebas se realizaron con usuarios que tienen dispositivos Android y iOS debido a que uno de los objetivos de este proyecto es poder tener una aplicación accesible para todos los usuarios. Para hacer esto se tuvo que crear cuentas de desarrollador tanto para la Playstore (Android) como para la App Store (iOS). Con estas cuentas de desarrollador fue posible subir los ejecutables a las tiendas como pruebas beta, así los usuarios pudieron descargarlos sin problema a sus dispositivos.

En cada actualización de la aplicación se incluyeron notas de las nuevas funcionalidades agregadas y los cambios que se realizaron desde la última actualización. Después de que los usuarios realizaran pruebas se les envió una encuesta en donde agregaban su retroalimentación. Estas encuestas incluían las siguientes preguntas:

1. ¿Cuál es tu nombre?
2. ¿Cómo calificarías el diseño de las pantallas de la aplicación?
 - a. Muy mala
 - b. Mala
 - c. Regular
 - d. Buena
 - e. Excelente
3. ¿Qué mejorarías en el diseño de las pantallas?
4. ¿Cómo calificarías la experiencia de usuario de la aplicación?
 - a. Muy mala
 - b. Mala
 - c. Regular
 - d. Buena

- e. Excelente
- 5. ¿Qué mejorarías de la experiencia de usuario de la aplicación?
- 6. ¿Cómo calificarías el funcionamiento de la aplicación?
 - a. Muy mala
 - b. Mala
 - c. Regular
 - d. Buena
 - e. Excelente
- 7. Si algo no funcionó como esperabas, especifica lo que te pasó.

Después de que los usuarios enviaran sus encuestas se analizaron los resultados para poder determinar los nuevos cambios que se tendrían que hacer en el siguiente sprint. De esta manera se pudo trabajar en las tareas asignadas por el cronograma de tareas y también en las correcciones que solicitaban los usuarios beta.

Debido a los resultados de la encuesta previa se definieron tres perfiles que pondrán a prueba la aplicación desarrollada. Los perfiles son los siguientes:

Tipo de perfil	Descripción de perfil
Experta	Una persona experta para el proyecto son los dueños y administradores de los parqueos de centros comerciales. En Guatemala existe la empresa Spectrum que está encargada de los parqueos de centros comerciales como Oakland y Miraflores.
Extremo	Para este perfil se encuentran las personas que van a centros comerciales, pero no muy frecuente, o personas que normalmente no manejan vehículos. Las personas de la tercera edad están consideradas para este perfil.
Común	En este perfil se encuentran las personas que visitan frecuentemente centros comerciales y también que normalmente manejan vehículos. Las personas entre 18 y 50 años se encuentran en este perfil. Este es el perfil que utilizará con mayor frecuencia el producto final.

Cuadro #3. Definición de perfiles que pusieron a prueba el prototipo.

H. Resultados del primer prototipo

En el primer prototipo los usuarios evaluaron los módulos de On Boarding, Registro de usuario, e Inicio de sesión. En esta fase el proyecto se encontraba 30% completado. La cantidad de personas que probaron este prototipo fue de 5 personas. Los resultados del primer prototipo fueron los siguientes:

1. ¿Cómo calificarías el diseño de las pantallas de la aplicación?

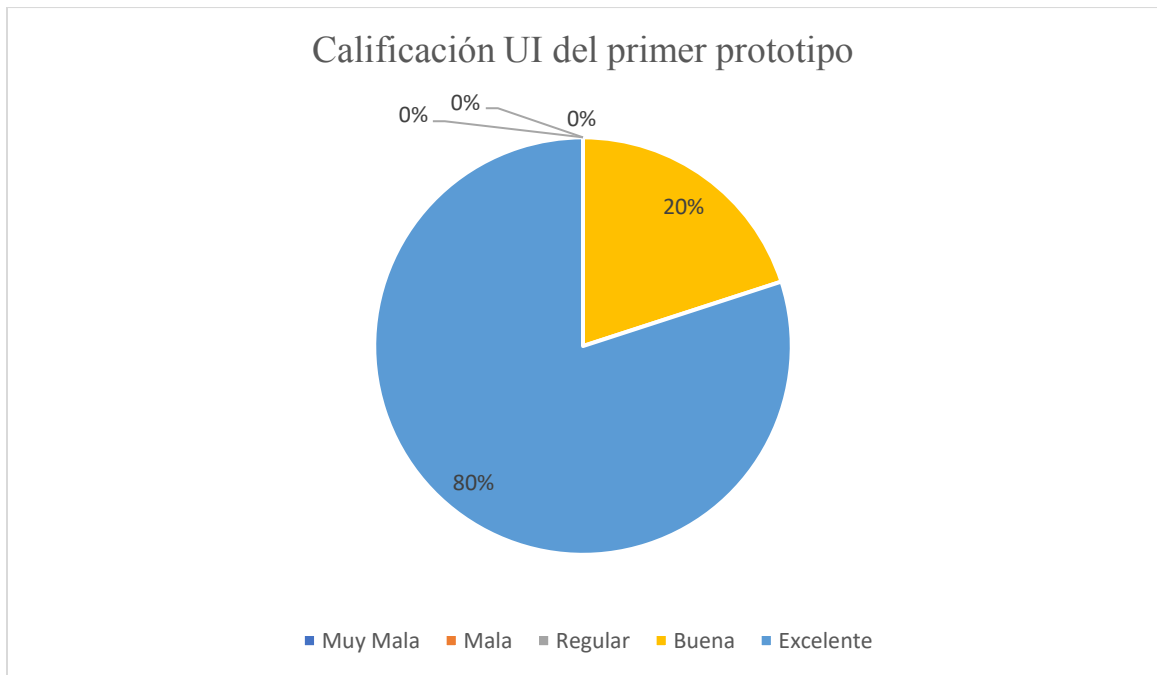


Figura 15. Gráfica de calificación del User Interface del primer prototipo.

2. ¿Cómo calificarías la experiencia de usuario de la aplicación?

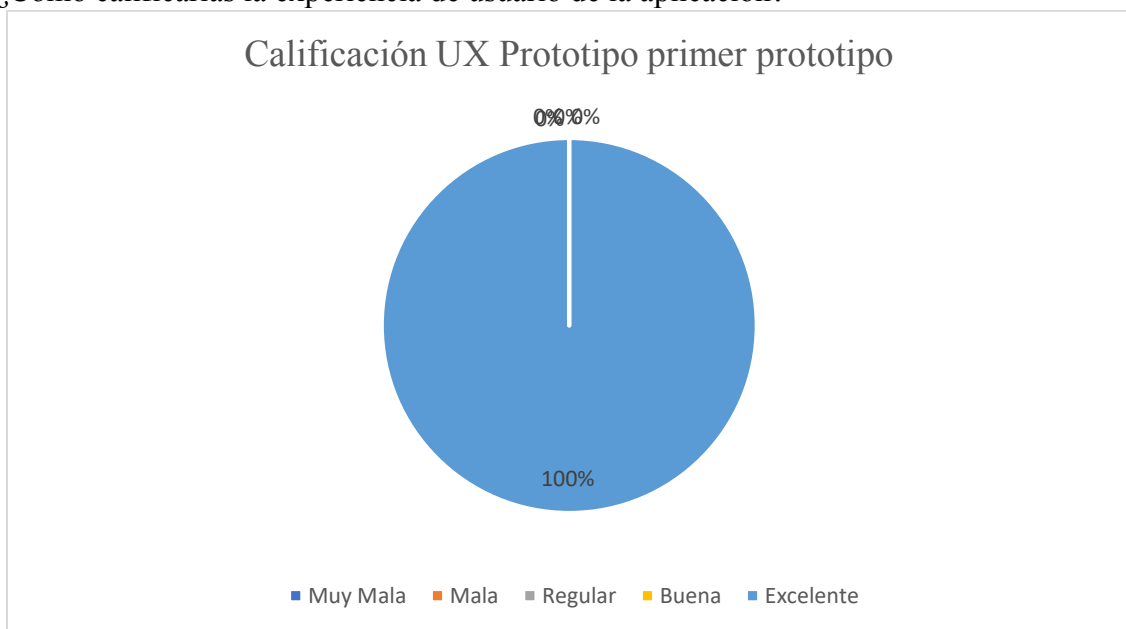


Figura 16. Gráfica de calificación del User Experience del primer prototipo.

3. ¿Cómo calificarías el funcionamiento de la aplicación?

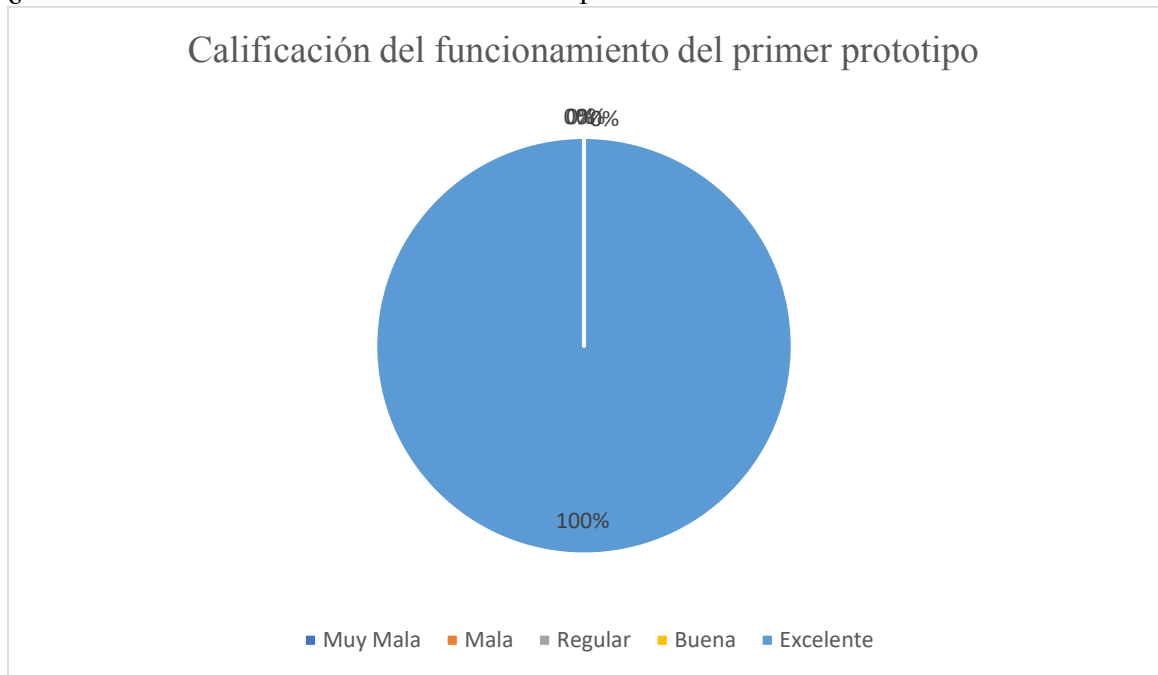


Figura 17. Gráfica de calificación del funcionamiento del prototipo 1.

