Desarrollo de software para el aplicativo Rinn Customer App en iOS de la empresa VICA Tecnhology

Informe final
Modalidad: Práctica Profesional

Juan David Hurtado Molano
104615021355

Asesor de la empresa: Ing. Fabián Yesid Vidal
Director: PhD. Sandra Milena Roa Martínez

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa Ingeniería de Sistemas
Popayán, diciembre de 2021

TABLA DE CONTENIDO

Tabla de figuras

Lista de Tablas

Lista de cuadros

1. INTRODUCCIÓN

1.1. PLANTEAMIENTO DEL PROBLEMA

La “Economía colaborativa” hace referencia a los nuevos sistemas de producción de bienes y servicios surgidos a principios de este siglo gracias a las posibilidades ofrecidas por los avances de la tecnología de la información para intercambiar y compartir dichos bienes y/o servicios [1]. Dentro de este tipo de sistemas se encuentran las *lean platform*s, las cuales no son propietarias de ningún recurso mas que del software que estas producen, es decir, no poseen los servicios o productos que ofrecen, pero sí el software que permite la conexión entre la oferta y la demanda [2]. La principal característica de estas plataformas es la externalización de los trabajadores, reconocidos como proveedores independientes [2].

En Colombia existen algunos ejemplos de empresas que han basado su modelo de negocio en este tipo de plataformas tecnológicas como lo son: Rappi, Domicilios.com y Rinn, etc. Esta última, Rinn, es uno de los principales desarrollos que soporta la estrategia de negocio de la empresa VICA Technology, la cual es una *startup* colombiana fundada el 27 de mayo de 2019, dedicada a desarrollar soluciones para negocios y mercados, de crecimiento exponencial en el ultimo año.

Rinn cuenta actualmente con más de 20 mil descargas a través de las principales tiendas de aplicaciones[1] y mas de 200 usuarios diarios activos, sin embargo, su versión nativa en iOS es soportada por una empresa externa a la compañía, la cual se encuentra ubicada en la India, lo que ha generado que, al no poseer personal capacitado para desarrollar nuevas funcionalidades, la empresa VICA Technology se vea obligada a contratar personal por fuera de la organización, es decir, se genera un intermediario considerado innecesario entre las necesidades que la empresa identifica dentro de su mercado y el producto final, y esto conlleva a un elevado costo de mantenimiento en tiempo y dinero de dicha plataforma, convirtiéndose en la principal problemática y motivación que pretende mitigarse con la realización de la presente práctica profesional.

Adicionalmente, la dependencia que existe entre la empresa VICA Technology y la compañía con la cual existe el *outsourcing* genera retrasos con respecto a su plataforma en *Android* puesto que la intensidad horaria que tienen de trabajo en conjunto no sobrepasa las diez horas semanales (debido a la diferencia horaria entre ambos países) y esto conlleva a una baja competitividad en el mercado.

Por otra parte, con respecto a los usuarios de la plataforma y más específicamente la aplicación en *iOS*, optan por no descargar e instalar la versión más reciente en tanto que no posee las ultimas funcionalidades que sí posee el aplicativo en *Android*.

[1] Tomado de su página web: www.rinnapp.com

Por lo anterior, la empresa ha decidido incorporar a un practicante con conocimientos en el desarrollo de aplicaciones móviles iOS nativas para suplir la necesidad anteriormente expuesta y, a futuro, espera poder establecer una relación laboral entre ambas partes.

Vale destacar que, en la presente practica profesional se propone el diseño, desarrollo e integración de nuevas funcionalidades que se requieren dentro del aplicativo móvil que permitan a la empresa mejorar su competitividad dentro del mercado en el que se encuentra.

El desarrollo de la practica profesional estuvo dado por la implementación de funcionalidades que la empresa solicitó y cada una de estas estuvo ligada a una necesidad del mundo real identificada por la compañía. Para esto, el practicante trabajó sobre una plataforma de economía colaborativa que se encuentra dentro del sector del *delivery* llamada Rinn. Para el correcto desarrollo de esta práctica profesional, la empresa brindó apoyo al practicante mediante un tutor, el cual se encargó, entre otras cosas, de realizar el proceso de inducción y posteriormente de programar reuniones semanales para la asignación de tareas y la revisión del trabajo realizado en ese periodo de tiempo por parte del practicante.

Las tecnologías que VICA Technology utiliza, según entrevista con el tutor, para el desarrollo de su versión en iOS son: *UIKit*, el cual es un *framework* que permite construir y gestionar interfaces gráficas mediante el paradigma de programación *event-driven*[2], la librería *Alamofire*, escrita en *Swift* que permite gestionar conexiones HTTP[3] y *RxSwift* para implementar la programación reactiva dentro de la aplicación. Además, se utilizó *Trello* como herramienta que permitió gestionar las tareas a realizar.

Cabe resaltar que la empresa utiliza *Bitbucket* para almacenar los repositorios de toda su plataforma. Esto les permite a los integrantes del equipo y al practicante gestionar todo lo relacionado con el versionamiento de la aplicación con respecto a los objetivos planteados, por lo tanto, las entregas que se realizan al finalizar un tiempo establecido van acorde a lo planeado.

Finalmente, es importante destacar que con este tipo de prácticas profesionales se visibiliza la calidad de los estudiantes del Programa de Ingeniería de Sistemas de la Universidad del Cauca, lo que a futuro abrirá nuevas oportunidades a otros estudiantes y egresados en la empresa VICA Technology que hace parte de la industria de software a nivel internacional.

1.2. METODOLOGÍA

Para la solución del problema, metodológicamente, el presente trabajo de grado se realizó a partir de la agrupación de un conjunto de actividades ejecutadas en tres etapas: 1) Caracterización, 2) Desarrollo de Software y 3) Documentación. Con estas etapas se

[2] Tomado de la documentación oficial de Apple.
[3] Tomado de su repositorio oficial: github.com/Alamofire/Alamofire.

pretende el cumplimiento de los objetivos planteados y requerimientos de la empresa según sus dinámicas de trabajo, a continuación, se listan dichas etapas:

- Etapa de caracterización:

Para poder iniciar el desarrollo de las nuevas funcionalidades requeridas, fue necesario analizar y caracterizar las tecnologías implicadas en el desarrollo del aplicativo. Para esto, la empresa VICA Technology apoyó al estudiante con un grupo de profesionales multiculturales que le guiaron y capacitaron en la o las tecnologías que la compañía utiliza. Además, el practicante tuvo que reportar errores (*bugs*) al equipo de desarrollo a medida que se familiariza con el entorno de trabajo.

  - Principales actividades:
    - Analizar y caracterizar el código fuente del aplicativo.
    - Reportar errores encontrados.

- Etapa de desarrollo de software:

Una vez analizado y caracterizado el aplicativo, se trabajó en los requerimientos establecidos por la empresa.

  - Principales actividades:
    - Diseñar e implementar un nuevo flujo de servicio para soporte de pagos electrónicos y recargas.
    - Analizar e integrar un nuevo método de pago PSE con pasarela de pago ePayco.
    - Revisar y mejorar la usabilidad del carrito de compras con la cual cuenta actualmente el aplicativo.

- Etapa de documentación:

El avance y desarrollo de lo expuesto anteriormente fue reportado mediante informes que el estudiante entregará mensualmente, donde se evidencie el trabajo realizado en cada periodo de tiempo.

Y, por ultimo, paralelo al desarrollo de la practica profesional, se realizaron reuniones periódicas con el tutor y asesor, y se documentaron las actividades realizadas dentro de la empresa.

  - Principales actividades:
    - Elaboración del documento final.
    - Elaboración de informes mensuales.
    - Reuniones periódicas con el asesor y tutor.

1.3. APORTES

Desde la perspectiva computacional, los aportes del presente trabajo de grado se centran en las siguientes habilidades técnicas obtenidas: implementación de patrones de diseño como el patrón *Observer* mediante el uso de la programación reactiva, consumo de

servicios mediante una API REST a través de *API's* propias y de terceros, desarrollo de aplicaciones multi-hilo (mediante las librerías que provee Apple) y principios básicos de usabilidad en aplicativos móviles. Lo anterior utilizando los conocimientos adquiridos a lo largo de la carrera como son: fundamentos de algoritmia, estructuras de datos, programación orientada a objetos para poder modelar las soluciones propuestas y entender la arquitectura de la aplicación, programación funcional, bases de datos relacionales para almacenar datos de manera local, entre otros. Además, se enfatizó, por parte de la empresa, en la resolución de problemas basándose en los siguientes pasos: detección del problema, comprensión del problema (¿por qué sucede?), solución del problema y documentación de este (en este ultimo paso se debe documentar el porqué del problema, la solución implementada y cómo evitar que vuelva a suceder).

Con respecto a la academia, el presente trabajo de grado pretende enaltecer la calidad de los estudiantes del Programa de Ingeniería de Sistemas de la Universidad del Cauca, lo que a futuro generará nuevas oportunidades a otros estudiantes y egresados en la empresa VICA Technology.

Por ultimo, en cuanto al producto generado, la empresa se vio beneficiada de las funcionalidades desarrolladas, ya que cada una de estas va ligada a una necesidad en particular, lo que le permite mejorar su competitividad dentro del mercado en el que se encuentra. Un ejemplo de esto: la funcionalidad de pagos y recargas, que permitió a Rinn ser mucho más que una plataforma de *delivery* y poder abarcar una cantidad mayor de usuarios que con los que cuenta actualmente.

1.4. OBJETIVOS

Los objetivos aprobados en el anteproyecto de este trabajo de grado, por parte del Consejo de Facultad de la Facultad de Ingeniería Electrónica y Telecomunicaciones de la Universidad del Cauca, se presentan a continuación.

1.4.1. GENERAL

Contribuir en la construcción de software que soporte nuevas funcionalidades dentro del aplicativo móvil Rinn Customer App en iOS de la empresa VICA Technology.

1.4.2. ESPECIFICOS

- Caracterizar el estado actual del aplicativo móvil Rinn Customer App respecto a las tecnologías, arquitecturas y patrones utilizados por la empresa.

- Diseñar e implementar los requerimientos de la empresa asociados a método de pago PSE, otros pagos y recargas, usando SCRUM.

- Rediseñar e implementar la billetera móvil y gestión de tarjetas (CRUD) usando SCRUM.

- Documentar la experiencia como estudiante en práctica profesional con respecto a consideraciones de la incorporación en un ambiente laboral real buscando con ello retroalimentar al Programa de Ingeniería de Sistemas y la empresa VICA Technology.

1.5. ORGANIZACIÓN DEL DOCUMENTO

A seguir, se describe de manera general el contenido y organización del presente informe final:

CAPITULO 1: INTRODUCCIÓN: Hace referencia al presente capitulo donde se describe el planteamiento del problema, los aportes generados y los objetivos a cumplir.

CAPITULO 2: MARCO TEÓRICO: En este capitulo se describen los conceptos teóricos más relevantes que se emplearon para la realización del trabajo de grado.

CAPITULO 3: CARACTERIZACIÓN DEL APLICATIVO: En este capitulo se describen las características técnicas del aplicativo tales como: el lenguaje de programación, los patrones utilizados, los servicios de terceros, etc.

CAPITULO 4: DESARROLLO DE FUNCIONALIDADES: En este capitulo se detallan las actividades realizadas y el producto generado a través de cada sprint para poder cumplir con los objetivos del trabajo de grado.

CAPITULO 5: CONCLUSIONES: En este capitulo se describen las conclusiones generadas a partir del desarrollo de practica profesional.

CAPITULO 6: BIBLIOGRAFIA: En el ultimo capitulo del presente documento se listan las referencias bibliográficas de los artículos, libros y demás recursos utilizados para el desarrollo del trabajo de grado.

Cabe aclarar que, aunque se procura explicar a detalle cada una de las actividades y funcionalidades realizadas, se firmó un acuerdo de confidencialidad entre la empresa y el practicante, por lo tanto, el presente documento no contiene información que la compañía considera sensible. Es por esto por lo que, en el capitulo 3, se documentó de forma general y no a detalle cada uno de sus módulos y sus funcionalidades (por ejemplo, realizar un diagrama de componentes de la aplicación no fue posible en tanto que la compañía determinó que este tipo de documentación no cumplía con el acuerdo de confidencialidad anteriormente mencionado).

2. MARCO TEÓRICO

2.1. COMERCIO ELECTRONICO

Según [3], el comercio electrónico se origina de la demanda de las empresas y de la administración, para hacer un mejor uso de la informática y buscar una mejor forma de aplicar las nuevas tecnologías para así mejorar la interrelación entre cliente y proveedor. Para [4], la reformulación de procesos que posibilita el comercio electrónico genera nuevas posibilidades a la hora de redefinir y simplificar los canales de distribución y la gestión de la cadena de suministro. Permite replantear los objetivos en la empresa con un claro direccionamiento estratégico, facilitando crear nuevos productos y mercados, nuevos canales de distribución, reducir el coste de las actividades empresariales y favorecer la apertura de nuevos mercados [5].

El desarrollo del comercio electrónico en América Latina, pero en especial en Colombia, ha logrado una representación importante en la economía del País; la tendencia está dada para seguir creciendo y lograr un posicionamiento dentro del mercado; esto gracias a los operadores o pasarelas de pago electrónico, quienes cada día evolucionan con nuevas e innovadoras cualidades de servicio, hecho que está siendo aprovechado por los empresarios y por el sector financiero para alcanzar competitividad o posicionamiento, además de lograr que la experiencia de comercializar o adquirir productos a través de internet cada vez más fácil [6]. Las pasarelas de pagos permiten a los establecimientos afiliados puedan realizar pagos vía internet para ser competitivos en el mercado y lograr el crecimiento de los ingresos a través del incremento de las transacciones, el adquiriente presenta nuevos modelos de negocios entre la que se encuentra la agregación de comercios [7]. Sobre la oferta local de productos y servicios que se pueden comprar online, vale la pena resaltar que aun hoy, casi una década después de que una primera camada de colombianos realizan compras por Internet, sigue faltando ampliar mucho más la oferta de productos y servicios ofrecidos por empresas colombianas a través de esta practica [8].

2.2. PASARELAS DE PAGO

A través del desarrollo del comercio electrónico, los comercios han desarrollado plataformas que le permitan aceptar pagos electrónicos desde su pagina o portal y es donde para poder satisfacer la necesidad de las empresas de menor tamaño se han venido creando empresas que se encargan de proveer los desarrollos tecnológicos, en Colombia se conocen como pasarelas tipo Gateway las cuales con contratadas por el comercio directamente [9]. Son plataformas de pago claves para el comercio digital o electrónico que se usan hoy en día para el pago de casi cualquier producto o servicio, siendo esencial para las transacciones en línea ya sea en el sitio web, aplicación o generando códigos, recibos entre otros que permiten identificar al comprador y lo comprado [10]. Son un servicio intermediario entre las tiendas en línea y una entidad bancaria, pues a través de esta se permite la transferencia de fondos de un usuario o

cliente hacia otro vendedor [11]. Una de las pasarelas de pago tipo Gateway en Colombia son *ePayco* de Davivienda y *Wompi* de Bancolombia.

2.2.1. **EPAYCO** [12]

La pasarela de pagos ePayco es una solución de pagos y recaudos integral en Colombia que resuelve todas las necesidades de vender y recaudar a través de internet con las mejores gestiones y experiencia para sus clientes. ePayco Brinda las siguientes herramientas para la recepción de pagos de forma fácil y segura:

- Activación instantánea: los clientes de ePayco pueden realizar transacciones en dos (2) minutos. Además permite la incorporación completamente en línea con documentación mínima.
- Integración fácil: garantiza los complementos para las principales plataformas e idiomas. Integración de ePayco en menos de una hora.
- Impulsado por API: permite desarrollar negocios a escala con su automatización completa basada en API que no requiere intervención manual.
- Múltiples medios de pagos: ePayco tiene a su disposición más de veintidós (22) medios de pago disponibles.
- El mejor soporte: se fundamenta por medio de los tickets, teléfono y chat, los cuales siempre están disponibles para ayudar al cliente a solventar cada paso que realice.
- Panel de administración: se refiere a los datos e información en tiempo real en su panel de control de ePayco para tomar decisiones comerciales informadas.
- Seguro: cabalidad y certeza en el cumplimiento estándar en el procesamiento transaccional PCI DSS Nivel 1.