Algunos de los comentarios de retroalimentación que se recibieron en este prototipo fueron:

- *“Algunos dibujos se ven pixeleados (bien poco)”*
- *“En la pantalla donde se ingresan todos los datos que se ponga en mayúsculas automáticamente al poner el nombre.”*

I. Resultados del segundo prototipo

En el segundo prototipo los usuarios evaluaron los módulos de Cambio de contraseña y Manejo de tarjetas de crédito. En esta fase el proyecto se encontraba 60% completado. La cantidad de personas que probaron este prototipo fue de 5 personas. Los resultados del segundo prototipo fueron los siguientes:

1. ¿Cómo calificarías el diseño de las pantallas de la aplicación?

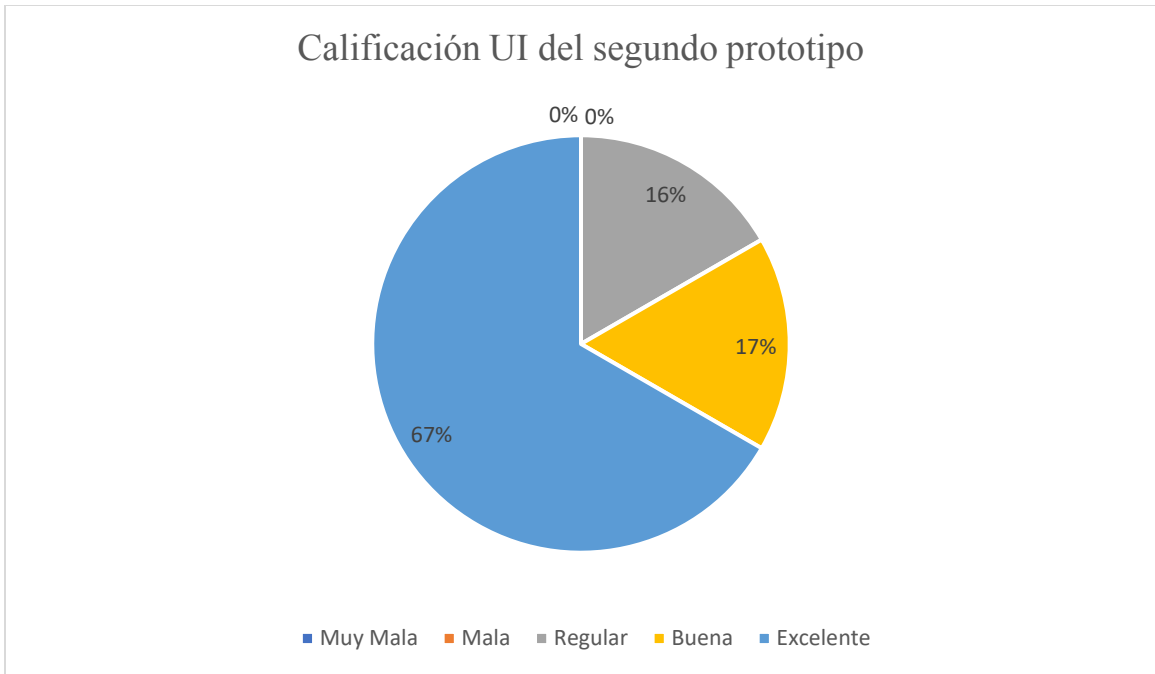


Figura 18. Gráfica de calificación del User Interface del segundo prototipo.

2. ¿Cómo calificarías la experiencia de usuario de la aplicación?

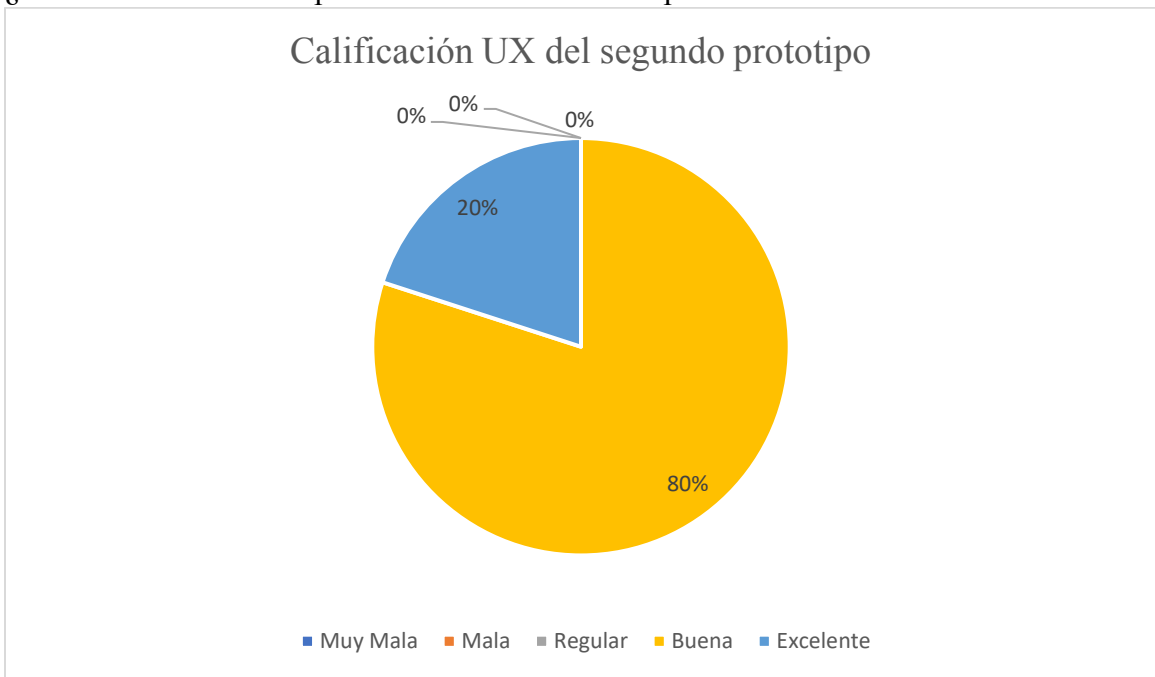


Figura 19. Gráfica de calificación del User Experience del segundo prototipo.

3. ¿Cómo calificarías el funcionamiento de la aplicación?

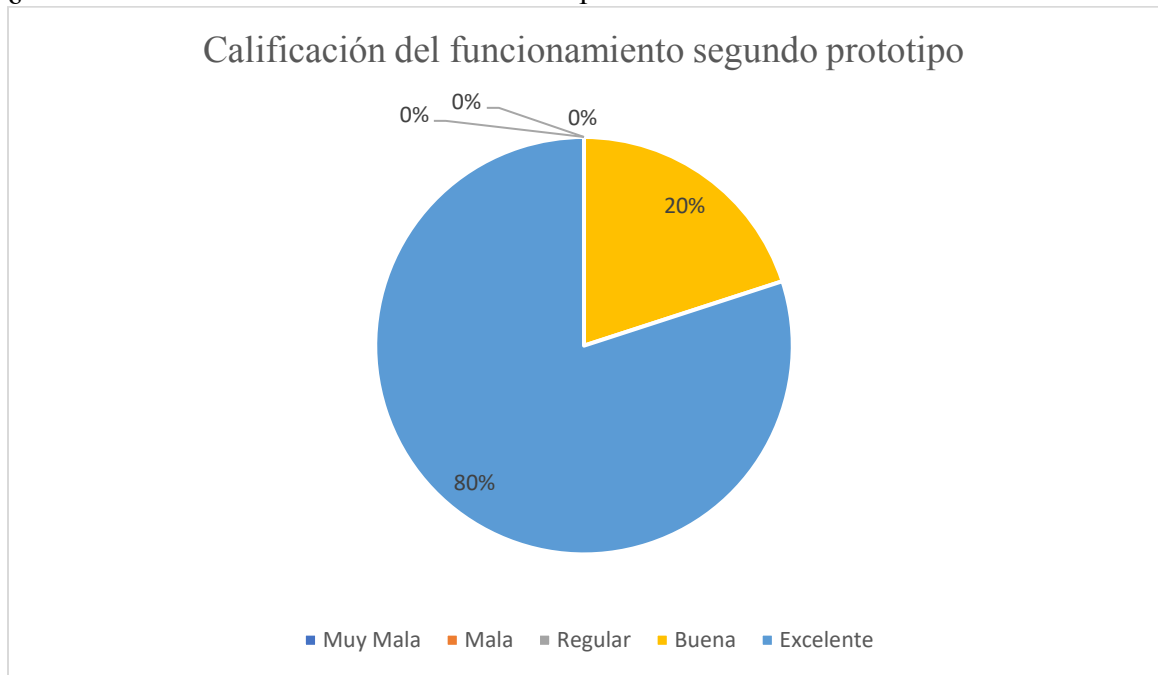


Figura 20. Gráfica de calificación del funcionamiento del segundo prototipo.

Algunos de los comentarios de retroalimentación que se recibieron en este prototipo fueron:

- *“En la parte de restablecer contraseña que tenga como debe ser la contraseña (tamaño y que tan compleja) y también en la parte de la tarjeta poner un botón de información para la fecha. Y falta el número de seguridad de atrás de la tarjeta.”*
- *“Tal vez solo cuando entro a la pantalla de home me gustaría que me salieran como flechas hacia los botones indicando que hace cada uno como paso a paso.”*
- *“Agregarle estilo a la página web de recuperar contraseña.”*
- *“Modificar el diseño del “Tap” de pago.”*

J. Resultados del tercer prototipo

En el tercer prototipo los usuarios evaluaron los módulos de Transacciones y Pago con NFC. En esta fase el proyecto se encontraba 100% completado. La cantidad de personas que probaron este prototipo fue de 5 personas. Los resultados del segundo prototipo fueron los siguientes:

1. ¿Cómo calificarías el diseño de las pantallas de la aplicación?

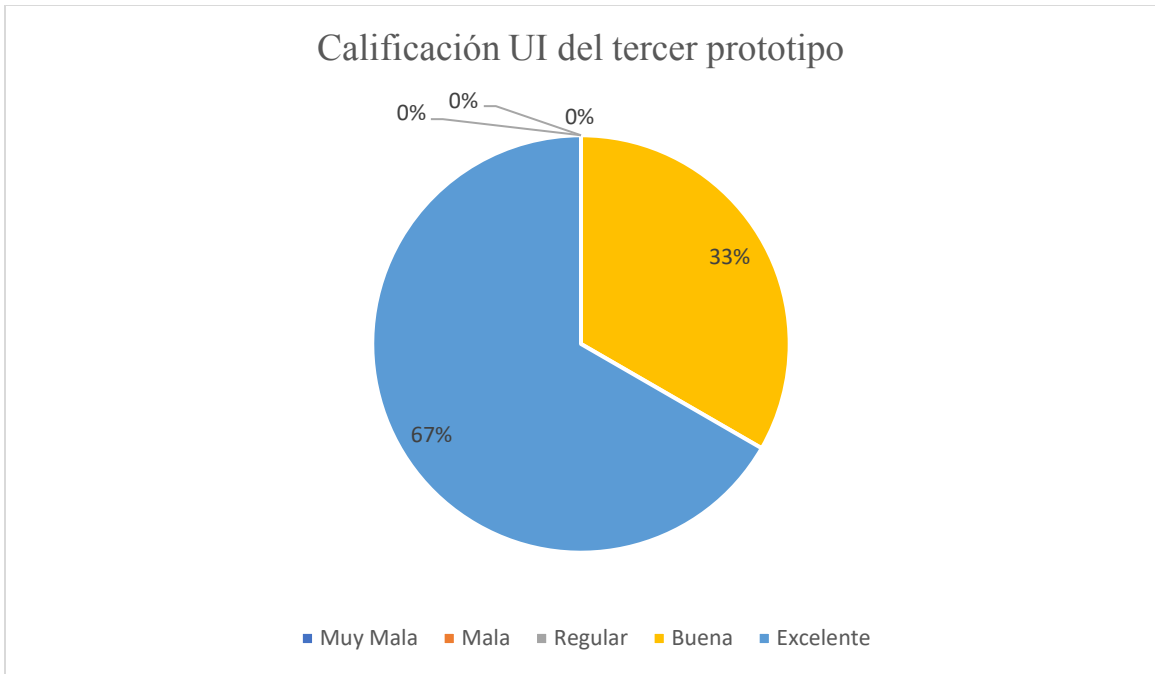


Figura 21. Gráfica de calificación del User Interface del tercer prototipo.

2. ¿Cómo calificarías la experiencia de usuario de la aplicación?

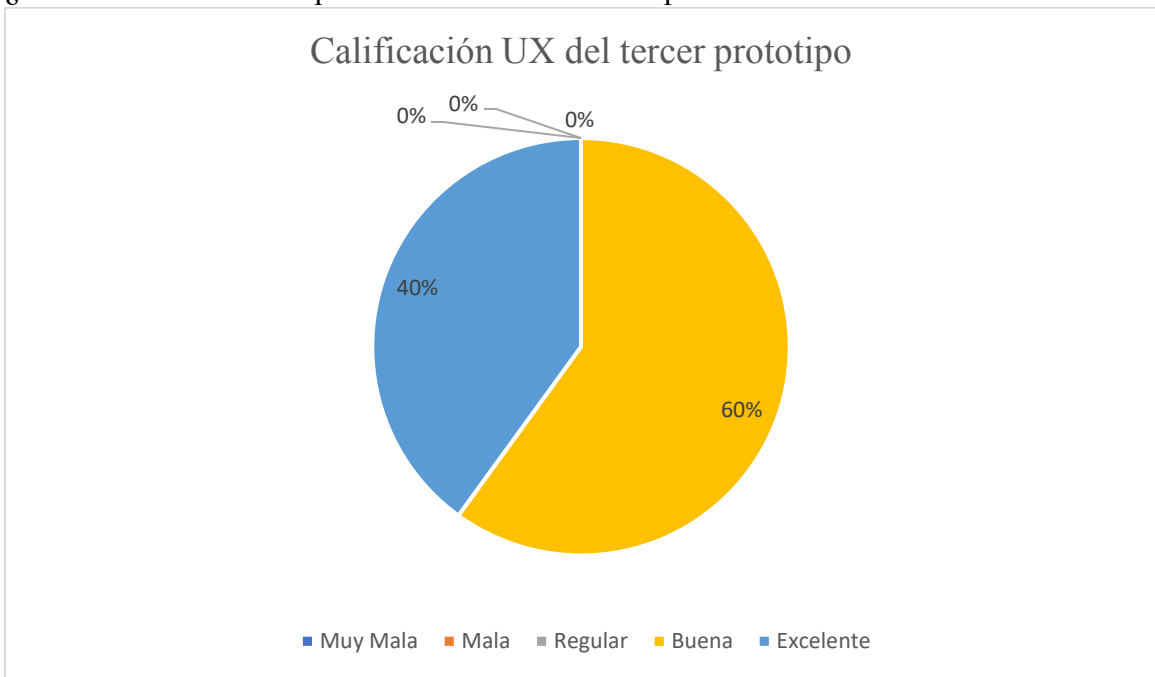


Figura 22. Gráfica de calificación del User Experience del tercer prototipo.

3. ¿Cómo calificarías el funcionamiento de la aplicación?

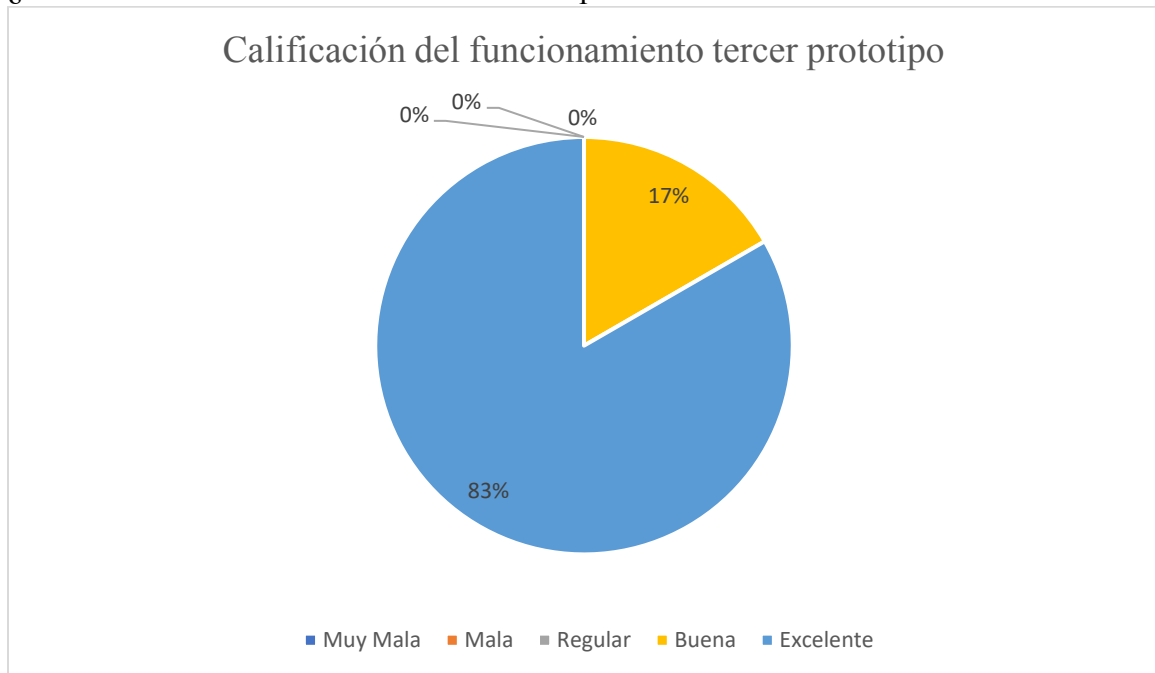


Figura 23. Gráfica de calificación del funcionamiento del tercer prototipo.

Algunos de los comentarios de retroalimentación que se recibieron en este prototipo fueron:

- *“Sería bueno agregarle una forma de autenticar al empezar la lectura NFC.”*
- *“No se puede ver el detalle de la transacción.”*
- *“Me gusto el tutorial al iniciar sesión.”*
- *“Agregarle estilo a la página web de recuperar contraseña.”*

K. Propuesta de integración con talanqueras reales

Previamente se mencionó que se necesita una propuesta para poder llevar este proyecto a un ambiente de producción. Después de analizar lo que conllevaría llevar este proyecto al mundo real, se identificó el siguiente cambio:

- En lugar de que la Raspberry Pi controle un Servomotor, ésta se encargará de controlar la talanquera real.

La talanquera que se utilizaría en esta propuesta es la Amano AGP-1711, este modelo es el que se utiliza en el centro comercial de Paseo Cayalá. Esta talanquera cuenta con una placa de entradas y salidas que puede ser utilizada para controlar la talanquera con dispositivos externos, en este caso la Raspberry Pi. En la placa de entradas y salidas existe una sección de *Control Inputs* como se puede ver en la Figura 4. En esta sección existe un input que se llama *Trans Open*, la cual al recibir una señal la talanquera se abre según su configuración.