Esta pasarela posee dos (2) modelos de negocios o modelos de procesamiento de transacciones, que se adaptan a las necesidades cliente: agregador o *gateway*. El modelo agregador consta de tres (3) procesos: procesamiento transaccional, compensación en cuentas ePayco y abono posterior a la cuenta de la empresa. El modelo Gateway consta de dos (2) procesos: procesamiento transaccional y compensación en cuentas bancarias de la empresa.

2.2.2. **WOMPI** [13]

La pasarela de pagos *Wompi* es una solución desarrollada por Bancolombia enfocada en agilizar lo negocios de pequeñas y medianas empresas facilitando diferentes medios de pago. Esta pasarela proporciona las siguientes herramientas a los comercios registrados:

- Link de pagos: permite vender por internet sin tener una página web. Los comercios deben compartir con sus clientes el link a través de las redes sociales o correo electrónico.
- Datafono virtual: recibe pagos en el punto de venta del comercio utilizando un celular donde los clientes podrán pagar sin efectivo.

- Pagos en pagina web: permite integrar los servicios de *Wompi* a través de una *API* y un *WebCheckout.*
- *Plugin*: permite integrar el *plugin* de *Wompi* en un e-commerce.

Los clientes de los comercios registrados pueden pagar mediante tarjetas de crédito o debito, tarjetas e-card de Bancolombia, tarjeta e-prepago de Bancolombia, botón Bancolombia, efectivo en corresponsal bancario, Nequi, tarjeta digital Nequi y cuentas de ahorro y corriente mediante PSE.

2.3. RINN[4]

Dentro de los comercios en Colombia que utilizan las pasarelas de pago, se encuentra Rinn, que hace parte de la economía de plataformas (que surgió a principio de siglo en los Estados Unidos y Europa). Dentro de este tipo de economía se pueden realizar distintos tipos de trabajos, pero en el que Rinn se encuentra es conocido como "*Work on demand"* dado que la cantidad de tareas solicitadas no se encuentra previamente determinada, sino que varía de acuerdo con el nivel de demanda de los clientes o consumidores [2]. Rinn, al ser una plataforma de *delivery* bajo demanda, tiene la particularidad de que también depende de la proximidad geográfica entre consumidores y trabajadores [2].

Rinn permite a sus usuarios encontrar comidas, productos y servicios de la zona y recibirlos en la puerta de su casa, y a las empresas acceder a nuevos clientes gracias al fenómeno de la digitalización. Cuenta con más de 20.000 descargas en las principales tiendas de aplicaciones, 240 domiciliarios a quienes denominan como "Rinneros" y con más de 12.500 pedidos realizados. A las empresas que desean ingresar a la plataforma y promocionar sus productos se las conoce como "Aliados Rinn". Entre las principales empresas que se pueden encontrar dentro de la plataforma se encuentran: restaurantes, droguerías y mini-markets. Hasta el momento, se encuentra funcionando en la ciudad de Jamundí y el suroccidente de la ciudad de Cali, a futuro, se espera que Rinn empiece a funcionar en Tuluá.

Esta plataforma cuenta con tres aplicaciones móviles: una para los clientes (disponible para iOS y Android) que permite observar las tiendas disponibles y los productos que estas ofrecen, junto con la funcionalidad que permite realizar pedidos. Otra para los conductores (únicamente disponible en Android). Y la ultima para las tiendas o "Aliados Rinn" registrados (disponible en iOS y Android), donde pueden agregar, modificar o eliminar sus productos y ver las estadísticas más importantes de sus ventas.

Por último, en el año 2021, Rinn fue seleccionada dentro de las 50 startups con mayor potencial de Colombia por iNNpulsa en la categoría de economía naranja donde participaron casi 500 emprendimientos.

[4] Tomado de la pagina oficial de Rinn: www.rinnapp.com

2.4. METODOLOGIAS AGILES

VICA Technology utiliza metodologías agiles para soportar los nuevos *features* de su plataforma, ya que este tipo de enfoque es el que mejor se adapta al desarrollo de aplicativos móviles debido a las características que tienen este tipo de desarrollos: entorno altamente volátil, equipo pequeño de desarrollo, entorno de desarrollo orientado a objetos, sistemas pequeños y pequeños ciclos de desarrollo [14]. Se caracterizan principalmente por: ser tolerantes a los cambios, entregas incrementales y la activa participación del cliente [15]. Presentan como principal característica la flexibilidad, los proyectos en desarrollo son subdivididos en proyectos más pequeños, incluye una comunicación constante con el usuario, son altamente colaborativos [16].

Para el desarrollo funcionalidades de Rinn, la empresa utiliza *Scrum*, el cual es un proceso ágil de desarrollo que, además de facilitar la gestión de proyectos con un rápido cambio de requisitos, permite llevar a cabo desarrollos de software iterativos e incrementales. Por lo general es usado en ambientes de desarrollo ágiles [15]. Es un marco que permite el desarrollo y mantenimiento de productos complejos, reduciendo el riesgo durante la realización de un proyecto trabajando de manera colaborativa [17]. En *Scrum*, el trabajo es realizado por equipos auto-organizables en iteraciones, iniciando con la planificación y terminando con una reunión retrospectiva, donde cada iteración cuenta con un conjunto de funcionalidad que han de ser implementadas, estas, a su vez, se encuentran registradas en el *Product Backlog,* que define todo el trabajo requerido para completar el proyecto [18].

Según [19] el desarrollo de esta metodología viene dado de la siguiente manera:

- Crear una pila de producto que es una lista priorizada de las características o funcionalidad de que deberá tener el producto y las cuales se obtienen de los usuarios potenciales, los colegas y otras personas relevantes al producto.
- Realizar el trabajo en cortos ciclos iterativos generalmente llamado iteración o sprint. El resultado de cada iteración debe ser un producto listo para entregar.

Entre los principales roles utilizados en *Scrum* se encuentran el *Product Owmer*, el Scrum *Master* y el Scrum Team. El *Product Owner* decide qué elementos del *backlog* han de ser desarrollados en el próximo sprint por parte de los miembros del *Scrum Team*, quienes participan en la estimación, coordinación, ejecución y seguimiento del trabajo diario durante el proceso de desarrollo. Uno de los miembros del equipo, el *Scrum Master*, se encarga de garantizar que el proyecto es llevado a cabo acorde a los valores, prácticas y reglas definidos por *Scrum*, y que el mismo progresa según lo planificado. Este último rol, además, interactúa tanto con el equipo de desarrollo como con el cliente, procurando resolver cualquier impedimento que obstaculice el avance del trabajo de forma efectiva [18].

En [19] también se resalta que el beneficio de trabajar por iteraciones es que cada una de ellas tiene como resultado un producto listo para entregar, de tal manera que, si la

siguiente iteración genera un software inestable y/o con muchos errores, el equipo simplemente se revierte al ultimo hito sin necesidad de empezar de cero.

2.5. SWIFT

"El código de Swift fue diseñado para ser seguro, y aun así produce software que corre a la velocidad de la luz"[5]. Fue introducido por Apple en la WWDC (Conferencia anual de desarrolladores de Apple) de 2014 y al año siguiente se anunció que se convertiría en un proyecto de código abierto. Actualmente se encuentra en su versión 5.5. Es el resultado de amplias investigaciones sobre lenguajes de programación, combinadas con décadas de experiencia de construcción en plataformas Apple. Los parámetros nombrados se expresan en una sintaxis limpia que hace que las *API* en Swift sean aún más fáciles de leer y mantener. Aún mejor, puntos y comas no son necesarios. Los tipos inferidos hacen que el código sea más limpio y menos propenso a errores, mientras que los módulos eliminan encabezados y proporcionan espacios de nombres. Para admitir mejor los idiomas internacionales y los *emojis*, los *Strings* soportan *Unicode* y utilizan una codificación basada en *UTF-8* para optimizar el rendimiento para una amplia variedad de casos de uso. La memoria se gestiona automáticamente utilizando un recuento de referencias (ARC) estrecho y determinista, manteniendo el uso de memoria al mínimo sin la sobrecarga de la recolección de basura [20].

Según [21], la Tabla 1 contiene las mejoras más relevantes que posee Swift con respecto a su predecesor (Objective-C):

Tabla 1 - Mejora de Swift con respecto a Objective-C

Mejora	Descripción
Inferencia de tipo	Swift puede deducir el tipo de una variable o contante, basado en su valor inicial.
Generalización	Permite escribir código que realiza idénticas tareas para distintos tipos de objetos.
Opcionales	Es posible definir variables que puede que no tengan un valor.
Tuplas	Las funciones pueden retornar varios tipos de objetos mediante el uso de tuplas.
Sobrecarga de operadores	Las clases pueden tener su propia implementación de operadores ya existentes
Protocolos	Introducido por Apple en la segunda versión del lenguaje, es una nueva forma de escribir aplicaciones puesto que no solo se introdujo un nuevo tipo de dato, sino un nuevo paradigma de programación (programación orientada a protocolos).

Estas mejoras anteriores fueron posibles dado que Swift no mantiene la misma compatibilidad con C como sí la tenia Objective-C, por lo tanto, Apple tenia plena libertad

[5] Tomado de la documentación oficial de Apple.

de agregar cualquier tipo de funcionalidad al lenguaje [21]. Una frase que sintetiza bien lo anterior es: "*Swift* es *Objective-C* pero sin la C"[6].

Ahora bien, dentro del mundo del desarrollo del ecosistema Apple existen distintos *frameworks* que permiten desarrollar aplicativos, tales como: *AppKit* para el desarrollo de apps en *MacOS*, *UIKit* para el desarrollo de apps en iOS y tvOS, y *SwiftUI* para el desarrollo de apps multiplataforma (*iOS, iPadOS, MacOS, tvOS y WatchOS*).

2.6. UIKIT[7]

El framework UIKit proporciona la infraestructura necesaria para el desarrollo de aplicaciones iOS o tvOS. Proporciona la arquitectura de ventana y vista para implementar su interfaz, la infraestructura de manejo de eventos para entregar Multi-Touch y otros tipos de entrada a una aplicación, y el bucle de ejecución principal necesario para administrar las interacciones entre el usuario, el sistema y la aplicación. Otras características que ofrece el framework incluyen soporte de animación, soporte de documentos, soporte de dibujo e impresión, información sobre el dispositivo actual, gestión y visualización de texto, soporte de búsqueda, soporte de accesibilidad, soporte de extensión de aplicaciones y gestión de recursos.

La estructura de las aplicaciones UIKit se basa en el patrón de diseño Model-View-Controller (MVC) presentado en la **Figura 1**, en el que los objetos se dividen por su propósito. Los objetos modelo gestionan los datos y la lógica de negocio de la aplicación. Los objetos de vista proporcionan la representación visual de sus datos. Los objetos controlador actúan como un puente entre el modelo y los objetos de la vista, moviendo datos entre ellos en los momentos apropiados.

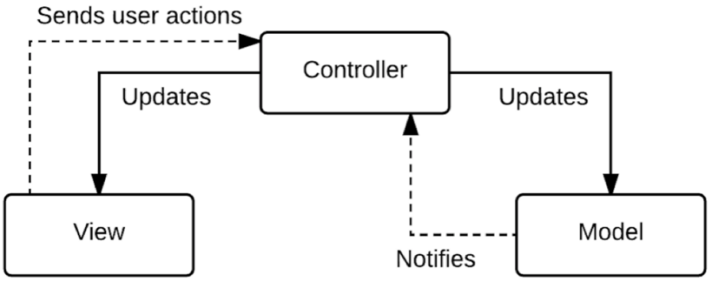


Figura 1. Patrón MVC de Apple

UIKit provee la mayoría de clases necesarias para los objetos de la vista y del controlador. Para la vista, se tiene UIView, que son las vistas con las que interactua el usuario, es por esto que sus responsabilidades se enmarcan dentro de las siguientes: dibujar y animar en pantalla, configurar el *layout* de sus sub-vistas y responder a eventos del usuario. Para el controlador, provee UIViewController cuyas responsabilidades principales son las siguientes: actualizar el contenido de las vistas (usualmente esto

[6] Frase dicha por un empleado de Apple en la presentación de Swift en la WWDC de 2014.
[7] Tomado de la documentación oficial de UIKit: https://developer.apple.com/documentation/uikit

responde a los cambios contenidos en el modelo), responder a las interacciones del usuario, redimensionar las vistas y coordinar otros objetos dentro de la aplicación.

Para el modelo, *UIKit* no provee clases especificas ya que estas van a variar según el tipo de lógica de negocio de se vaya a implementar, por lo tanto, queda en manos del desarrollador la creación de este tipo de objetos.

2.7. PATRÓN *OBSERVER* [22]

Este patrón es muy utilizado dentro del desarrollo de aplicaciones móviles modernas de iOS en conjunto con UIKit, prueba de esto es una de las librerías de terceros más populares como lo es *RxSwift.* Se utiliza cuando existe una relación de uno a muchos entre objetos, de tal manera que si un objeto es modificado, los objetos dependientes son notificados automáticamente. En la **Figura 2** se puede apreciar un diagrama de clases que contiene un ejemplo de lo anterior.

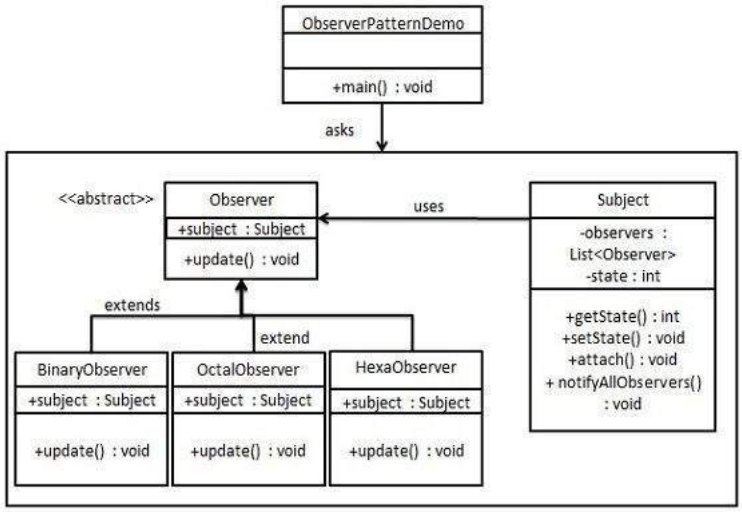


Figura 2. Patrón observer

Está compuesto por tres (3) componentes. Observer, *Client* y *Subject,* que es un objeto que contiene unos métodos que permiten suscribir unos observadores u *Observers* a un objeto cliente. Por último, cabe resaltar que este patrón hace parte de los patrones de diseño comportamentales.

3. CARACTERIZACIÓN DEL APLICATIVO

Al iniciar la practica profesional, la empresa, a través del desarrollo de pequeñas funcionalidades, permitió caracterizar el aplicativo para poder tener un conocimiento más amplio con respecto a las tecnologías con las que fue desarrollado, su patrón de arquitectura, los servicios de terceros que contrata y los módulos que lo componen. En esta sección serán descritos cada uno de estos elementos en detalle que permitan identificar los diferentes componentes y caracterizar la aplicación.

3.1. PATRON DE ARQUITECTURA

Para el desarrollo de Rinn en su versión en iOS se utilizó Swift, lenguaje de programación de código abierto (originalmente desarrollado por Apple), en su versión 4.2, UIKit con el uso de Storyboards para la implementación de la interfaz gráfica y la versión mínima de iOS requerida es la 11.0 (equivalente al 99,90% de todos los iPhone del mercado [22]).