Por lo tanto, para poder integrar el proyecto con una talanquera real se necesita conectar la Raspberry Pi con la entrada de *Trans Open* de la talanquera. Así cuando se necesite abrir una talanquera todo lo que hay que hacer es mandar una señal desde la Raspberry Pi hacia la talanquera Amano. El resto de la arquitectura se mantendría igual ya que no es necesario realizar cambio para poder integrarla con una talanquera real.

L. Costos

El prototipo desarrollado en este proyecto es específicamente para un ambiente de desarrollo, el cual tuvo un costo. Sin embargo, como también se presentó una propuesta para llevar esta solución a un ambiente de producción, es importante poder detallar los costos que conllevan desarrollar esta solución tanto en un ambiente de desarrollo como también para un ambiente de producción.

Ambiente de desarrollo		
Nombre	Cantidad	Costo (GTQ)
MacBook Air para desarrollo de aplicaciones híbridas (la más barata actualmente)	1	9,999.00
Dispositivo Android (para pruebas)	1	949.00
Dispositivo iOS (para pruebas)	1	6,999.00
Raspberry Pi	1	390.00
Stickers NFC	1 paquete	78.00
Servomotor	2	78.00
Cuenta de desarrollador Android	1	195.00
Cuenta de desarrollador de iOS	1	780.00
Hosting en Heroku	1	0.00
Base de datos en Heroku	1	0.00
Total		Q. 19,468.00

Cuadro #4. Inversión inicial para ambiente de desarrollo (algunos precios pueden variar dependiendo de los modelos que se compren de algunos dispositivos físicos).

El cuadro de inversión inicial del ambiente de desarrollo se calculó bajo los siguientes supuestos:

- El equipo de desarrollo no cuenta con equipo (computadora) para poder desarrollar.
- El equipo de desarrollo no cuenta con dispositivos móviles para desarrollar, tanto para Android como para iOS.
- La cuenta de desarrollador de iOS solo se utilizará un año, en caso se utilice por más de un año hay que agregarle Q. 780.00 por año extra.

Ambiente de producción		
Nombre	Cantidad	Costo (GTQ)
MacBook Air para desarrollo de aplicaciones híbridas (la más barata actualmente)	1	9,999.00
Dispositivo Android (para pruebas)	1	949.00
Dispositivo iOS (para pruebas)	1	6,999.00
Raspberry Pi	2	780.00
Stickers NFC	1 paquete	78.00
Cuenta de desarrollador Android	1	195.00
Cuenta de desarrollador de iOS	1	780.00
Hosting en la nube	1	3,033.00
Talanquera Amano AGP-1711	2	7,800.00
Total		Q. 30,613.00

Cuadro #5. Inversión inicial para ambiente de producción (algunos precios pueden variar dependiendo de los modelos que se compren de algunos dispositivos físicos).

El cuadro de inversión inicial del ambiente de producción se calculó bajo los siguientes supuestos:

- El equipo de desarrollo no cuenta con equipo (computadora) para poder desarrollar.
- El equipo de desarrollo no cuenta con dispositivos móviles para desarrollar, tanto para Android como para iOS.
- El centro comercial al que se le está instalando el sistema no cuenta con talanqueras en las entradas y salidas de su parqueo. En caso el centro comercial si cuenta con talanqueras entonces el costo de éstas se puede descartar.

En los cuadros de inversión inicial para cada ambiente existen ciertos servicios que se tiene que pagar cada cierto tiempo, ya sea mensual o anual. Estos servicios son:

- Cuenta de desarrollador de iOS: Las cuentas de desarrollador para iOS son cuentas que se tiene que pagar anualmente, y el costo anual es de Q. 780.00.
- Hosting en la nube: Por costos se decidió utilizar servidores en la nube, ya que a través de los años reduce el costo a comparación de los servidores On Premise. Utilizando la herramienta de SherWeb el cálculo del costo mensual es de Q. 3,033.732

IX. Análisis de resultados

A. Análisis de resultados de la encuesta previa

Los resultados de la encuesta previa ayudaron a aclarar el contexto en donde se encuentra la problemática del proyecto. A pesar de la pandemia del Covid-19, muchos usuarios siguen visitando los centros comerciales. Si no se toman las medidas adecuadas dentro del centro comercial entonces existe un gran riesgo de contagio del virus.

La mayoría de las personas encuestadas utilizan el método de pago de tarjeta de papel y máquina, lo cual aumenta el riesgo de contagio. Con este resultado se puede afirmar que a pesar de que el Compass de Credomatic es una buena solución para el pago de parqueos de centros comerciales, no es accesible para todos.

La quinta pregunta presenta diferentes opciones de método de pago que se pudieran implementar en una nueva solución. En primer lugar, quedó implementar una solución que utilice NFC y códigos QR. En segundo lugar, quedó utilizar únicamente NFC. En tercer lugar, quedó utilizar solamente códigos QR. Debido los tiempos limitados de desarrollo, se seleccionó implementar el segundo lugar, que sería NFC. Es posible que los encuestados seleccionaran esta opción debido a la facilidad que tendría utilizar NFC, ya que lo ven como *“solo acercar el teléfono a un sensor”*.

B. Análisis de resultados del primer prototipo

En el primer prototipo se obtuvieron muy buenos resultados. La satisfacción de los tres aspectos evaluados fue muy alta. Esto se debe a que en el primer prototipo no se evaluaron módulos que le agregan valor a la aplicación, en esta fase el prototipo solo era una app que registraba usuarios e iniciaba una sesión. A pesar de que estos son módulos importantes para una aplicación, no la hicieron única.

El motivo por el cual no se logró desarrollar más funcionalidades es porque en esta fase se realizaron muchas actividades que el usuario no puede ver por medio de la aplicación. Entre estas actividades se encuentran:

- Crear los proyectos de la aplicación en Flutter y del backend en Django.
- Establecer la arquitectura de cada uno de los proyectos.
- Subir a Heroku la base de datos y el proyecto de Django.
- Crear las cuentas de desarrollador para Android y para iOS.
- Mandar a revisión la aplicación de Android y de iOS en sus respectivas tiendas.

C. Análisis de resultados del segundo prototipo

En el segundo prototipo se obtuvo más retroalimentación para mejorar lo que se desarrolló. En aspectos generales del funcionamiento los usuarios obtuvieron una buena respuesta de parte de la aplicación.

En la experiencia de usuario recomendaron aspectos importantes para mejorar la experiencia de usuario. En esta sección hubo dos comentarios que se relacionaron mucho respecto a la pantalla del Home. Un usuario recomendó mejorar el diseño del “Tap” debido a que no es muy intuitivo, pero otro usuario recomendó mostrar un tutorial interactivo de esta pantalla la primera vez que se ingrese. El segundo comentario del tutorial interactivo puede satisfacer la necesidad que solicita el primer usuario.

Otro aspecto importante que mencionaron entre los usuarios es que hay que mejorar la página web de recuperar contraseña. Los usuarios comentan que en funcionamiento está bien, pero que le falta mucho en el diseño y experiencia de usuario. Uno de los usuarios comentó que sería bueno colocar puntos que indiquen lo que debería contener una contraseña. Otro usuario comentó que le generó desconfianza la página web debido a que no tiene un diseño profesional, sino que se encuentra muy simple. Para el siguiente prototipo se podrían juntar ambos comentarios para poder generar una mejor experiencia de usuario.

D. Análisis de resultados del tercer prototipo

La retroalimentación del tercer y último prototipo fue similar a la retroalimentación del segundo prototipo. Se recibieron aspectos a mejorar y también nuevas funcionalidades que le pueden agregar valor al producto final.

Una de las recomendaciones que se recibieron es agregar una manera de autenticar al usuario cuando se inicie la lectura NFC. Esto puede agregar un poco de tiempo en el proceso de abrir una talanquera, lo cual puede generar disconformidad con otros usuarios. La mejor ruta para tomar con esta recomendación es agregar una opción en el menú de ajustes en donde el usuario pueda activar la autenticación cuando se inicie una lectura NFC. Para evitar que este proceso de autenticación sea largo es mejor aprovechar las lecturas de biométricos que el dispositivo ya tenga, como lectura de huella o de rostro.

En general la respuesta de los usuarios fue muy positiva con la experiencia de usuario y de la interfaz de usuario. Agregar un pequeño tutorial en la pantalla de Home ayudó a que la interfaz de usuario se entendiera mejor. Por esa misma razón no se recibió retroalimentación relacionado con la interfaz de usuario de esta pantalla. La hipótesis que se planteó en el análisis del segundo prototipo se cumplió.

X. Conclusiones

1. La implementación de la tecnología NFC ayudó a cumplir con el objetivo de “*crear un método de pago seguro*” sanitariamente debido a que el usuario no toca máquinas ajenas, solamente su dispositivo móvil.
2. El framework de Flutter ayuda a que la mayoría de las personas puedan tener acceso a esta solución debido a que puede compilar la aplicación móvil para los sistemas operativos Android y iOS.
3. En la actualidad, la mayoría de los fabricantes de dispositivos móviles están fabricando dispositivos que tienen el chip NFC integrado, esto ayuda a que el producto final de este proyecto pueda ser accesible a la mayor cantidad de personas posible.
4. La tecnología de Flutter ayudó a que la aplicación desarrollada tuviera un rendimiento casi nativo. Este aspecto aportó de una buena manera a la experiencia de usuario de la aplicación porque los usuarios sintieron la aplicación muy fluida a comparación de otras aplicaciones que utilizan otros frameworks.
5. La pequeña cantidad de módulos dentro de la aplicación móvil ayudó a que los usuarios no la sintieran compleja o difícil de usar. Esto ayuda a la experiencia de usuario porque la aplicación móvil se vuelve a una aplicación de fácil uso.
6. El uso de la Raspberry Pi ayudó a la implementación de la simulación una talanquera debido a la facilidad de programar el funcionamiento de componentes como los servomotores y también por su compatibilidad con el SDK de Firebase.
7. Agregar otro método de pago dentro de la aplicación móvil utilizando tecnología QR puede ayudar a mejorar a la experiencia de usuario debido a que el usuario puede elegir el método con el que se sienta más cómodo.
8. Las iteraciones de pruebas que se realizaron con los usuarios ayudaron a que se optimizara el funcionamiento, la experiencia de usuario, y la interfaz de usuario de la aplicación desarrollada.
9. La retroalimentación recibida en forma de encuestas ayudó a ordenar y organizar los aspectos a mejorar en cada iteración.
10. La propuesta para poder integrar la solución con una talanquera real le agrega valor a la accesibilidad del proyecto para los posibles compradores del proyecto, ya que no es necesario que estos tengan que comprar nuevas talanqueras para que puedan integrar esta solución.
11. Si el cliente que desee integrar este proyecto ya cuenta con una infraestructura On Premise entonces el costo de mantenimiento del proyecto sería menor al proyectado.