Como se menciono en la sección 2.6, la estructura de las aplicaciones que utilizan UIKit está basada en el patrón *MVC* (Model-View-Controller) presentado en la Figura 1. Aunque esta arquitectura funciona bien, no provee la flexibilidad y reusabilidad necesaria en proyectos a gran escala y esto puede generar problemas en el proceso de desarrollo [23]. Es por esto por lo que Rinn, utiliza una variante al MVC: MVVM (Model-View-ViewModel), donde el modelo y la vista tienen la misma función que MVC y, en vez de tener un controlador, se tiene un View Model.

Esta arquitectura está compuesta de tres componentes principales:

- **View:** Es la vista del aplicativo y punto de entrada de la aplicación, es lo que usuario ve y con lo que este interactúa. Observa los cambios realizados en el *ViewModel* y se actualiza al momento en que se le notifica.
- **ViewModel:** Funciona como puente o interlocutor entre la vista y la lógica de negocio. Permite acceder y/o modificar los datos contenidos en el modelo. También, notifica a la vista en caso de que el modelo haya sido modificado.
- **Model:** Contiene la lógica de negocio del aplicativo. Notifica al *ViewModel* en el momento en que haya sido modificado.

Lo anterior puede apreciarse en la **Figura 3**, donde se presenta la interacción de los distintos componentes de esta arquitectura entre sí.

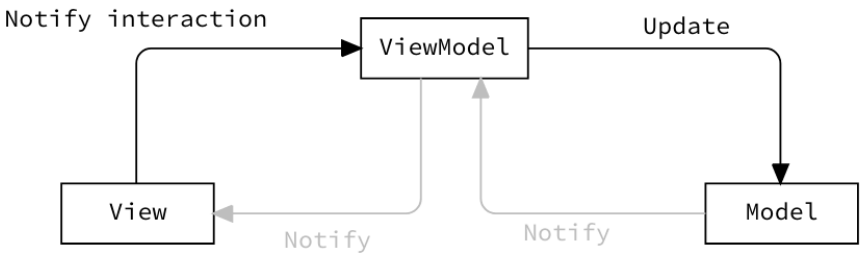


Figura 3. Arquitectura MVVM

Para el caso especifico de UIKit, es preciso mostrar en la **Figura 4** la variante a la **Figura 3**, donde se puede apreciar la incorporación de un nuevo componente llamado *ViewController* que, en este caso (es posible que realice tareas distintas), se encarga de actualizar la vista junto con las referencias a los componentes visuales o vistas de los Storyboards mediante *IBAction (acciones que se ejecutan al ocurrir un evento. Ej: dar click en un botón)* y *IBOulet (referencias a componentes visuales)*. En palabras más simples, es el controlador de la vista, pero al estar tan ligada a la misma se los toma como uno solo.

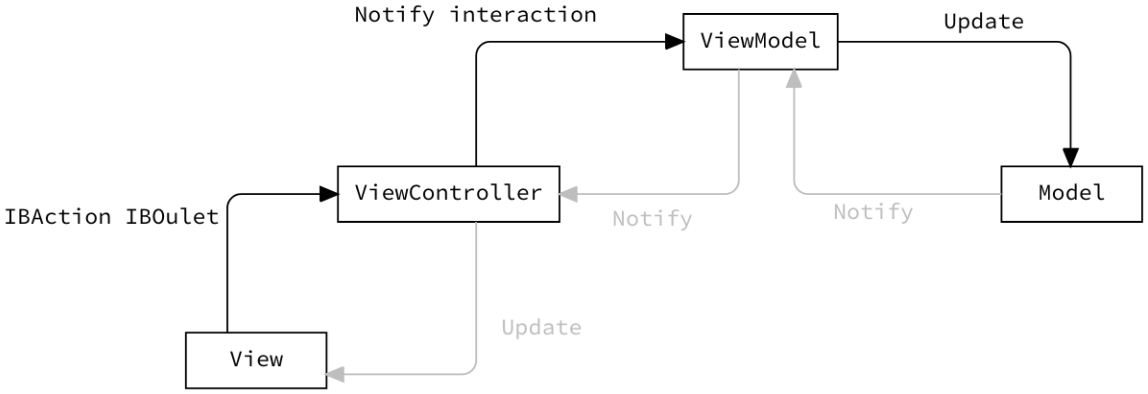


Figura 4. Arquitectura MVVM con UIKit

El ViewModel se comunica con las Vista mediante *data binding*, es decir, enlazar objetos de la vista con objetos del modelo de tal forma que en el momento en que los datos dentro del Modelo sean modificados, la vista los refleje automáticamente. El problema con *UIKit* es que no es reactivo por defecto, por lo tanto, resulta necesario la utilización de librerías de terceros que, en términos generales, permiten realizar la implementación el patrón *Observer*. En el caso de Rinn, se utiliza las librería de código abierto *RxSwift* y *RxCocoa.*

Po ultimo, se puede decir que la arquitectura MVVM es muy poderosa, ya que permite mantener el estado de cada componente de la vista separado y el data *binding* es una manera extremadamente conveniente de solucionar el problema de la comunicación entre la vista y el modelo. Además, ayuda a los desarrolladores a escribir un código mucho más testeable [23].

3.2. SERVICIOS DE TERCEROS QUE UTILIZA RINN

Para que Rinn pueda funcionar correctamente, VICA Technology contrata servicios de terceros para poder disminuir sus costos de operación y que además le permiten monitorear su aplicación en tiempo real. A continuación, se documentan estos servicios que Rinn utiliza en sus distintas plataformas:

3.2.1. FIREBASE[8]

Para VICA Technology, la mejora continua de sus productos es un objetivo constante, por lo tanto, es esencial monitorear el comportamiento de sus aplicativos que se encuentran en distintas plataformas para poder entregar un producto de calidad a sus clientes. Es por esto por lo que se utiliza los servicios de *Firebase*. En su aplicación de iOS, utiliza el SDK que provee *Google* en su versión 6.29.0.

Firebase es una plataforma en la nube soportada por el equipo de Google que contiene más de una decena de herramientas que permiten desarrollar aplicaciones móviles (iOS/Android) y web de alta calidad. Estas herramientas están divididas en tres productos: compilación, lanzamiento y supervisión, y participación, los cuales se describen a continuación:

3.2.1.1. Compilación

Hace referencia a las herramientas que permiten acelerar y escalar el desarrollo de aplicaciones sin la necesidad de administrar su infraestructura. Esto quiere decir que, si una aplicación cuyo backend está soportado únicamente por Firebase, no será necesario preocuparse por el mantenimiento de los servidores, la lógica de negocio del lado del servidor, las notificaciones push e incluso el almacenamiento de los datos.

Dentro de las herramientas que contiene este producto, Rinn utiliza únicamente *Firebase Cloud Messaging* para poder enviar notificaciones de manera segura a las distintas plataformas donde se encuentra. Estas pueden ser gestionadas de manera centralizada desde la consola de Firebase.

3.2.1.2. Lanzamiento y Supervisión

Hace referencia a las herramientas que permiten realizar lanzamientos (*deploys*) con confianza y supervisar el rendimiento y la estabilidad de un aplicativo. Dentro de las herramientas que contiene este producto, Rinn utiliza *Crashlytics* para poder hacer seguimiento en tiempo real de los problemas que afectan directamente a la estabilidad de la plataforma, esto se realiza mediante los metadatos que *Firebase* obtiene de los dispositivos de los usuarios tales como: especificaciones del dispositivo, fecha, versión del aplicativo, etc.

3.2.1.3. Participación

Hace referencia a las herramientas que permiten aumentar la participación de los usuarios con estadísticas enriquecidas, pruebas A/B y compaña de mensajes. Gracias a esto se puede comprender mejor a los usuarios finales para poder personalizar la aplicación y mejorar aspectos como la asistencia y retención.

[8] Información obtenida de la documentación oficial de Firebase: https://firebase.google.com/docs

Dentro de las herramientas que contiene este producto, Rinn utiliza *Google Analytics* para poder supervisar el uso de la aplicación en tiempo real mediante la metadata que esta herramienta provee, por ejemplo: tiempo de sesión de los usuarios, ubicación geográfica, el numero de veces que la aplicación fue ejecutada por primera vez, etc.

3.2.2. ONESKY[9]

OneSky es una plataforma en la nube que provee servicios de traducción para todo tipo de negocios. Soporta más de 50 idiomas y cuenta con más de 1000 traductores alrededor del mundo. Su aplicación web permite crear proyectos para gestionar traducciones de manera centralizada, esto resulta bastante útil para Rinn puesto que, por ejemplo, se pueden cambiar elementos gráficos de texto que generalmente son estáticos (*labels*) sin tener que lanzar una nueva versión del aplicativo.

Esta plataforma funciona como un diccionario de la forma llave-valor, es decir, una traducción posee un identificador único y tiene asociado un valor para cada idioma soportado por el proyecto. Esto quiere decir que existe un diccionario por cada idioma, un ejemplo de lo anterior se puede apreciar a continuación:

Supóngase que se quiere traducir lo siguiente: "*Hola*" y "¿Cómo estás?". Para esto se crea un diccionario con un registro por cada traducción.

Tabla 2. Ejemplo OneSky

Diccionario español	
Identificador	**Valor**
"Saludo"	"Hola"
"Pregunta"	"¿Cómo estás?"

De esta forma, desde el aplicativo iOS mediante el SDK de OneSky, se llama al diccionario del lenguaje solicitado utilizando el identificador requerido y se le muestra al usuario el valor que retorna. Por ultimo, cabe resaltar que Rinn soporta, hasta el momento, los idiomas español e inglés

3.3. MÓDULOS DE RINN

A continuación, se presentan los distintos módulos con los cuales está compuesto Rinn en su versión en iOS.

3.3.1. AUTENTICACIÓN

El módulo de autenticación está compuesto por el inicio de sesión y el registro de usuarios dentro del aplicativo. Cada usuario es identificado mediante su numero de celular y su correo electrónico. Además, las sesiones son únicas, es decir, no puede haber más de una sesión activa a la vez.

[9] Tomado de la pagina web de OneSky: https://www.oneskyapp.com

En la plataforma iOS de Rinn, existen dos tipos de usuarios: *customer*, que hace referencia a un usuario que ya se encuentra registrado en la aplicación y *guest*, que hace referencia a un usuario no registrado (este tipo de usuario puede ver los productos y servicios que presta la plataforma, pero debe iniciar sesión para poder adquirir alguno de ellos).

En el inicio de sesión los usuarios deben ingresar su numero de celular y en caso de que estén registrados, se desplegara el campo de texto donde deberán ingresar su respectiva contraseña, esto puede apreciarse en el formulario de la **Figura 5**.

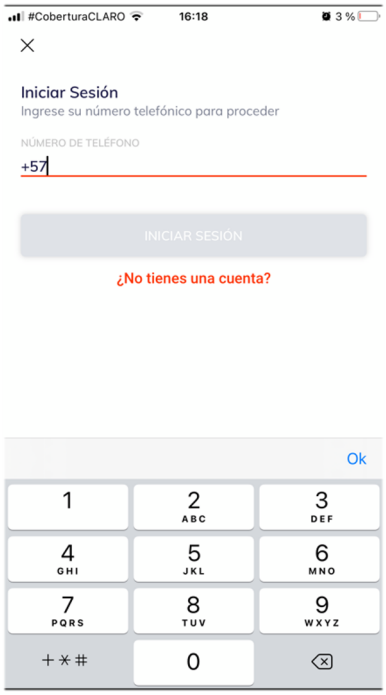


Figura 5. Inicio de sesión

Por ultimo, en el registro los usuarios deben ingresar su nombre, correo electrónico, celular (por el momento, únicamente los números de Colombia se encuentran soportados), contraseña y un código de referido o código de cupón (estos últimos permiten a los usuarios obtener descuentos y recargas en su billetera movil), esto puede apreciarse en el formulario de la **Figura 6**.

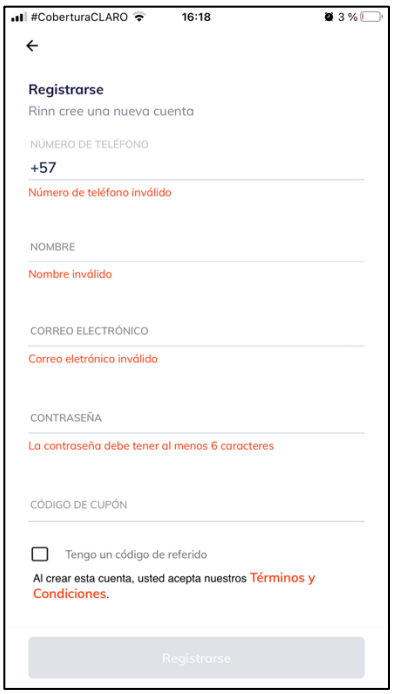


Figura 6. Registro

Una vez el usuario ingresa sus datos y han sido validados exitosamente. El sistema enviará un código SMS al celular ingresado el cual deberá ser ingresado en la aplicación como se puede apreciar en la **Figura 7**. Esta validación se realiza para poder verificar que la persona sí es dueña del celular.

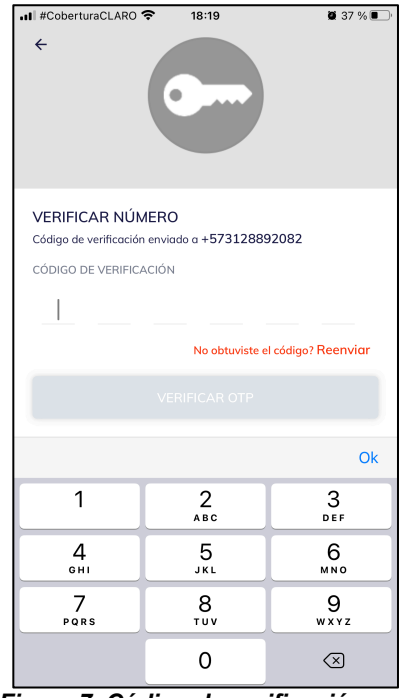


Figura 7. Código de verificación

Cabe aclarar que es posible que un usuario sea baneado por los administradores del sistema, en cuyo caso, el usuario debe comunicarse con el equipo de ayuda de la plataforma para poder volver a ingresar.

3.3.2. GESTIÓN DE PERFIL

Hace referencia a todas las funcionalidades que le permiten a un usuario modificar o actualizar su perfil dentro de la plataforma. Dentro de estas se encuentran el actualizar su foto de perfil, su correo electrónico y su numero de celular como se puede apreciar en la **Figura 8**.

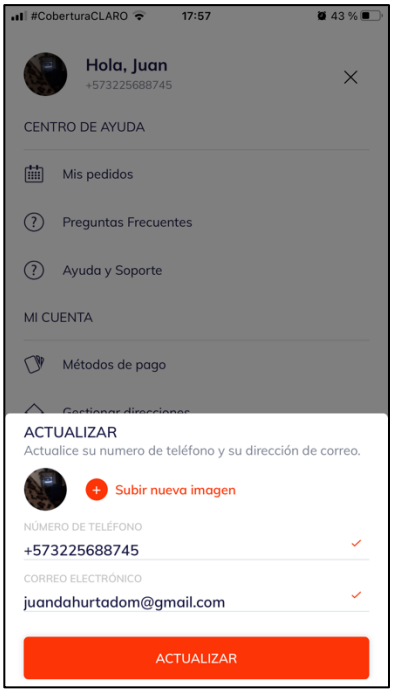


Figura 8. Actualización de perfil

Además, desde su perfil, el usuario puede acceder a sus métodos de pago (billetera y tarjeta de crédito/debito) y a la gestión de sus direcciones.

3.3.3. BILLETERA

Cada usuario en Rinn cuenta con una billetera móvil a la cual puede acceder desde su perfil o la pantalla principal del aplicativo. Esta billetera permite al usuario pagar pedidos dentro de la plataforma de forma parcial o total. Como puede apreciarse en la **Figura 9**, al ingresar a esta, el usuario puede visualizar su balance actual, un acceso directo a sus últimos movimientos y las tarjetas de crédito / debito (previamente registradas) con las cuales puede recargar su cuenta.

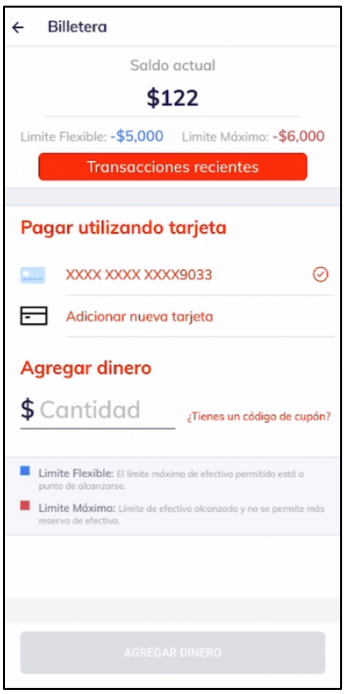


Figura 9. Billetera

Al dar clic en el botón “Transacciones recientes”, el usuario es dirigido a la pantalla del historial de movimientos de la billetera como puede verse en la **Figura 10**. Estos están divididos en dos categorías: créditos que hacen referencia a las recargas que se han realizado y débitos que hacen referencia a las ordenes cuyo método de pago seleccionado fue la billetera.

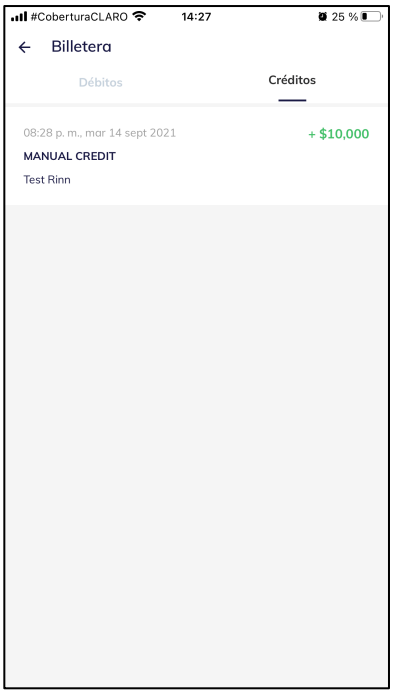


Figura 10. Historial de movimientos

Cabe aclarar que es posible que los administradores de la plataforma realicen créditos y débitos a las billeteras de los usuarios de forma manual.

3.3.4. GESTION DE TARJETAS

Hace referencia al CRUD de las tarjetas crédito / debito de los usuarios de la plataforma. La pantalla principal de este modulo le muestra al usuario sus tarjetas registradas y la opción que le permite adicionar, como se puede ver en la **Figura 11**.

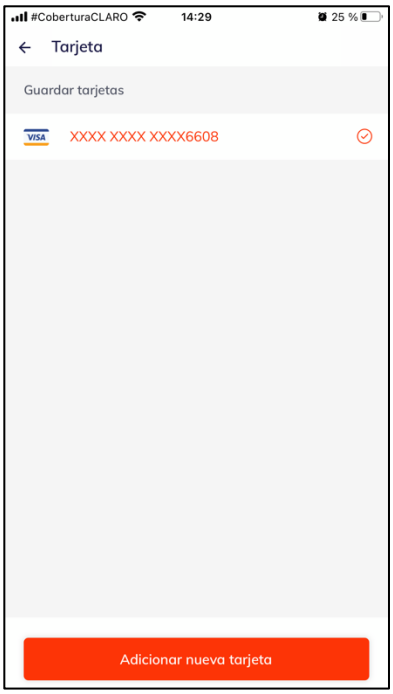


Figura 11. Lista de tarjetas

Un usuario puede registrar n tarjetas ingresando los datos los siguientes datos: numero de la tarjeta, fecha de expiración y CVV (código de seguridad de las tarjetas) esta funcionalidad se traduce en el formulario que se encuentra en la **Figura 12**. Además, es importante resaltar que este servicio es proporcionado por ePayco.

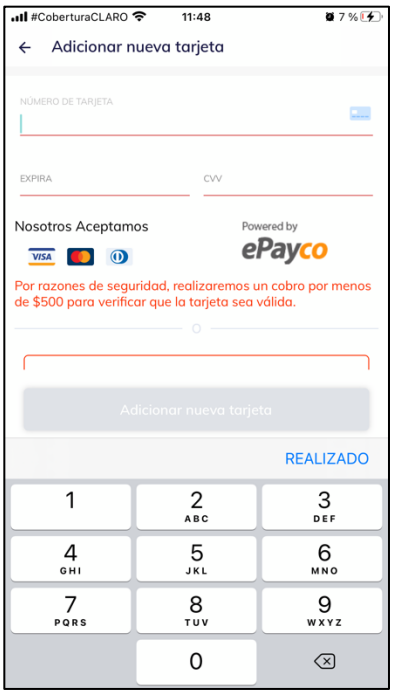


Figura 12. Agregar tarjeta

Para poder acceder al detalle de una tarjeta, el usuario debe seleccionar una tarjeta de la lista que se encuentra en la **Figura 11**. Dentro del detalle, que se encuentra en la **Figura 13**, se puede observar la marca de la tarjeta (por el momento, Rinn soporta VISA, MASTERCARD y DINERS), sus últimos 4 dígitos, la opción que permite eliminarla y la opción que permite seleccionar esa tarjeta como tarjeta por defecto.

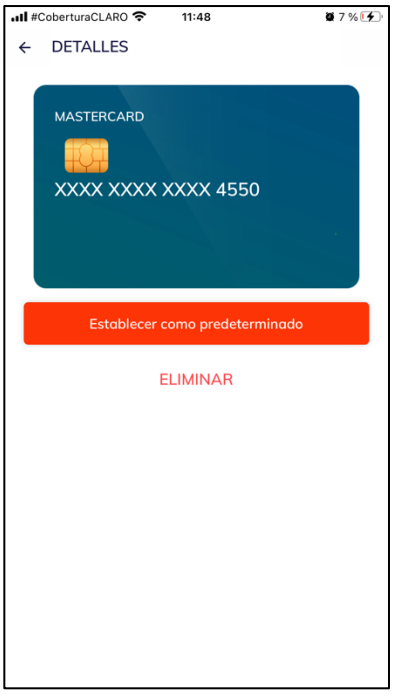


Figura 13. Detalle de tarjeta

Por ultimo, el que una tarjeta sea por defecto significa que, al momento de pagar una orden, si se desea pagar con tarjeta, por defecto se seleccionará esa tarjeta para realizar el pago.

3.3.5. CARRITO DE COMPRAS

El carrito de compras, presentado en la **Figura 14**, puede contener cuantos artículos se deseen, pero de una única tienda, por lo tanto, si se quiere realizar pedidos que contengan productos de varias tiendas tendrán que ser creadas varias ordenes: una por cada tienda.

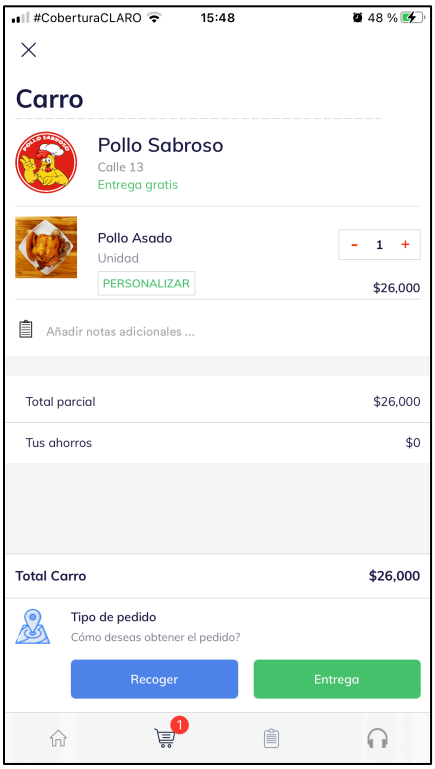


Figura 14. Carrito

El *checkout* o confirmación de orden que aparece justo antes a la creación de cada orden que se presenta en la **Figura 15**, está compuesto por los siguientes campos obligatorios: lugar de entrega (que accede al modulo de gestión de direcciones para seleccionar la dirección requerida) y método de pago. Además, el campo código promocional (opcional) que le permite al usuario recibir un descuento en su orden.

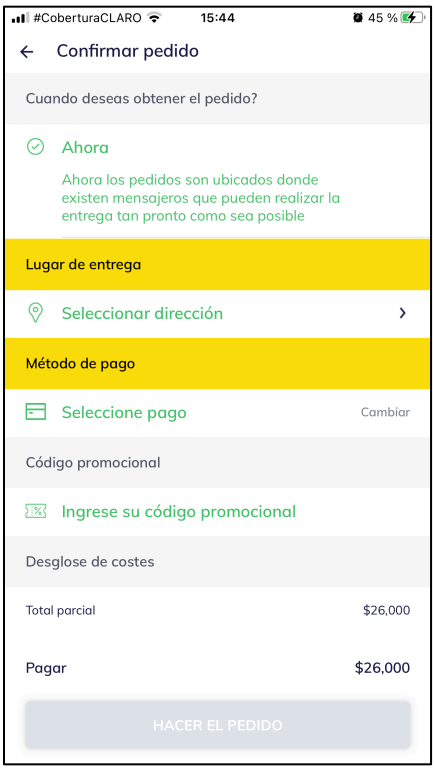


Figura 15. Checkout

3.3.6. TIENDAS

Es el módulo mas grande del aplicativo y lo componen las distintas tiendas que se han registrado en la plataforma. Estas están divididas en las siguientes categorías: restaurantes, ágil mercado, licores, farmacia, belleza, heladería y mascotas. A las que se pueden acceder desde la pantalla principal como se puede observar en la **Figura 16**.

Figura 16. Home

Cada categoría contiene una o muchas tiendas, en la **Figura 17**, se puede apreciar las tiendas disponibles en la ciudad Jamundí dentro de la categoría de restaurantes. En ella se puede apreciar que no solo se muestran las tiendas, sino que también las promociones más destacadas de las mismas.

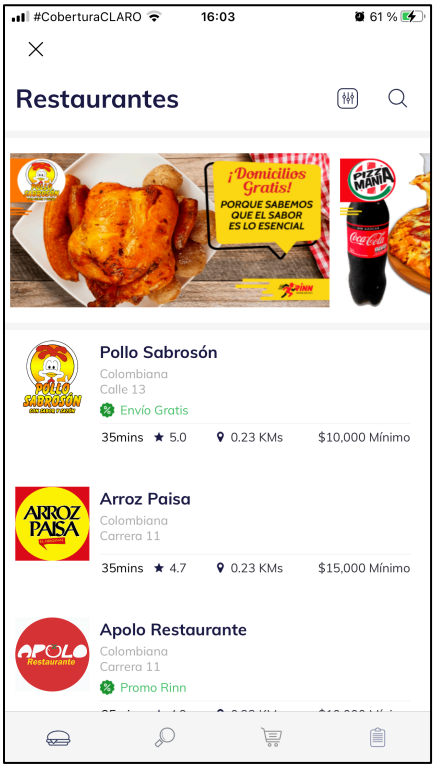


Figura 17. Restaurantes

Ahora, es posible que dentro de una tienda existan categorías de productos y que estas a su vez tengan subcategorías. Un ejemplo de lo anterior, se encuentra en la **Figura 18** donde dentro de la categoría de Ágil Mercado, mas específicamente en la tienda “Cat – Dog (Mascotas)” de la ciudad de Jamundí donde la categoría de productos “Alimentos para perros” contiene una serie de subcategorías como: alimentos húmedos, alimentos secos, etc.

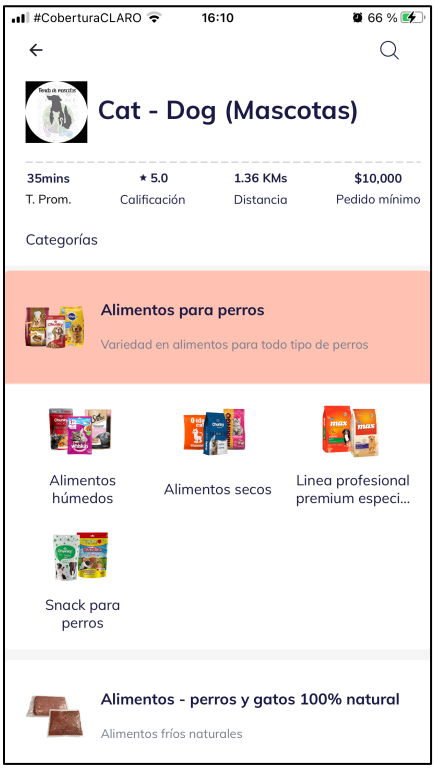


Figura 18. Categorías de productos

Existe un tipo de categoría especial de tiendas que es completamente diferente a las demás y esta se llama “Rinn favor”, en donde los productos son ingresados por el usuario y solamente se cobra el valor de la gestión, el valor de los productos es pagado cuando se realiza la entrega.

3.3.7. GESTIÓN DE DIRECCIONES

Este modulo permite al usuario gestionar sus direcciones, estas pueden ser creadas, editadas y eliminadas. Además, se pueden etiquetar con cualquiera de las siguientes etiquetas: casa, trabajo u otro como se apreciar en la **Figura 19** que muestra la pantalla inicial del modulo donde se listan las direcciones registradas y la opción que permite agregar una nueva.

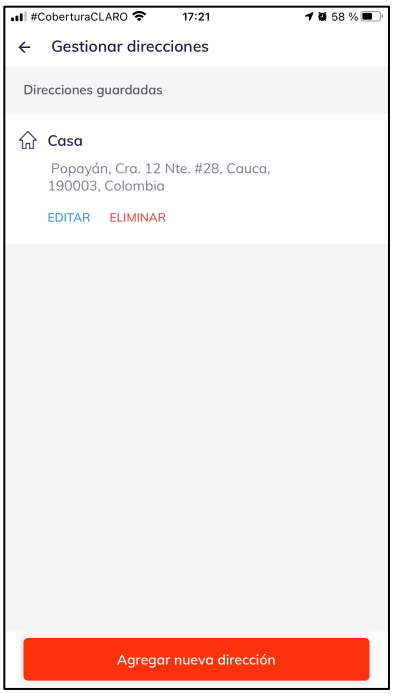


Figura 19. Lista de direcciones

Ahora, si el usuario da clic en el botón “Agregar nueva dirección”, se le redirigirá a la pantalla de registro de direcciones como se puede ver en la **Figura 20**, donde se encuentra el formulario que tiene los atributos: dirección (que puede ser buscada utilizando el teclado o arrastrando el puntero del mapa), piso / casa (opcional) y punto de referencia (también opcional).

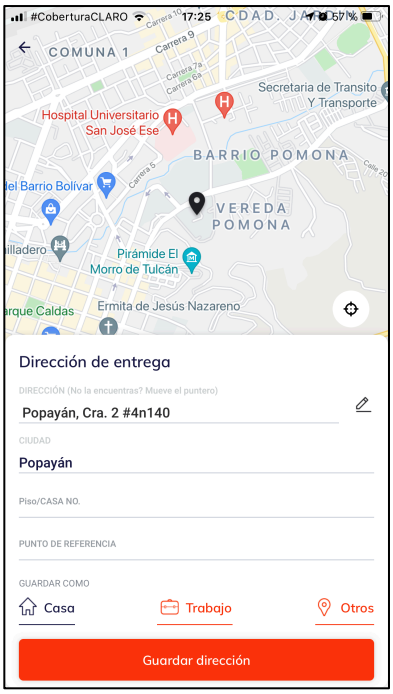


Figura 20. Registro de direcciones

Cabe resaltar que las direcciones que el usuario quiera ingresar deben estar dentro de las áreas en donde Rinn tiene cobertura, hasta el momento estas son: Jamundí y el suroccidente de la ciudad de Cali.

3.3.8. AYUDA Y SOPORTE

Por último, el módulo de ayuda y soporte permite a los usuarios registrados crear tiquetes para generar quejas y/o reportar errores dentro de la plataforma. En la pantalla inicial de este módulo (a la cual se puede acceder mediante la barra de navegación principal) presentada en la **Figura 21** se pueden apreciar los tiquetes generados y una opción en la parte superior derecha para generar uno nuevo.