XI. Recomendaciones

Para poder darle un seguimiento a este proyecto y agregarle más valor al producto desarrollado se deben de seguir las siguientes recomendaciones:

1. Se recomienda integrar los pagos realizados en la aplicación con el SDK de Visa. Esta es una empresa que es reconocida internacionalmente, lo cual es un factor que generará mas confianza por parte del usuario al utilizar la aplicación. El proceso de certificación puede llegar a ser largo, pero es un aspecto que le agregará valor al producto.
2. Uno de los objetivos de este proyecto es poder hacer accesible la aplicación para todos los usuarios, y en el desarrollo se logró subir la aplicación a las tiendas de Playstore y App Store. Debido a los problemas que tuvo la empresa de Huawei con Google surgió una nueva tienda que se llama AppGallery, por lo cual se recomienda subir la aplicación a esta tienda para que pueda ser accesible en las tiendas más utilizadas mundialmente. Sobre las tiendas se recomienda:
 - a. Investigar sobre todas las funcionalidades que tienen las plataformas de las tiendas para hacer al realizar pruebas Alpha y Beta.
 - b. Subir las aplicaciones a sus tiendas con un tiempo prudencial porque el proceso que realizan las tiendas de verificación y aprobación es un poco complicado y tardado.
3. Si se desea mover este proyecto a un ambiente de producción, se recomienda cambiar el hosting del backend a una plataforma en la nube que soporte una carga mas pesada de solicitudes simultaneas.
4. Si se desea optimizar más el funcionamiento, experiencia de usuario, e interfaz de usuario de la aplicación se recomienda seguir haciendo iteraciones de prueba con usuarios para poder recibir más retroalimentación de la misma.
5. Se recomienda integrar la tecnología QR como método de pago adicional en la aplicación para que el usuario pueda utilizar el método con el que se sienta más cómodo, y así mejorar la experiencia de usuario del producto.
6. Se recomienda trabajar con una empresa que administre parqueos de centros comerciales para poder integrar el uso de talanqueras reales; de esta manera se puede dejar de utilizar la Raspberry Pi como simulación de una talanquera.
7. Para agregarle una capa de seguridad y asegurar más el intercambio de datos entre la aplicación móvil y el backend, se recomienda encriptar el *body* de cada solicitud HTTP.
8. Se recomienda verificar la infraestructura del cliente que desee integrar este proyecto para poder minimizar los costos de mantenimiento.
9. Se recomienda leer el manual de uso de la talanquera a utilizar en el ambiente de producción para poder verificar que la placa de entradas y salidas sea igual o similar a la de la talanquera Amano AGP-1711. Esto ayudará a que el flujo de integración en el ambiente de producción sea más fluido.

XII. Bibliografía

- Abellán, E. 2020. “*Scrum: qué es y cómo funciona esta metodología*”. Referencia de: <https://www.wearemarketing.com/es/blog/metodologia-scrum-que-es-y-como-funciona.html> [extraído el 30 de septiembre de 2021]
- Alonso, R. 2021. “*¿Qué es el internet de las cosas y por qué se llama así?*”. Referencia de: <https://hardzone.es/reportajes/que-es/internet-cosas-iot/> [extraído el 30 de septiembre de 2021]
- Amano. 2021. “*AGP-1700 Series Parking Gate. Installation and Operation Manual*”. Referencia de: http://www.saffordkennedy.com/Portfolio/enter/amano_port/M20045-10.pdf [extraído el 5 de enero de 2022]
- App Store, Apple. 2021. “*Spectrum App*”. Referencia de: <https://apps.apple.com/gt/app/spectrum-app/id1474578786#see-all/reviews> [extraído el 22 de febrero de 2021]
- Atlassian. 2021. “*Flujo de trabajo de Gitflow*”. Referencia de: <https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow> [extraído el 30 de septiembre de 2021]
- Biblioguías Cepal. 2021. “*Qué son los Códigos QR*”. Referencia de: <https://biblioguias.cepal.org/QR> [extraído el 30 de septiembre de 2021]
- Borges, S. 2019. “*Servidor PostgreSQL*”. Referencia de: <https://blog.infranetworking.com/servidor-postgresql/> [extraído el 30 de septiembre de 2021]
- Camino, P. 2018. “*Qué es Django y por qué usarlo*” Referencia de: <https://openwebinars.net/blog/que-es-django-y-por-que-usarlo/> [extraído el 30 de septiembre de 2021]
- Celis, R. 2017. “*Heroku: qué es, cómo funciona y para qué sirve*”. Referencia de: <https://platzi.com/blog/que-es-heroku/> [extraído el 30 de septiembre de 2021]
- Cervantes, H. 2021 “*Arquitectura de software*”. Referencia de: <https://sg.com.mx/revista/27/arquitectura-software> [extraído el 30 de septiembre de 2021]
- CILSA. 2021. “*¿Qué es un sistema operativo?*”. Referencia de: <https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-sistema-operativo/> [extraído el 30 de septiembre de 2021]
- CLR. 2021. “*¿Qué es un servomotor y cuándo se utiliza?*”. Referencia de: <https://clr.es/blog/es/servomotor-cuando-se-utiliza/> [extraído el 30 de septiembre de 2021]
- Coalla, J. 2021. “*React | Qué es, para qué sirve y cómo funciona*”. Referencia de: <https://tech.tribalyte.eu/blog-que-es-react> [extraído el 30 de septiembre de 2021]
- Delgado, A. 2020. “*¿Qué es Raspberry Pi y para qué sirve?*”. Referencia de: <https://www.geeknetic.es/Raspberry-Pi/que-es-y-para-que-sirve> [extraído el 30 de septiembre de 2021]
- GCF Global, 2021. “*¿Qué es hardware y software?*”. Referencia de: <https://edu.gcfglobal.org/es/informatica-basica/que-es-hardware-y-software/1/> [extraído el 30 de septiembre de 2021].

- Google, “Flutter”. Referencia de: <https://flutter.dev/> [extraído el 28 de marzo de 2021].
- López, M. 2020. “¿Qué es un lenguaje de programación?”. Referencia de: <https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/> [extraído el 30 de septiembre de 2021]
- López, S. 2020. “*Firestore: qué es, para qué sirve, funcionalidades y ventajas*”. Referencia de: <https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/> [extraído el 30 de septiembre de 2021]
- MDN. 2021. “*Generalidades del protocolo HTTP*”. Referencia de: <https://developer.mozilla.org/es/docs/Web/HTTP/Overview> [extraído el 30 de septiembre de 2021]
- Ministerio de Salud Pública y Asistencia Social. 2020. “*MSPAS supervisa protocolos COVID-19 en centros comerciales; cinco establecimientos reciben avisos por incumplimientos*”. Referencia de: <https://www.mspas.gob.gt/noticias/noticias-ultimas/5-noticias-mspas/1158-mspas-supervisa-protocolos-covid-19-en-centros-comerciales-cinco-establecimientos-reciben-avisos-por-incumplimientos.html> [extraído el 22 de febrero de 2021]
- Neoland. 2018. “¿Qué es el diseño UX/UI?”. Referencia de: <https://www.neoland.es/blog/que-es-el-ux-ui-design> [extraído el 30 de septiembre de 2021]
- Nephos IT. 2022. “*Infraestructura On-Premise vs Cloud*”. Referencia: <https://www.nephosit.com/infraestructura-on-premise-vs-cloud-caracteristicas-ventajas-y-desventajas/> [extraído el 3 de marzo de 2022]
- Phillips, T. 2021. “*App lets drivers in Hawaii find parking spaces and pay with NFC*”. Referencia de: <https://www.nfcw.com/nfc-world/app-lets-drivers-in-hawaii-find-parking-spaces-and-pay-with-nfc/> [extraído el 30 de septiembre de 2021].
- Phillips, T. 2021. “*Washington Metro lets Android users add SmarTrip transit cards to Google Pay*”. Referencia de: <https://www.nfcw.com/transit-ticketing-today/washington-metro-lets-android-users-add-smartrip-transit-cards-to-google-pay/> [extraído el 30 de septiembre de 2021].
- Red Hat. 2021. “¿Qué es una arquitectura de software?”. Referencia de: <https://www.redhat.com/es/topics/cloud-native-apps/what-is-an-application-architecture#:~:text=Una%20arquitectura%20de%20aplicaciones%20describe,obtenga%20una%20aplicaci%C3%B3n%20bien%20estructurada.> [extraído el 30 de septiembre de 2021]
- Rodríguez, E. 2020. “¿Qué es un framework y por qué es conveniente usarlo?”. Referencia de: <https://www.seoestudios.es/blog/que-es-un-framework/> [extraído el 30 de septiembre de 2021]
- SherWeb. 2022. “*TCO Calculator*”. Referencia: <https://info.sherweb.com/iaas-tco-comparison-tool.html> [extraído el 3 de marzo de 2022].
- Souza, I. 2020. “*Entiende las diferencias entre Front-End y Back-end en el ambiente de los sitios web*”. Referencia de: <https://rockcontent.com/es/blog/front-end-y-back-end/> [extraído el 30 de septiembre de 2021]
- Soy502. 2018. “*Compass, la novedosa opción para pagar parqueos*”. Referencia de: <https://www.soy502.com/articulo/maria-guevara-primera-persona-vacunada-guatemala-100931> [extraído el 22 de febrero de 2021]

- Xataka Movil, 2021. “*NFC en el móvil: que es, para que sirve y siete usos para sacarle todo el partido*”. Referencia de: <https://www.xatakamovil.com/tutoriales/nfc-movil-que-sirve-siete-usos-para-sacarle-todo-partido> [extraído el 30 de septiembre de 2021]

XIII. Anexos

A. Retroalimentación escrita del primer prototipo

Encuestado	¿Qué mejorarías en el diseño de las pantallas?	¿Qué mejorarías de la experiencia de usuario de la aplicación?	¿Algo no funcionó como esperabas?
1	No mejoraría nada.	Por el momento nada.	Todo me funcionó como esperaba, solo me hubiera gustado que cuando terminara el test me indicara que había finalizado la prueba.
2	Algunos dibujos se ven pixeleados (bien poco).	Lo mencionado de las mayúsculas en los nombres automático.	Nada.
3	En la pantalla donde se ingresan todos los datos que se ponga en mayúsculas automáticamente al poner el nombre.	Nada.	Nada.
4	Cambiaría la posición de los botones en el login.	Lo mismo de cambiar la posición de los botones.	Se tardó un poco cuando me registré.
5	Las imágenes no se ven claras.	Todo estuvo bien, el flujo es muy intuitivo.	Nada.

Cuadro #6. Retroalimentación del primer prototipo

B. Retroalimentación escrita del segundo prototipo

Encuestado	¿Qué mejorarías en el diseño de las pantallas?	¿Qué mejorarías de la experiencia de usuario de la aplicación?	¿Algo no funcionó como esperabas?
1	En la primera pantalla la resolución del logo es baja, se ve pixeleado.	En la pantalla para hacer tap para el pago no es suficientemente intuitiva la forma en la que funciona.	Llené el espacio de e-mail y contraseña antes de ver que había un botón para registrarme.
2	Todas están perfectas y son intuitivas.	Tal vez solo cuando entro a la pantalla de	Todo funcionó como esperaba.

Encuestado	¿Qué mejorarías en el diseño de las pantallas?	¿Qué mejorarías de la experiencia de usuario de la aplicación?	¿Algo no funcionó como esperabas?
		home me gustaría que me salieran como flechas hacia los botones indicando que hace cada uno como paso a paso.	
3	Nada, me parece que está bien. Simple y funcional.	En la parte de restablecer contraseña que tenga como debe ser la contraseña(tamaño y que tan compleja) y también en la parte de la tarjeta poner un botón de info para la fecha.	Pues solo cuando se restableció la contraseña tuvo lag.
4	Me gustó mucho la tarjeta que se muestra al agregar o modificar una tarjeta de crédito.	En la creación de cuenta o login me gustaría que los botones estuvieran antes de llenar los espacios.	Todo funcionó como esperaba.
5	La pantalla de recuperación es muy simple.	Nada, todo estuvo bien.	Cuando solicité recuperar contraseña se tardó un poco mas de lo que esperaba.

Cuadro #7. Retroalimentación del segundo prototipo

C. Retroalimentación escrita del tercer prototipo

Encuestado	¿Qué mejorarías en el diseño de las pantallas?	¿Qué mejorarías de la experiencia de usuario de la aplicación?	¿Algo no funcionó como esperabas?
1	Agregarle estilo a la página web de recuperar contraseña.	Agregar instrucciones de lo que debe de contener una contraseña al recuperar contraseña.	Todo funcionó como esperaba.
2	Todo me gusta.	Sería bueno agregarle una forma de	Me dejó abrir la talanquera sin tener

Encuestado	¿Qué mejorarías en el diseño de las pantallas?	¿Qué mejorarías de la experiencia de usuario de la aplicación?	¿Algo no funcionó como esperabas?
		autenticar al empezar la lectura NFC.	tarjetas de credito agregadas.
3	Me gustan mucho como se muestra el estado de la transacción.	Nada. Es muy simple y funcional.	No se puede ver el detalle de la transacción.
4	Cambiaría los colores de la pantalla del home.	Me gustó el tutorial al iniciar sesión.	Todo bien.
5	Todo esta bien.	Sería bueno mostrar el monto que se debe de pagar en las transacciones pendientes.	Me gustó que no tengo que iniciar sesión cada vez que abro la aplicación.

Cuadro #8. Retroalimentación del tercer prototipo

D. Prototipo final

En la siguiente sección se presentan imágenes de la versión final de la aplicación después de la retroalimentación recibida por parte de los usuarios.



Figura 24. Pantalla de On Boarding, paso #1.



Figura 25. Pantalla de On Boarding paso #2.

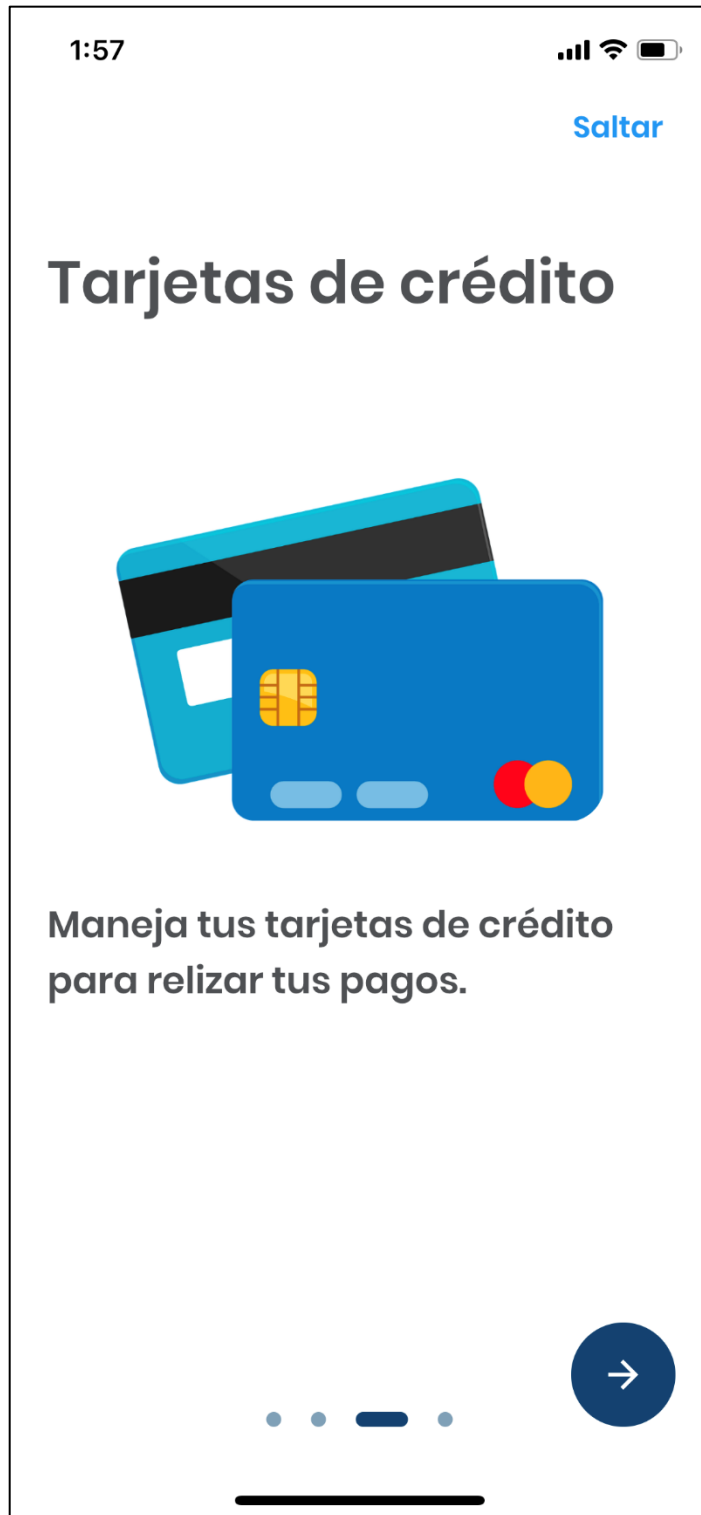





Figura 26. Pantalla de On Boarding paso #3.



Figura 27. Pantalla de On Boarding paso #4.

1:57   


<


Registro

Nombre

Apellido

Correo

Contraseña 

Confirmar contraseña 

Registrarme

Figura 28. Pantalla de registro del usuario.