Figura 21. Ayuda y soporte

Ahora, si un usuario da clic en la opción de generar tiquetes se le redirigirá a la pantalla que contiene esa funcionalidad como se puede ver en la **Figura 22**. Cada tiquete tiene un asunto, una prioridad y un mensaje.

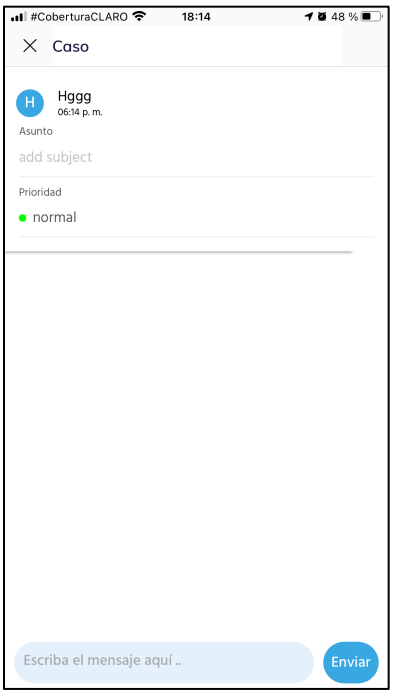


Figura 22. Generar tiquete

Es gracias a este módulo que los clientes de Rinn pueden informar al equipo de ayuda al cliente los problemas que tengan con la aplicación o la plataforma en general, de esta forma las soluciones son mas rápidas y eficientes.

4. CONSTRUCCIÓN DE SOFTWARE

Este capitulo presenta las distintas funcionalidades realizadas sobre la versión de Rinn e iOS a lo largo de toda la practica profesional. En la primera sección se muestra el desarrollo de cada una de las funcionalidades explicando las soluciones propuestas y la implementación de cada una de estas. En la segunda sección de documentan las iteraciones realizadas y sus respectivas actividades junto con un análisis con respecto a su duración, estimación y producto generado.

4.1. DESARROLLO POR ITERACIONES

Para poder desarrollar correctamente todas las funcionalidades dentro de la práctica profesional, en la empresa se utilizó SCRUM como enfoque ágil para poder llevar a cabo los requerimientos asociados a cada funcionalidad, cabe resaltar que el tiempo destinado para el desarrollo de las actividades de cada iteración tenia un valor fijo de ciento sesenta (160) horas, es decir, en caso de que, por ejemplo, existan actividades que no fueron culminadas en su totalidad al cumplir el limite de tiempo, estas pasan a ser parte de la siguiente iteración.

A continuación, se presentan las funcionalidades solicitadas por la empresa junto con los requerimientos asociados a estas, estos se agrupan en componentes lógicos para facilitar el diseño de cada una de las iteraciones o *sprints*.

Tabla 3 - Requerimientos de método de pago PSE

FUNCIONALIDAD 1: METODO DE PAGO PSE	
Componente	**Requerimientos**
Lista de métodos de recarga	Mostrar al usuario los dos métodos de recarga que soporta la plataforma (tarjetas, PSE)
	Al dar clic en un método de recarga, el usuario debe se redirigido al flujo especifico del método.
Formulario de PSE	Implementar el formulario que permite crear una transacción PSE que ya se encuentra desarrollado en la versión de Android.
Web view	Implementar un web view que permite al usuario realizar el pago con la entidad bancaria seleccionada
	Una vez se haya realizado la interacción con la plataforma web de la entidad bancaria, se debe regresar a la billetera.
Billetera	Ya dentro de la billetera, se debe mostrar el balance actualizado.

Tabla 4 - Requerimientos del rediseño del CRUD de tarjetas

FUNCIONALIDAD 2: REDISEÑO DEL CRUD DE TARJETAS	
Componente	**Requerimientos**
Agregar Tarjeta	Diseñar un formulario que contenga los siguientes campos: numero de tarjeta, nombre que aparece en la tarjeta, fecha de expiración y cvv
	Validar que los datos ingresados en el formulario sean correctos
	Presentar la tarjeta e ir llenándola a medida que se va llenando el formulario

	La tarjeta gira con respecto al campo actual que el usuario se encuentra llenando. Ej.: Mostrar el frente de la tarjeta cuando se esta llenando el numero y mostrar la parte de atrás cuando se esta llenando el cvv. También se gira si se da clic sobre ella
	Permitir escanear la tarjeta y llenar los campos automáticamente
	Permitir al usuario ingresar cualquier tipo de tarjeta de crédito.
Listar Tarjetas	Listar las tarjetas que el usuario tiene registradas
	Diseñar e implementar una vista que permita ver las tarjetas del usuario y desde ahí, poder acceder al Agregar tarjeta y, si se da clic sobre una de ellas, mostrar el detalle de la misma
	La tarjeta por defecto debe tener un indicador.
Detalle de tarjeta	Mostrar los últimos 4 dígitos de la tarjeta junto con la franquicia de la misma
	En caso de ser la tarjeta por defecto, mostrar el indicador utilizado en la lista
	Permitir al usuario establecer por defecto y eliminar la tarjeta

Tabla 5 - Requerimientos del rediseño de la billetera

FUNCIONALIDAD 3: REDISEÑO DE LA BILLETERA	
Componente	**Requerimientos**
Pantalla principal	Rediseñar la pantalla principal de la billetera.
	Eliminar el limite flexible y el limite máximo
	Mostrar los últimos 5 movimientos de la billetera
	Darle mas relevancia a la billetera como tal que a las acciones que se pueden hacer con ella (recargar, ver el historial)
Historial de movimientos	Mostrar el historial de los movimientos del usuario
	Realizar paginación mediante *scroll*
	Al dar clic en un movimiento, se debe mostrar sus detalles
Detalles de un movimiento	Mostrar el valor, el motivo, el identificador, la referencia, el tipo de transacción, la descripción y la fecha de un movimiento
	El valor debe tener una mayor relevancia que el resto de los datos
Recargas	Actualizar el estilo grafico de la vista de métodos de recarga
	Obtener los métodos de recarga mediante un servicio del *backend*
	Tomar en cuenta que un método de recarga puede tener una categoría y subcategoría.
	Rediseñar el flujo de una recarga para que sea genérico (sin importar el método de recarga seleccionado)
	Adicionar una pantalla que muestra el resultado de una recarga. Debe contener el valor recarga, el método seleccionado, la fecha, el identificador de la transacción, el estado de la transacción y el nuevo valor de la billetera.
	Adicionar una pantalla de carga que se muestra mientras la transacción esta siendo realizada
Recarga PSE	Rediseñar el formulario de creación de la transacción
	Adaptar el flujo de PSE con el flujo de recarga
	Encriptar los datos ingresados por el usuario
Recarga con tarjeta	Rediseñar el flujo de recarga con tarjeta y adaptarlo al flujo de recarga
	Al seleccionar el método de pago con tarjeta, el usuario es redirigido a lista de tarjetas donde podrá seleccionar una (la tarjeta por defecto es seleccionada automáticamente)

Tabla 6 - Requerimientos del modulo de pagos y recargas

FUNCIONALIDAD 4: MÓDULO DE PAGOS Y RECARGAS	
Componente	**Requerimientos**

Billetera	Agregar los productos que se pueden compran con la billetera en la pantalla principal de la misma.
	Si se da clic sobre uno de los productos, se debe redirigir al flujo especifico de cada uno
Pantalla principal	En la pantalla principal de la aplicación se debe mostrar, como si fuese una categoría de tiendas, los productos de la billetera (pagos y recargas)
	Si de da clic sobre esta categoría, se debe mostrar, en otra pantalla, los productos disponibles.
SOAT	Diseñar e implementar un formulario donde se le pida al usuario ingresar la placa del vehículo, el tipo de identificación del propietario, el numero de identificación y aceptar los términos y condiciones del servicio
	Si se da clic sobre los términos y condiciones, se deben cargan en un web view
	Una vez ingresados los datos, se obtiene la información del vehículo
	Si el vehículo tiene varios tipos de póliza, el usuario debe poder seleccionar uno (en otra pantalla). En caso de que solo tenga uno, se dirige al usuario directamente a la información del vehículo
	Se debe mostrar la información del vehículo junto con un formulario que le usuario debe llenar con los datos del propietario
	Una vez finalizado, se debe dirigir a la confirmación.
Recarga celular	Diseñar e implementar una pantalla que le permita al usuario seleccionar el operador de su celular
	Diseñar e implementar una pantalla que le permita al usuario ingresar su numero de celular
	Una vez finalizado, se debe dirigir a la confirmación.
Servicios públicos	Diseñar e implementar una pantalla que permita buscar las empresas proveedoras.
	Diseñar e implementar una pantalla que permita ingresar el numero de referencia de la factura.
	Una vez finalizado, se debe dirigir a la confirmación.

Tabla 7 - Requerimientos de la validación de tarjetas

FUNCIONALIDAD 5: VALIDACIÓN DE TARJETA	
Componente	**Requerimientos**
Agregar tarjeta	En caso de que la tarjeta registrada no se encuentre validada, se le debe pedir al usuario que realice la validación. Esta validación consiste en pedirle al usuario que ingrese el valor aleatorio que fue cobrado a su tarjeta (por defecto el usuario tendrá hasta 3 intentos).
Detalles de tarjeta	Si una tarjeta no se encuentra validada, se debe mostrar un botón dentro del detalle que permita acceder a la validación

Tabla 8 - Requerimientos del método de pago Nequi

FUNCIONALIDAD 6: METODO DE PAGO NEQUI	
Componente	**Requerimientos**
Perfil	Se deben obtener los métodos de pago desde un servicio del backend
Agregar Nequi	Diseñar e implementar un formulario donde se le pida al usuario ingresar el numero de celular, email, nombre, tipo de documento y numero de documento para agregar un Nequi. Aclarar que se debe aceptar en la app de Nequi
Listar Nequi	Al igual que las tarjetas, el usuario podrá listar sus Nequi
	Si se encuentra dentro de esta vista y la app vuelve del background, se debe actualizar la lista de Nequi

Detalles Nequi	El usuario podrá ver los detalles de su Nequi (numero de celular). Desde ahí podrá eliminarlo. En caso de que no haya sido validado, se le debe indicar al usuario que debe hacerlo desde la app de Nequi
Recarga con Nequi	El flujo debe ser el mismo que con tarjetas (ingresar a la lista de Nequi, seleccionar uno, ingresar el valor a recargar)
Checkout	El usuario podrá seleccionar cualquiera de sus tarjetas o Nequi validados para realizar pedidos. También puede adicionar un Nequi desde esta pantalla

Para desarrollar los requerimientos que la empresa solicita utilizando SCRUM, se dividen los requerimientos en actividades y cada sprint o iteración tiene una duración de 1 mes, donde cada actividad dentro de cada sprint tiene una prioridad (baja, media o alta, según lo indique la empresa) y dos duraciones: una prevista y otra real (ambas en horas). A continuación, se documentan las iteraciones realizadas a lo largo de la practica profesional basándose en un proceso enmarcado en 3 pasos: agrupar las actividades de manera lógica, analizar el sprint con respecto a las estimaciones y prioridades, y mostrar el producto generado. Cabe resaltar que, para el caso de las estimaciones, en la empresa se tiene las siguientes rúbricas:

- Una estimación es buena si existe un desfase con respecto a la duración real menor o igual al 10%
- Una estimación es mala si existe un desfase con respecto a la duración real mayor al 10%

4.1.1. PRIMERA ITERACIÓN

En la primera iteración, las actividades realizadas tenían como énfasis una de las funcionalidades principales (método de pago PSE), la modificación de la lógica del módulo de gestión de las direcciones y solución de bugs junto con funcionalidades menores. Estas actividades fueron clasificadas de la siguiente manera:

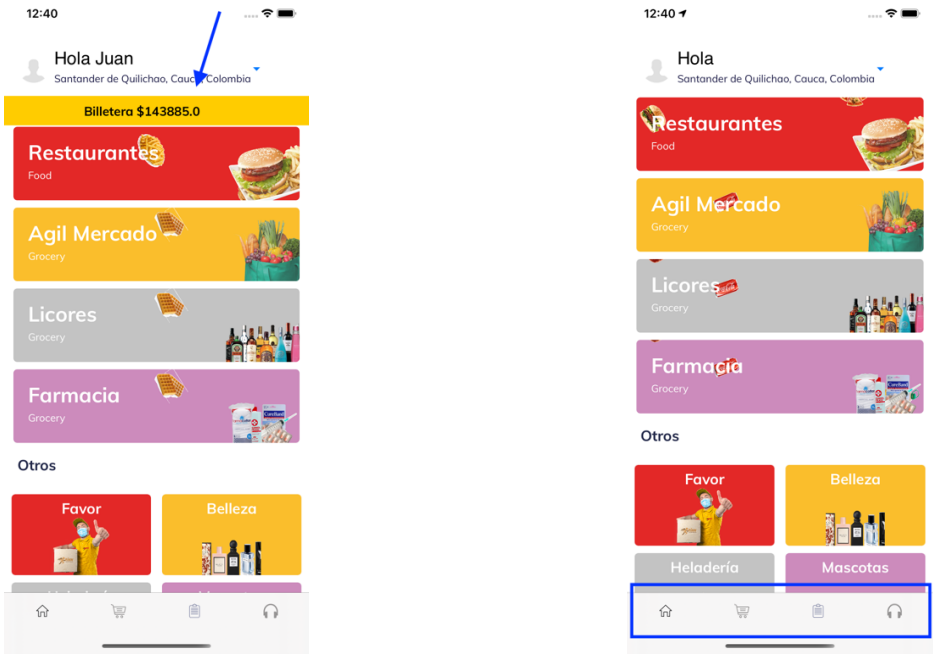
Tabla 9 - Primera iteración: actividades referentes a solución de bugs y funcionalidades pequeñas

Actividad	Prioridad	Duración Prevista	Duración Real
Agregar un banner en la pantalla principal que muestre el balance actual de la billetera.	Media	8	12
Agregar las traducciones faltantes.	Baja	5	8
Agregar una barra de navegación en la pantalla principal que permite navegar entre las vistas: home, carrito, historial y ayuda.	Media	4	5
Solucionar bug que bloqueaba la aplicación al realizar un pedido mediante tarjeta de crédito / débito.	Alta	2	2
Modificar la lógica del formulario de registro para que el usuario no pueda ingresar un código de referido y un cupón.	Alta	4	4
Modificar la posición del botón principal del formulario de registro.	Baja	2	2
Agregar un pop-up en la pantalla principal que indique al usuario que este se ha alejado de la ubicación seleccionada.	Media	6	8

Solucionar bug que no permitía modificar al usuario el código OTP al momento de registrarse.	Media	5	6

El producto que generaron el desarrollo de las actividades presentadas en la **Tabla 9** se presentan a continuación:

- Un pequeño banner que muestra el balance actual de la billetera móvil y que, al dar clic en él, se redirecciona al usuario a la pantalla de la billetera. Se puede apreciar en la **Figura 23**.
- Se agregó una barra de navegación a la pantalla principal. Permite navegar entre las siguientes pantallas: home, carrito, historial de pedidos y ayuda. Se puede apreciar en la **Figura 23**.



Nuevo banner de billetera en la pantalla principal ***Nueva barra de navegación de la pantalla principal***

Figura 23. Producto generado de pequeñas funcionalidades en el primer sprint

Tabla 10 - Primera iteración: actividades referentes al cambio de lógica de la gestión de las direcciones

Actividad	Prioridad	Duración Prevista	Duración Real
Solucionar bug que permitía al usuario ingresar una dirección inexistente.	Media	6	8
Agregar funcionalidad que permite ingresar al modo edición de la dirección.	Alta	16	24
Solucionar problema en que, en algunos casos, al seleccionar una ubicación, la etiqueta que se colocaba en el campo no era la misma que se seleccionaba.	Alta	6	8
Agregar el campo “Ciudad” en el formulario.	Baja	2	2

Las funcionalidades realizadas dentro de la gestión de direcciones se describen a continuación:

- Se agregó el modo edición que permite al usuario modificar los últimos 3 caracteres de su dirección.
- Se agregó el campo “ciudad”, el cual es no editable y se actualiza cuando el usuario busca una nueva dirección.