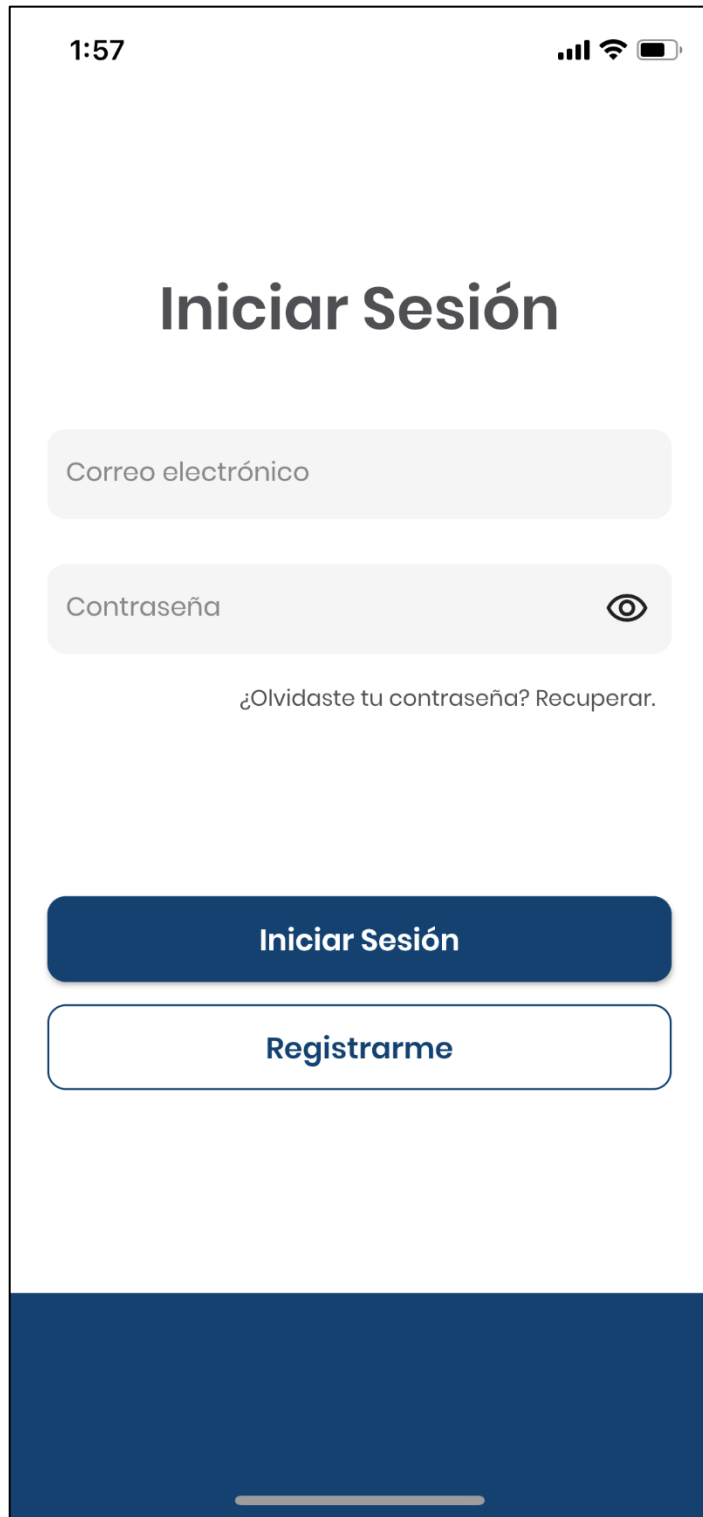


Figura 29. Pantalla de inicio de sesión del usuario.

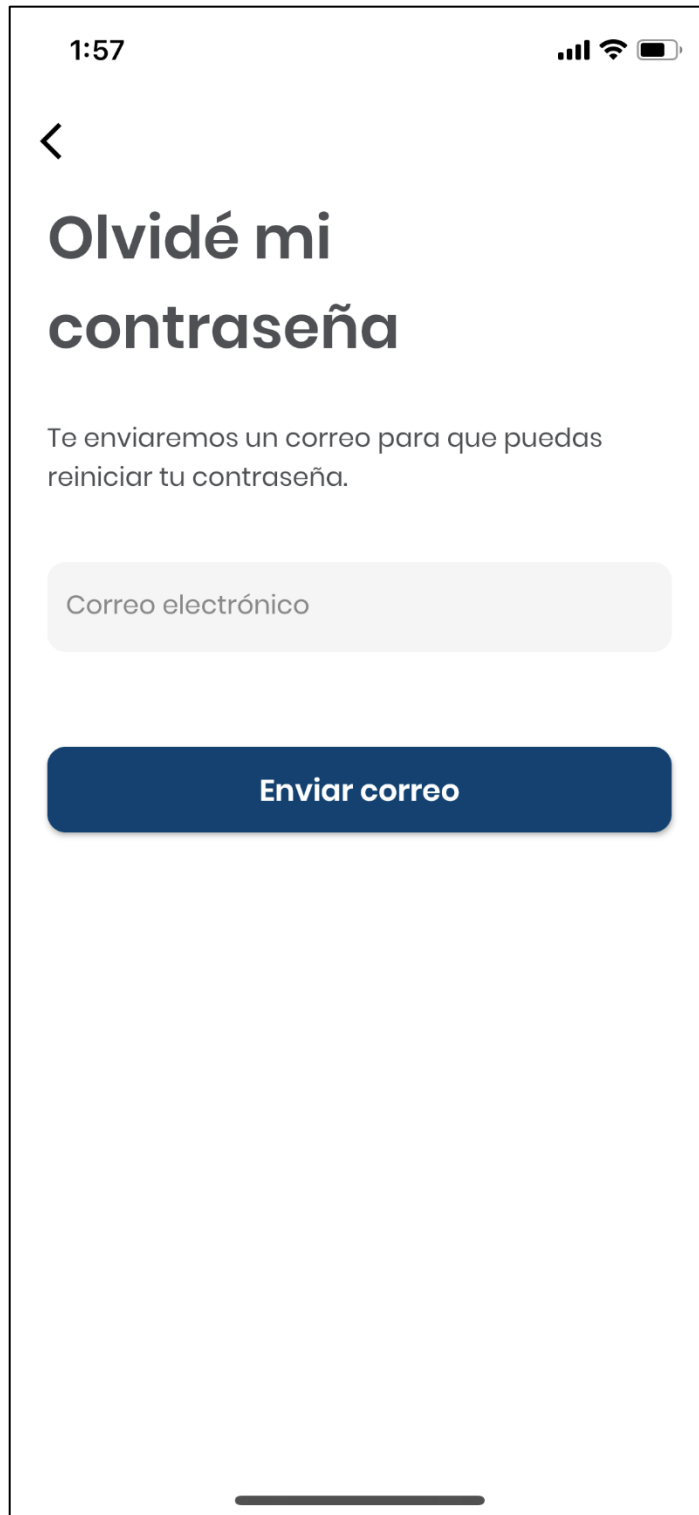


Figura 30. Pantalla de olvidé mi contraseña.

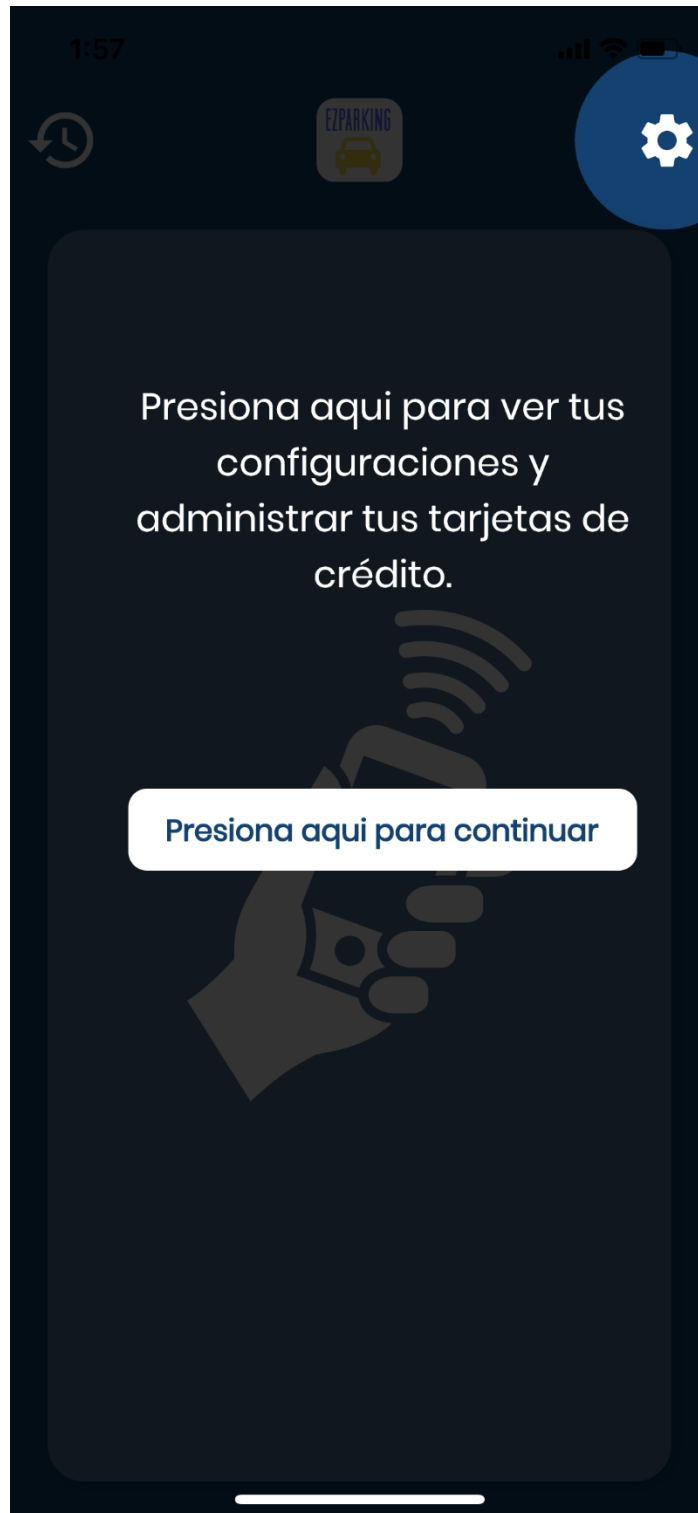


Figura 31. Pantalla de Home con tutorial.



Figura 32. Pantalla de Home.

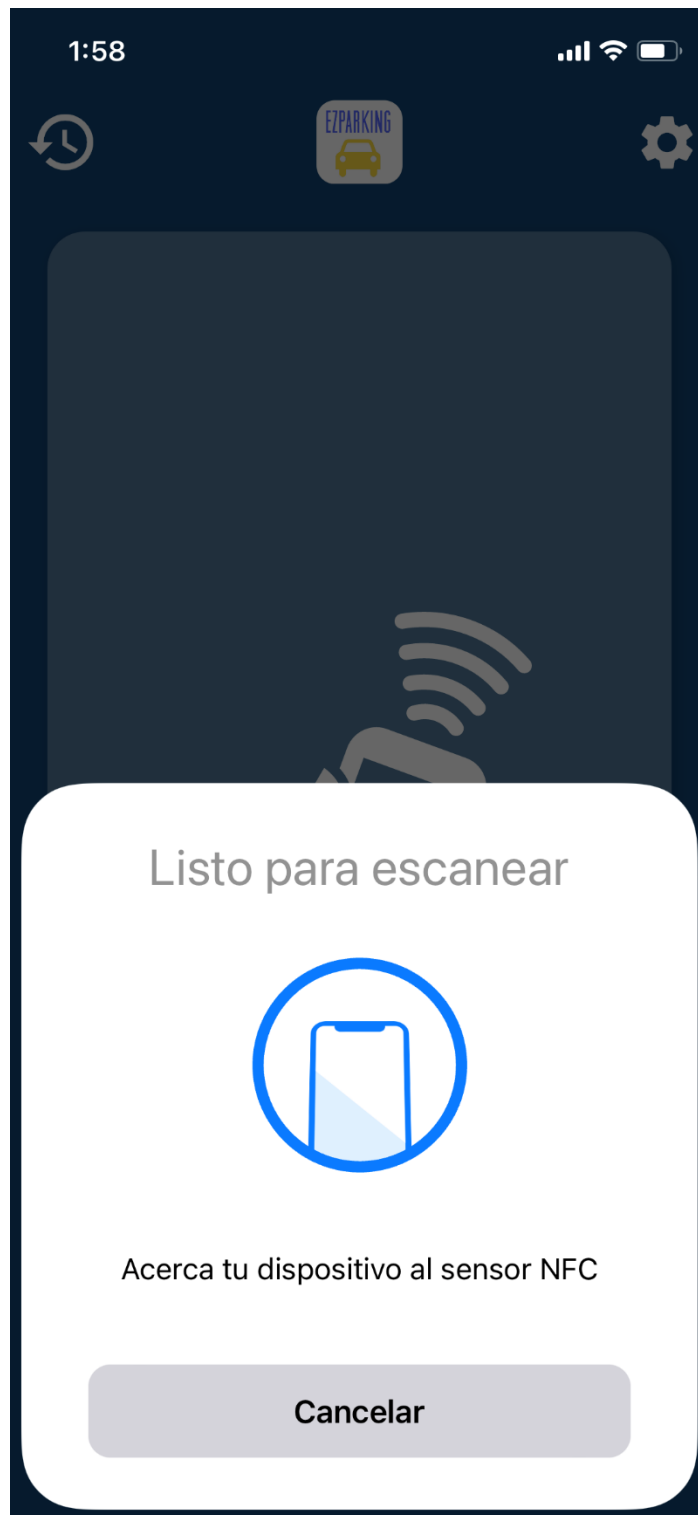


Figura 33. Lectura NFC activada en el dispositivo.



Figura 34. Pantalla de historial de transacciones del usuario.

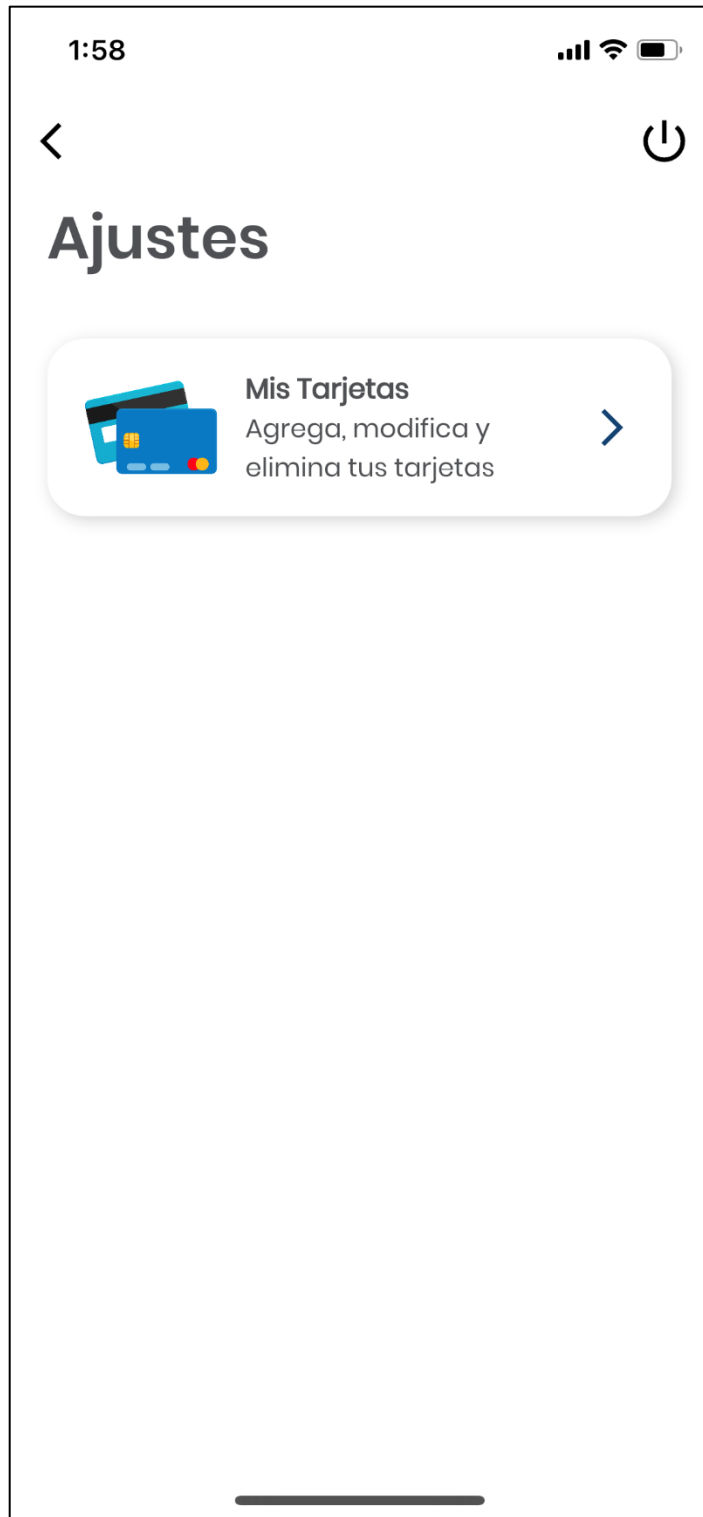


Figura 35. Pantalla de ajustes.

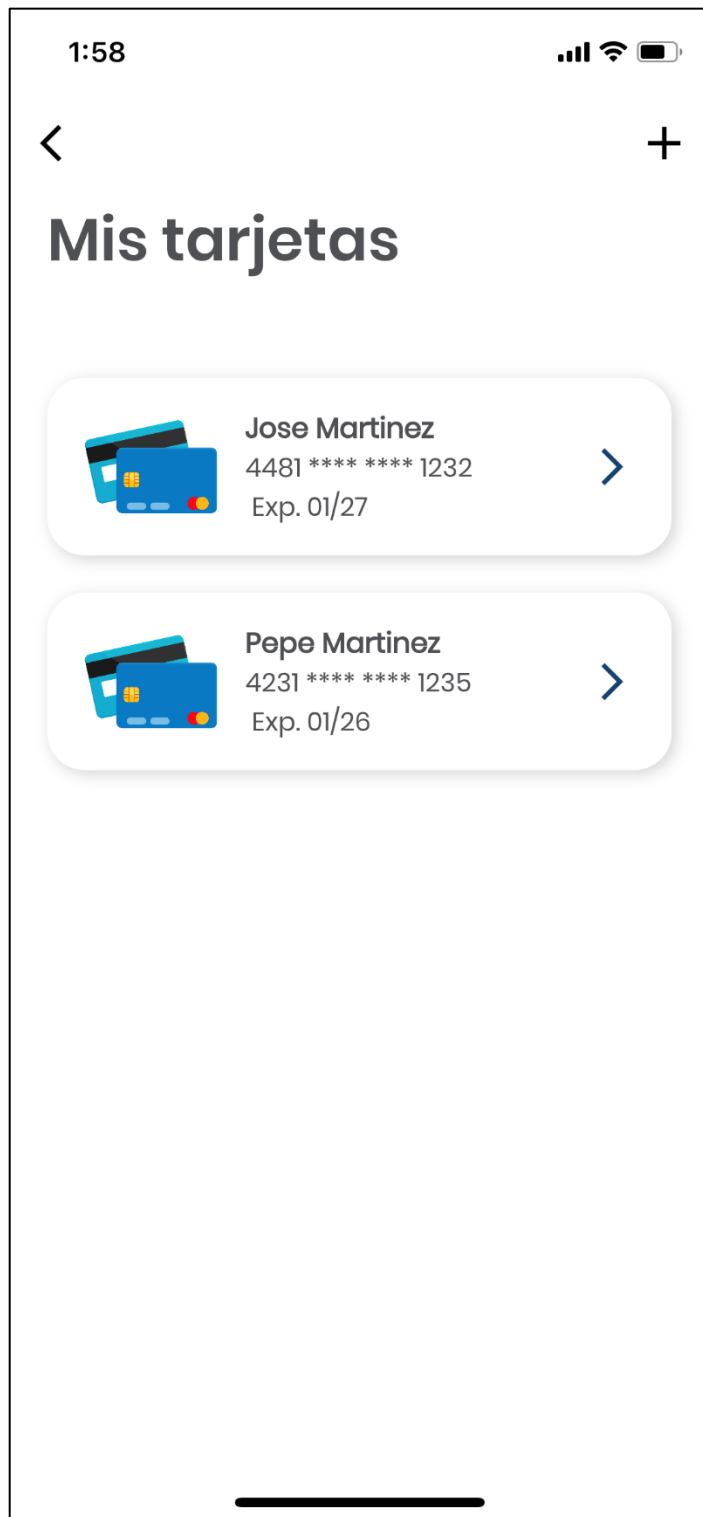


Figura 36. Pantalla de administración de tarjetas de crédito del usuario.



Figura 37. Pantalla para agregar una tarjeta de crédito.



Figura 38. Pantalla para editar o eliminar una tarjeta de crédito.