Nuevo modo edición en la gestión de direcciones

Nuevo campo de Ciudad en la gestión de direcciones

Figura 24. Producto generado gestión de direcciones primer sprint

Tabla 11 - Primera iteración: actividades referentes al método de pago PSE

Actividad	Prioridad	Duración Prevista	Duración Real
Diseñar el diagrama de clases	Media	6	8
Diseñar el diagrama de secuencia	Media	4	6
Cambiar el *layout* de la billetera.	Media	5	7
Lectura de la documentación de ePayco.	Alta	8	10
Encriptación de los datos	Alta	8	16
Diseñar el formulario de pago.	Media	4	5
Implementar el formulario de pago.	Media	5	5

El producto generado por la realización de estas actividades se encuentra dentro de la sección “FUNCIONALIDAD 1: METODO DE PAGO PSE”.

Ahora bien, al analizar el desarrollo del sprint, se puede apreciar que se realizaron 19 actividades cuya duración total (real), es decir, el tiempo (en horas) que tomó realizar todas estas fue de 160 horas y el promedio de duración de cada una de ellas fue de 8,42 horas. En cuanto a la estimación realizada, se preveía finalizar con todas las actividades en un plazo de 106 horas, por lo tanto, no se realizó una buena estimación al inicio del sprint en tanto que, con respecto a la duración real, hubo un desfase de 54 horas, es

decir, un 33% más de lo que se esperaba. Además, el 16% de las actividades fueron de prioridad baja, 50% de prioridad media y 33% de prioridad alta.

4.1.2. SEGUNDA ITERACIÓN

En la segunda iteración, las actividades realizadas tenían como énfasis dos de las funcionalidades principales (solución de bugs en la implementación del método de pago PSE y el análisis del nuevo módulo de pagos y recargas). También, solucionar los bugs encontrados en la aplicación y la solución de los *crashes* más importantes del aplicativo. Estas actividades fueron clasificadas de la siguiente manera:

Tabla 12 - Segunda iteración: actividades referentes a la solución de bugs del aplicativo

Actividad	Prioridad	Duración Prevista	Duración Real
Al recargar la billetera, el mensaje que indica el éxito o fracaso de la transacción tarda en aparecer.	Media	10	12
Mostrar mensaje "saldo insuficiente" cuando la billetera no tiene saldo suficiente para realizar un pedido.	Baja	3	3
No se muestra el método de pago en el detalle de una orden	Media	5	5
Al agregar una tarjeta, el mensaje de respuesta tarda en aparecer.	Alta	5	6
Las direcciones en el Rinn Favor se borran al ir al *checkout*.	Alta	5	5
Solución de bug en validación de formulario de registro	Baja	2	2
Solución de bug en vista de confirmación de pedido	Media	3	4

Para las actividades referentes a la solución de bugs, cabe aclarar que, para la empresa es de suma importancia que no se divulgue a detalle los bugs del aplicativo ni su solución, por lo tanto, solo se mencionan cuales fueron encontrados y solucionados.

Tabla 13 - Segunda iteración: actividades referentes a la solución de crashes

Actividad	Prioridad	Duración Prevista	Duración Real
Solución de *crash* al iniciar la app cuando hay una nueva versión disponible	Media	8	7
Solución de *crash* al cargar los gifs de la pantalla inicial	Media	24	32

Gracias a las actividades realizadas dentro de la solución de *crashes* del aplicativo, la cantidad de *crashes* reportados se redujeron, en promedio semanal, de más de 100 a al rededor 20, es decir, una reducción de más del 80%.

Tabla 14 - Segunda iteración: actividades referentes a la solución de bug en el método de pago PSE

Actividad	Prioridad	Duración Prevista	Duración Real
Cambio en lógica de PSE	Media	4	4
Bloqueo en alertas de PSE	Alta	3	4

Se regresaba a la pantalla principal al ingresar a la pagina del banco sin consentimiento del usuario	Alta	4	8
Solución de bug bloqueante en el navegador de PSE	Alta	4	8
Al pagar con PSE, la pantalla se queda bloqueada	Alta	4	2

Al igual que las actividades referentes a la solución de bugs, la empresa deja en claro que no pueden detallar, solo nombrarlos.

Tabla 15 - Segunda iteración: actividades referentes al análisis del nuevo módulo de pagos y recargas

Actividad	Prioridad	Duración Prevista	Duración Real
Análisis de los módulos implementados por la competencia	Media	24	24
Diseño de prototipos para el pago de recarga de celulares	Media	16	16
Rediseño de la pantalla principal de la billetera	Media	8	8
Análisis de los elementos gráficos de la pantalla principal de la billetera		4	4
Diseño e implementación de la utilidad que permite validar un formulario	Media	8	8

A medida que se fue caracterizando el aplicativo, se encontró una falencia en la validación de los formularios ya que no existe validador global que sea independiente del formulario que lo utilice y por tanto reutilizable. Es por esto por lo que se toma la decisión de diseñar e implementar una utilidad que pueda ser utilizada por cualquier formulario sin importar el numero de campos que se requieran ni la lógica propia de la pantalla en la que se encuentre. La solución propuesta se puede apreciar en la **Figura 25**, donde se encuentra su diagrama de clases.

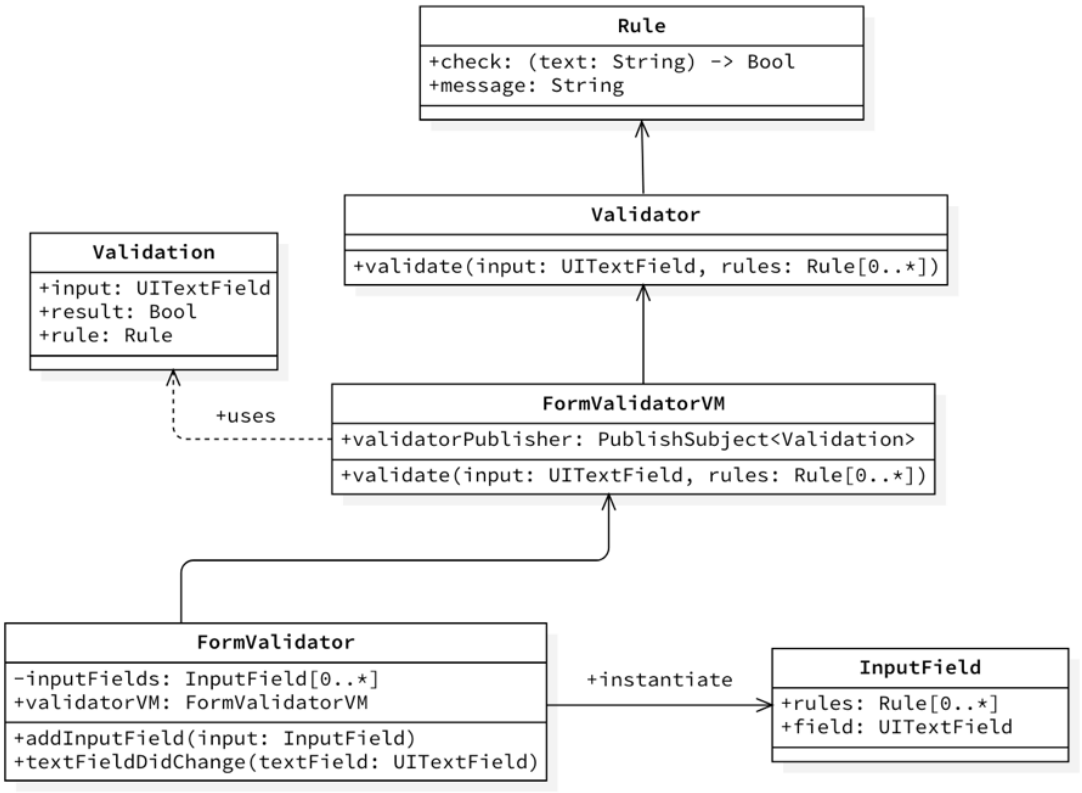


Figura 25. Diagrama de clases de la utilidad que permite validar un formulario

Este diagrama está compuesto por las siguientes clases:

Rule: Hace referencia a una regla que debe cumplir un campo (por ejemplo, que tenga una cantidad mínima de caracteres). Cada regla tiene como atributo una función que recibe el contenido del campo de texto y que se encarga de validar el cumplimiento de la regla. Además, tiene un mensaje que se le mostrará al usuario cuando no se cumple la validación.

Validator: Encargada de realizar la validación de cada regla mediante la función *validate.*

Validation: Representa el resultado de una validación. Contiene el campo de texto que se validó, el resultado de la validación y, en caso de que este sea *false*, la regla que no se cumplió.

FormValidatorVM: Se encarga de llamar al *Validator* a través de la función *validate*. Una vez realizada la validación, envía el resultado a través del atributo *validatorPublisher*.

InputField: Hace referencia a un campo de texto de un formulario. Contiene la referencia del campo y las reglas que este debe cumplir.

FormValidator: Es la puerta de entrada de la utilidad. Contiene los *inputFields* del formulario y observa el view model. Además, tiene dos métodos: *addInputField* que permite agregar un campo de texto al validador y *textFieldDidChange* que se ejecuta en el momento en que un campo de texto ha sido modificado.

El producto generado por las demás actividades referentes al análisis del nuevo módulo de pagos y recargas se encuentra dentro de la sección “FUNCIONALIDAD 4: MÓDULO DE PAGOS Y RECARGAS”.

Al analizar el desarrollo del sprint, se puede apreciar que se realizaron 19 actividades, al igual que el primer sprint, cuya duración total (real), es decir, el tiempo (en horas) que tomó realizar todas estas fue de 160 horas y el promedio de duración de cada una de ellas fue de 8,42 horas. En cuanto a la estimación realizada, se preveía finalizar con todas las actividades en un plazo de 144 horas, por lo tanto, no se realizó una buena estimación al inicio del sprint en tanto que, con respecto a la duración real, hubo un desfase de 22 horas, es decir, un 15,2% más de lo que se esperaba, pero, con respecto al primer sprint, ese desfase es menor. Además, el 5% de las actividades fueron de prioridad baja, 55% de prioridad media y 40% de prioridad alta.

4.1.3. TERCERA ITERACIÓN

En la tercera iteración, las actividades realizadas tenían como énfasis la siguiente funcionalidad principal: el análisis del nuevo modulo de pagos y recargas.

Tabla 16. Tercera iteración: actividades referentes al análisis del nuevo módulo de pagos y recargas

Actividad	Prioridad	Duración Prevista	Duración Real
Diseño del flujo del nuevo módulo	Alta	8	8
Diseño del flujo de pago de recarga celular	Media	4	5
Diseño de prototipos para el pago de recarga celular	Media	8	8
Implementación de prototipos para el pago de recarga celular	Media	8	8
Diseño del flujo de pago de servicios públicos	Media	4	5
Diseño de prototipos para el pago de servicios públicos	Media	8	8
Diseño e implementación de prototipos para el pago de servicios públicos	Media	8	16
Diseño de flujo de pagos de pines de entretenimientos	Media	4	5
Diseño de prototipos para el pago de pines de entretenimiento	Media	8	8
Diseño e implementación de prototipos para el pago de pines de entretenimiento	Media	16	16
Diseño de flujo de pago de SOAT	Media	4	5
Diseño de prototipos para el pago de SOAT	Media	12	12
Diseño e implementación de prototipos para el pago de SOAT	Media	16	16
Análisis de la documentación entregada por el proveedor del servicio	Alta	16	16
Diseño del modelo de datos del nuevo modulo	Alta	24	24

En este sprint no se desarrollaron funcionalidades especificas que no se encontraran dentro de los objetivos de la practica profesional. El producto generado se encuentra dentro de la sección “FUNCIONALIDAD 4: MÓDULO DE PAGOS Y RECARGAS”.

Una vez analizado el desarrollo del sprint, se puede apreciar que únicamente se realizaron 15 actividades, cuya duración total (real), es decir, el tiempo (en horas) que tomó realizar todas estas fue de 160 horas y el promedio de duración de cada una de ellas fue de 10,66 horas. En cuanto a la estimación realizada, se preveía finalizar con todas las actividades en un plazo de 148 horas, por lo tanto, se realizó una buena estimación al inicio del sprint en tanto que, con respecto a la duración real, hubo un desfase de únicamente 8 horas, es decir, un 8,1% más de lo que se esperaba. Además, el 0% de las actividades fueron de prioridad baja, 80% de prioridad media y 20% de prioridad alta.

4.1.4. CUARTA ITERACIÓN

En la cuarta iteración, las actividades realizadas tenían como énfasis una funcionalidad principal: implementación del nuevo diseño de la billetera. También, como funcionalidades extra: la encriptación de datos de tarjeta e integración con el SDK de Facebook. Estas actividades fueron clasificadas de la siguiente manera:

Tabla 17. Cuarta iteración: actividades referentes al rediseño de la billetera

Actividad	Prioridad	Duración Prevista	Duración Real
Diseño de nuevo flujo de recarga	Media	8	8
Análisis de los elementos gráficos de pantalla que lista las tarjetas crédito/debito	Media	4	5
Diseño de la pantalla que lista las tarjetas crédito/debito	Media	12	12
Implementar nuevo diseño de la pantalla que lista las tarjetas crédito/debito	Media	24	24
Ultimar detalles del nuevo diseño de la billetera	Media	2	2
Análisis de los elementos gráficos de la pantalla de adición de tarjetas crédito/debito.	Media	4	5
Diseño de la pantalla de adición de tarjetas crédito/debito.	Media	8	8
Implementar nuevo diseño de la pantalla de adición de tarjetas crédito/debito.	Media	16	16
Análisis de los elementos gráficos de la pantalla del detalle de tarjeta	Media	4	5
Diseño de la pantalla del detalle de tarjeta	Media	8	8
Implementar nuevo diseño de la pantalla del detalle de tarjeta	Media	8	8

El producto generado por la realización de las actividades referentes al rediseño de la billetera se encuentran dentro de la sección: “FUNCIONALIDAD 3: REDISEÑO DE BILLETERA”.

Tabla 18 - Cuarta iteración: actividades referentes a funcionalidades extra

Actividad	Prioridad	Duración Prevista	Duración Real
Encriptación de los datos de las tarjetas de los usuarios	Alta	14	16
Análisis de la documentación de Facebook	Media	4	4
Integración con el SDK de Facebook	Media	8	12
Realizar pruebas funcionales en adición de tarjeta	Alta	8	8
Realizar pruebas funcionales en recarga con tarjeta	Alta	8	8
Realizar pruebas funcionales en recarga con PSE	Alta	8	8
Cambiar validación de formulario en el Rinn favor.	Media	4	4

La encriptación de los datos de las tarjetas, al ser información sensible, por petición de la empresa, no puede ser detalla dentro del presente informe. Al analizar el desarrollo del sprint, se puede apreciar que se realizaron 18 actividades, cuya duración total (real), es decir, el tiempo (en horas) que tomó realizar todas estas fue de 160 horas y el promedio de duración de cada una de ellas fue de 8,88 horas. En cuanto a la estimación realizada, se preveía finalizar con todas las actividades en un plazo de 152 horas, por lo tanto, se realizó una buena estimación al inicio del sprint en tanto que, con respecto a la duración real, hubo un desfase de únicamente 8 horas, es decir, un 5% más de lo que se esperaba. Además, el 0% de las actividades fueron de prioridad baja, 83,33% de prioridad media y 16,66% de prioridad alta.

4.1.5. QUINTA ITERACIÓN

En la quinta iteración, las actividades realizadas tenían como énfasis las siguientes funcionalidades principales: rediseño de la billetera y la implementación del nuevo módulo de pagos y recargas. Estas fueron clasificadas de la siguientes manera:

Tabla 19. Quinta iteración: actividades referentes al rediseño de la billetera

Actividad	Prioridad	Duración Prevista	Duración Real
Agregar vista de resultado de recarga	Media	4	4
Conectar la vista de resultado de recarga con billetera	Media	2	2
Conectar la vista de resultado de recarga con el *checkout* de pagos	Media	3	5
Conectar la vista de resulta de recarga con PSE	Media	2	2
Agregar vista de carga en recarga con PSE	Media	2	2
Agregar vista de carga en recarga con tarjeta	Media	2	2
Pruebas de recarga con tarjeta	Media	5	5
Pruebas de recarga con PSE	Media	5	5

El producto generado por la realización de las actividades referentes al rediseño de la billetera se encuentran dentro de la sección: “FUNCIONALIDAD 3: REDISEÑO DE BILLETERA”.

Tabla 20 - Quinta iteración: actividades referentes al nuevo módulo de pagos y recargas

Actividad	Prioridad	Duración Prevista	Duración Real
Organizar los métodos de recarga por categorías	Media	8	8
Diseño de flujo de pago tipo SOAT	Media	4	5
Diseño de pago tipo SOAT	Media	6	6
Implementación de pago tipo SOAT	Media	16	24
Implementación de pago tipo recarga celular	Media	24	24
Implementación de pago tipo servicio publico	Media	24	20
Diseño de pantalla de menú de pagos y recargas	Media	8	8
Implementación de pantalla de menú de pagos y recargas	Media	16	16
Funcionalidad que permite buscar un servicio publico	Media	8	16
Ultimar detalles de diseño de SOAT	Media	8	8

El producto generado por la realización de las actividades referentes al módulo de pagos y recargas se encuentran dentro de la sección: “FUNCIONALIDAD 4: MÓDULO DE PAGOS Y RECARGAS”.

Al analizar el desarrollo del sprint, se puede apreciar que se realizaron 17 actividades, cuya duración total (real), es decir, el tiempo (en horas) que tomó realizar todas estas fue de 160 horas y el promedio de duración de cada una de ellas fue de 12 horas. En cuanto a la estimación realizada, se preveía finalizar con todas las actividades en un plazo de 147 horas, por lo tanto, se realizó una buena estimación al inicio del sprint en tanto que, con respecto a la duración real, hubo un desfase de 13 horas, es decir, un 8% más de lo que se esperaba. Vale resaltar que, en este sprint, todas las actividades tuvieron una prioridad media.

4.1.6. SEXTA ITERACIÓN

En la sexta y ultima iteración, las actividades realizadas tenían como énfasis las siguientes funcionalidades principales: validación de tarjetas y método de pago Nequi. Estas fueron clasificadas de la siguientes manera:

Tabla 21. Sexta iteración: actividades referencias a la validación de tarjetas

Actividad	Prioridad	Duración Prevista	Duración Real
Analizar y diseñar pantalla de validación de tarjeta	Media	12	12
Implementación de pantalla de validación de tarjeta	Media	16	16

El productos generado por las actividades referentes a la validación de tarjetas se encuentra dentro de la sección: “FUNCIONALIDAD 5: VALIDACIÓN DE TARJETA”.

Tabla 22. Sexta iteración: actividades referentes al método de pago Nequi

Actividad	Prioridad	Duración Prevista	Duración Real
Diseñar pantalla de agregar Nequi	Media	8	8
Implementar pantalla de agregar Nequi	Media	16	16

Conectar pantalla de agregar Nequi con el *backend*	Alta	2	2
Diseñar pantalla de listar Nequi asociados	Media	6	8
Implementar pantalla de listar Nequi asociados	Media	16	16
Implementar eliminar Nequi	Media	8	8
Diseño de recarga de billetera con Nequi	Media	6	6
Implementación de recarga de billetera con Nequi	Media	10	10
Realizar pruebas en agregar Nequi	Alta	8	8
Realizar pruebas en eliminar Nequi	Alta	8	8
Realizar pruebas en listar Nequi	Alta	8	8
Realizar pruebas en la validación de tarjeta	Alta	8	8
Relizar *deploy* a *testflight*	Alta	4	6
Subir la app a la tienda de aplicaciones	Alta	14	16

El producto generado por las actividades referentes al método de pago Nequi se encuentra dentro de la sección: “FUNCIONALIDAD 6: METODO DE PAGO NEQUI”.

Si se analiza el desarrollo del sprint, se puede apreciar que se realizaron 14 actividades, cuya duración total (real), es decir, el tiempo (en horas) que tomó realizar todas estas fue de 160 horas y el promedio de duración de cada una de ellas fue de 11,42 horas. En cuanto a la estimación realizada, se preveía finalizar con todas las actividades en un plazo de 150 horas, por lo tanto, se realizó una buena estimación al inicio del sprint en tanto que, con respecto a la duración real, hubo un desfase de 10 horas, es decir, un 7% más de lo que se esperaba. Además, el 0% de las actividades fueron de prioridad baja, 50% de prioridad media y 50% de prioridad alta.

Una vez finalizados los *sprints*, se procede realizar un análisis general de todas las iteraciones realizadas durante el desarrollo de la practica profesional. La duración de todos los *sprints* varía según la funcionalidad que se realiza dentro del mismo, esto se presenta en la **Figura 26**, donde se presenta las distintas estimaciones y duraciones reales de cada sprint.

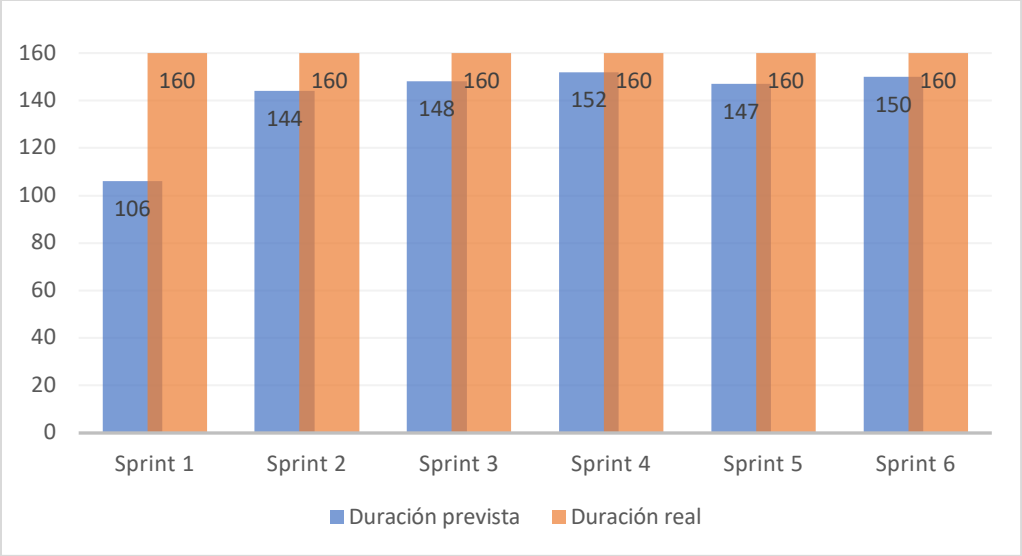


Figura 26. Duración estimada vs prevista

Al analizar la estimaciones realizadas en cada sprint y observar la **Figura 27**, donde se presenta la variación de los desfases de estimación de cada sprint (es decir, qué tan alejada está la duración prevista con respecto a la duración real de las actividades).

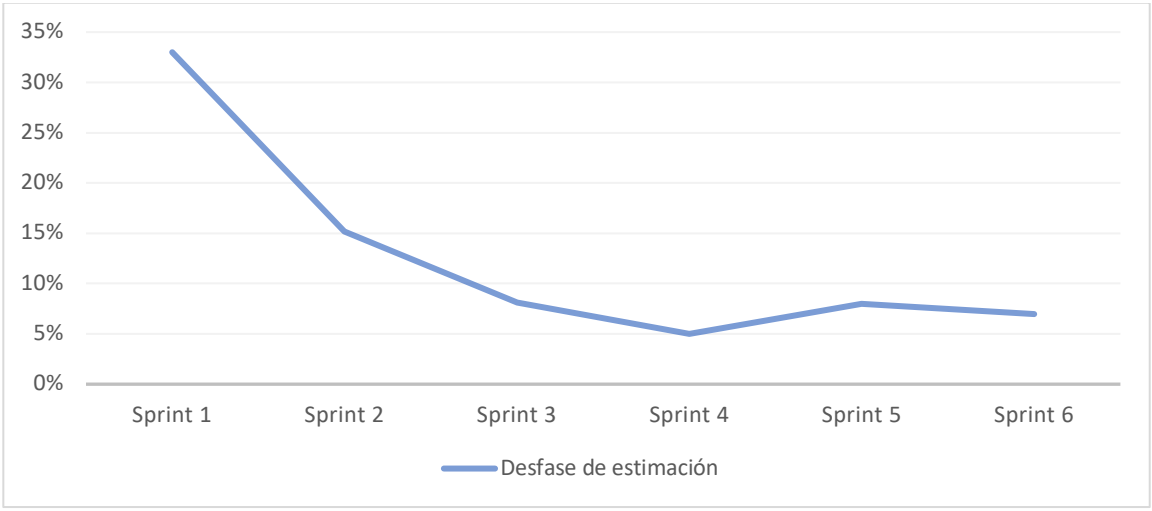


Figura 27. Desfase porcentual de estimación

Se puede apreciar que a medida que se iba iterando, la estimación era cada vez mejor, en tanto que el desfase presentado era cada vez menor, esto se debió a que, a medida que se iba realizando la practica profesional y se iba caracterizando el aplicativo, las estimaciones eran mucho más exactas ya que el conocimiento de las capacidades técnicas del equipo era mayor después de cada iteración.

4.2. DESARROLLO DE FUNCIONALIDADES

4.2.1. FUNCIONALIDAD 1: METODO DE PAGO PSE

Cuando se habla del nuevo método de pago PSE, se hace referencia a una nueva forma en la que el usuario puede recargar su billetera móvil. Es decir, ahora el usuario, además de poder agregar dinero mediante tarjetas de crédito / debito, podrá hacerlo mediante PSE. Para esto, se debe rediseñar el flujo de recarga de billetera. Lo primero a tomar en cuenta es la división de la pantalla de la billetera (**Figura 9**) en dos: una para mostrar la información general de la billetera (**Figura 28**) y otra para realizar la recarga mediante tarjetas (**Figura 29**).

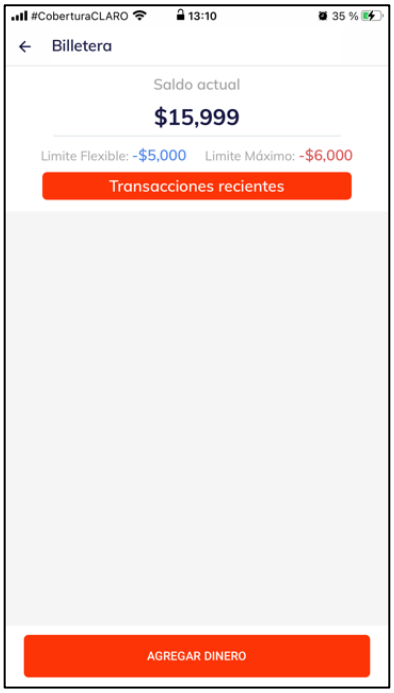


Figura 28. Refactorización billetera

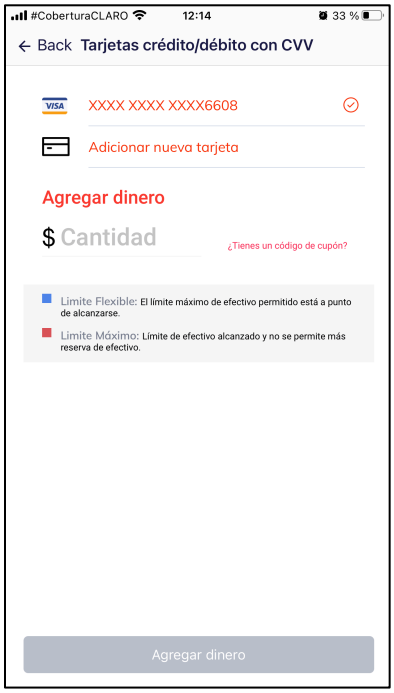


Figura 29. Refactorización de recarga con tarjeta

Ahora, para conectar estas dos pantallas, se crea una nueva que le permite al usuario seleccionar el método de pago que usará para realizar la recarga (**Figura 30**), en donde se encuentran las dos opciones soportadas: tarjetas de crédito/débito y PSE.

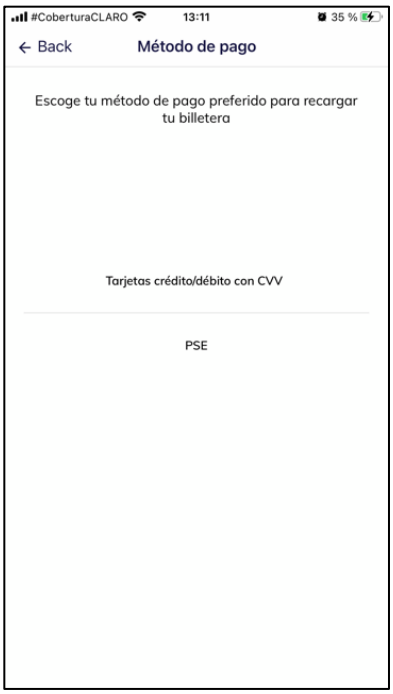


Figura 30. Métodos de recarga

Esto quiere decir que para que un usuario pueda recargar su billetera, deberá ingresar a su billetera móvil (**Figura 9**), dar clic en “Agregar dinero” para ser redirigido a los métodos de recarga (**Figura 30**) y, al seleccionar el método requerido, se le redirigirá al flujo especifico del método (en el caso de tarjeta, a la pantalla de recarga con tarjeta que se presenta en la **Figura 29**) y al final volverá a la billetera donde podrá ver su nuevo saldo. Este rediseño del flujo se puede apreciar en el diagrama presentado en la **Figura 31**, con la precisión de que se pone el flujo de recarga con PSE más visible en tanto que es una de las nuevas funcionalidades.

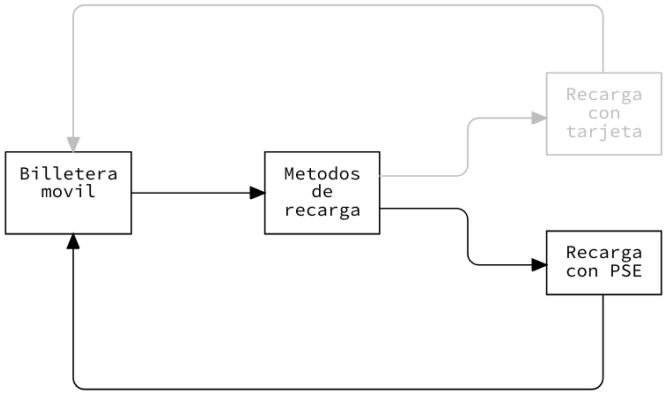


Figura 31. Flujo de recarga

Además, gracias a este rediseño, se pueden agregar *n* métodos de recarga sin tener que modificar la lógica de la billetera. Una vez implementado este nuevo flujo, se procede a diseñar la solución inicial al problema planteado por la empresa.

4.2.1.1. Solución Inicial Propuesta

Al revisar la documentación oficial de ePayco se encuentra que para iOS no existe un SDK (como sí lo hay para *Android, NodeJs*, etc.) que permite consumir los servicios de la pasarela, por lo tanto, el consumo se debe realizar de manera manual. Ahora, para crear una transacción con PSE, se deben tomar en cuenta lo siguiente:

- Antes de realizar cada transacción, se debe realizar una autenticación con ePayco.
- Los datos que se deben enviar a ePayco para crear una transacción siempre deben ir encriptados.
- Al crear una transacción, se obtiene un enlace (que enlaza a la entidad bancaria) a donde se debe redirigir al usuario para que realice el pago.
- ePayco cuenta con dos servidores, uno que provee servicios de autenticación y el otro que permite realizar transacciones.

Con lo anterior, se puede apreciar, desde el punto de vista del usuario final, que, para realizar una transacción con PSE, solo se requieren dos pantallas: la primera para que el usuario ingrese los datos que ePayco necesita (formulario inicial) y la segunda para que el usuario ingrese a la plataforma web de su entidad bancaria (*web view*).

Según lo anterior, se propone esta serie de pasos como primera solución al problema planteado:

1. Al ingresar al formulario inicial, se debe obtener la configuración de PSE (entidades bancarias, valor mínimo y máximo), esto para evitar que el usuario ingrese un valor erróneo, y para que el tope máximo y mínimo sea independiente a la aplicación. Esta configuración se obtiene del servidor de Rinn.
2. Una vez se han llenado el formulario y el usuario ha indicado que quiere iniciar el proceso con su banco:
    - Autenticar la transacción con ePayco. Esta autenticación se realiza con la llave publica y privada del comercio.
    - Una vez se haya autenticado, se procede a obtener el objeto que genera una nueva transacción PSE.
3. Registrar la transacción en los servidores de Rinn. Puesto que, hasta ese momento, Rinn desconoce dicha transacción.
4. Una vez finalizada la interacción con la web del banco, obtener la información de la transacción y almacenarla localmente.
5. Confirmar la transacción en los servidores de Rinn para completar el pago y agregar el valor a la billetera del usuario. Hace referencia a la validación necesaria que se realiza para validar el estado de la transferencia, en caso de que sea aprobada o pagada, se procede a realizar la recarga a la billetera.

Lo anterior, se puede apreciar de una manera más resumida en el diagrama de secuencia presentado en la **Figura 32**, donde *App* equivale al aplicativo móvil, *RinnServer* equivale

al servidor de Rinn, *EpaycoAuthServer* equivale al servidor de autenticación de ePayco, *EpaycoTransactionServer* equivale al servidor de transacciones de ePayco, getGatewayConfig equivale al paso 1, *authenticate* equivale al primer inciso del paso 2, *createTransaction* equivale al segundo inciso del paso 2, *saveTransaction* equivale al paso 3, *getTransactionInfo* equivale al paso 4.

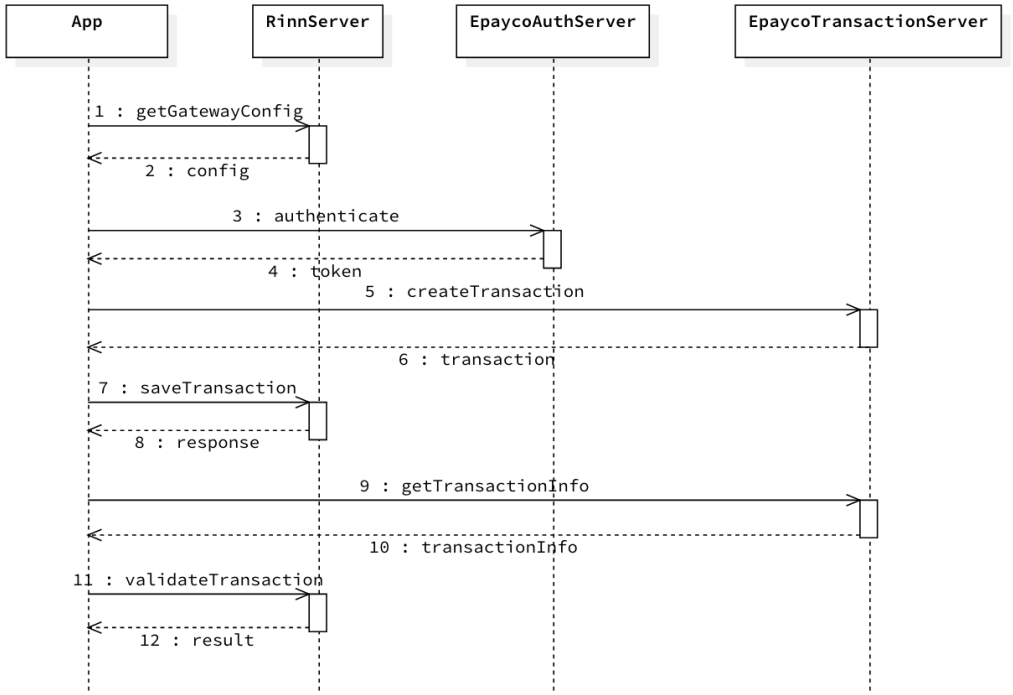


Figura 32. Primera solución propuesta PSE

Cabe aclarar que, para crear una transacción, ePayco necesita, entre otros, los siguientes atributos:

- Valor de la transacción.
- Entidad bancaria.
- Tipo de documento del usuario.
- Número de documento del usuario.

Los cuales deben ser encriptados utilizando como método de encriptación AES128 en modo CBC y con un *padding* PKSC7.[10] Para el usuario final, esto se traduce en el formulario presentado en la **Figura 33**.

[10] Tomado de la documentación oficial de ePayco.

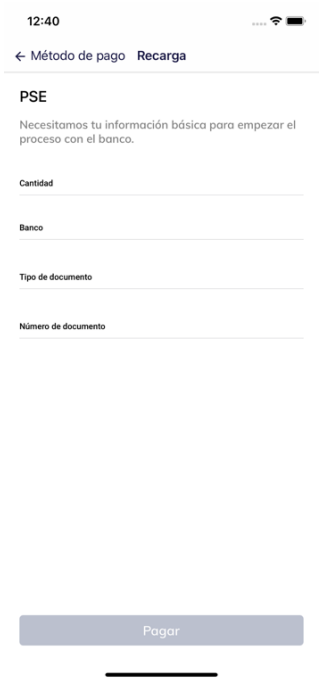


Figura 33. Formulario PSE

Una vez se haya diligenciado el formulario, el usuario será redirigido a la web de su entidad bancaria para continuar con el proceso de pago como se puede apreciar en la **Figura 34**.

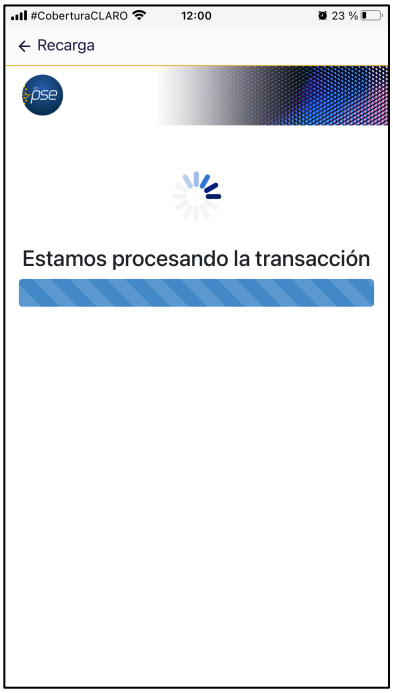


Figura 34. Web view PSE

Ahora, este primer acercamiento, aunque funciona y soluciona la necesidad de la empresa, posee un problema que no puede ser pasado por alto: la aplicación no debe de conocer la lógica que se requiere para interactuar con la pasarela de pago, ya que esto provocaría a futuro que si, por algún motivo, se decide cambiar de pasarela, se tenga que lanzar una nueva versión con una lógica completamente diferente, es decir, su capacidad para adaptarse a cambios es muy baja.

4.2.1.2. Solución Final Propuesta

Al detectar el problema mencionado con anterioridad, se toma la decisión de mover la lógica de la interacción con la pasarela ePayco (autenticación de cada transacción, creación de la transacción y obtención de la información de la transacción) al servidor de Rinn, por lo tanto, la solución (pasos requeridos para recargar la billetera con PSE) queda de la siguiente manera:

- Al ingresar al formulario inicial, se debe obtener la configuración de PSE (entidades bancarias, valor mínimo y máximo).
- Cuando se han diligenciado los campos satisfactoriamente, se crea la transacción.
- Al finalizar el proceso con el banco, se valida la transacción para completar el pago y actualizar el valor a la billetera del usuario.

Esto puede apreciarse mejor en el diagrama de secuencia presentado en la **Figura 35**.

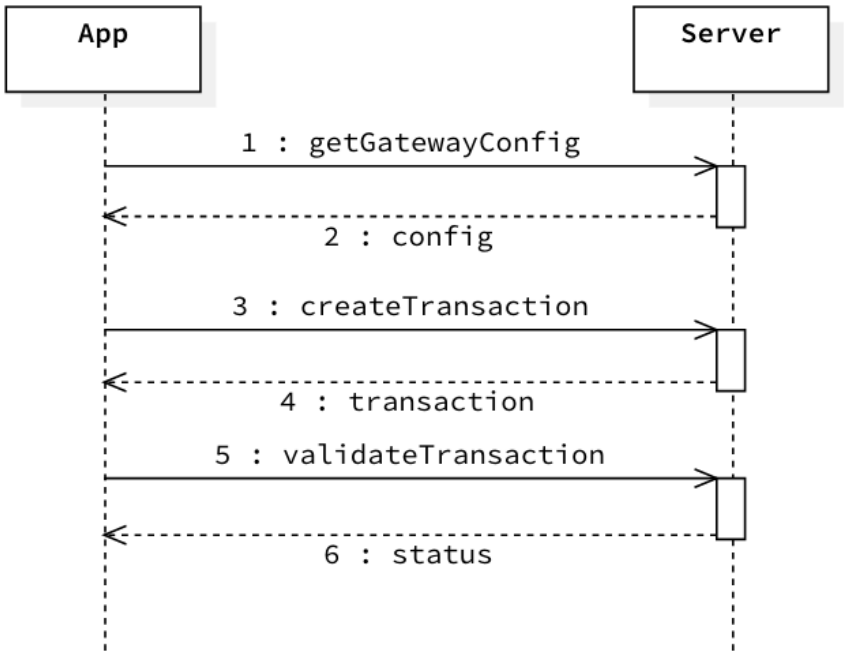


Figura 35. Diagrama de secuencia de la solución final del método de pago PSE

De esta manera, se simplifica el proceso y se minimizan la cantidad de llamadas al servidor que se hacen a través de la red permitiendo así un menor consumo de batería del dispositivo móvil.

Por ultimo, cuando el usuario ha finalizado el proceso con su entidad bancaria, se le mostrará el mensaje éxito o de rechazo según sea el caso y el nuevo balance de su billetera móvil.

4.2.2. FUNCIONALIDAD 2: REDISEÑO CRUD DE TARJETAS

Cuando se habla de rediseñar el CRUD de las tarjetas crédito/debito, se hace referencia a un rediseño completo del módulo (listar tarjetas, ver detalles de tarjeta y agregar tarjeta) ya que, según la empresa, el diseño actual se ha quedado atrás con respecto al de su competencia generando así una baja competitividad dentro del mercado. Además, la empresa deja en claro que se debe tener en cuenta la siguiente restricción: no se pueden crear más pantallas de las que ya tiene este modulo.

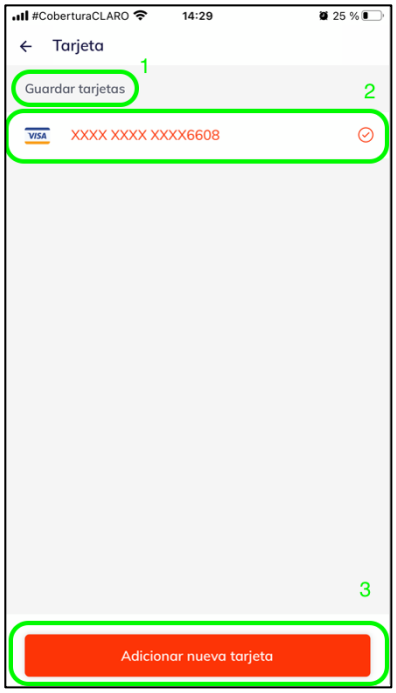
El proceso utilizado para solucionar este problema consta de tres pasos: análisis de los componentes gráficos que componen la vista (para esto se identifica cada uno con un identificador único), identificación de problemas en estos componentes y, por ultimo, solución de estos. A continuación, se documenta el proceso realizado para cada uno de los rediseños de los componentes que componen el módulo.

4.2.2.1. Listar Tarjetas

El **Cuadro 1**, presenta la pantalla principal de este modulo que contiene la implementación actual de la lista de las tarjetas registradas por el usuario junto con la identificación de los elementos gráficos que la componen.

Si se analizan los elementos, se puede apreciar que el elemento No. 1 no tiene relación alguna con la lista que presenta; en el lado derecho del elemento No. 2 se observa lo que parece ser una casilla de selección, pero al dar clic en él, se elimina la tarjeta, un comportamiento completamente inesperado.

Cuadro 1. Componentes gráficos del listar tarjetas

Vista	Descripción de los elementos
	**1:** Subtitulo de la vista. **2:** Celda que comparte cada una de las tarjetas que se listan en la tabla. **3:** Botón que permite adicionar una nueva tarjeta.

Ahora bien, una vez identificados los problemas que posee la implementación actual, se procede a realizar el nuevo diseño tomando como énfasis la solución de estos, la propuesta realizada se presenta en la **Figura 36**. Para el caso del elemento No. 1, se decide cambiar su texto, en tanto que el actual no aporta en nada al usuario; el elemento No. 2 es rediseñado para darle protagonismo agregando sombra y un color mas claro que el fondo, también se cambia la casilla de selección por una flecha que indica al usuario que, al dar clic sobre el elemento, este será redirigido, además, se añade un indicador (estrella) que indica cual es la tarjeta por defecto; por ultimo, el elemento No. 3 es modificado por un símbolo y ubicado en la parte superior derecha para quitarle protagonismo a esta acción y dejar como elemento principal la lista de tarjetas.

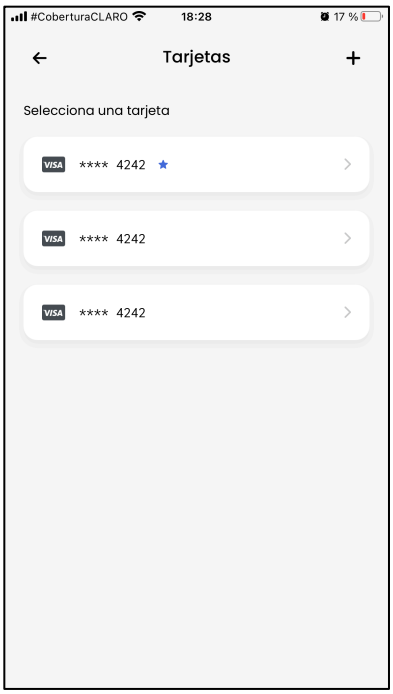


Figura 36. Rediseño de lista de tarjetas

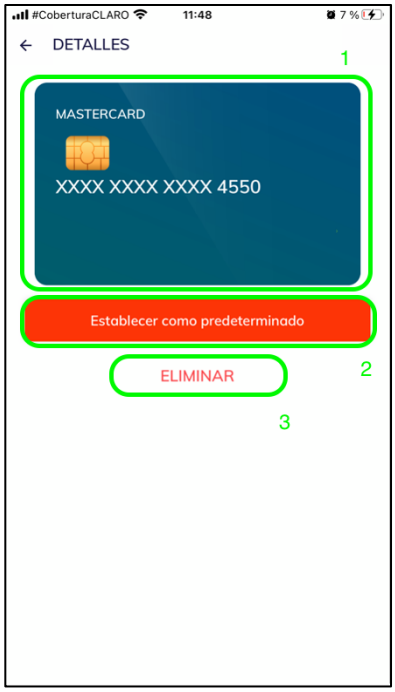
Cabe aclarar que, al dar clic en cualquiera de las tarjetas, el usuario será redirigido a la pantalla que muestra el detalle de la tarjeta.

4.2.2.2. Detalle De Tarjeta

El **Cuadro 2** presenta la pantalla contiene la implementación actual del detalle de las tarjetas registradas por el usuario junto con la identificación de los elementos gráficos que la componen.

Si se analizan los elementos se puede observar que el componente No. 1 presenta los elementos requeridos (marca de la tarjeta y los últimos 4 números) de una buena manera, pero su UI se queda corta con respecto a las tendencias actuales del mercado; el componente No. 2 posee problemas en el sentido de que nunca se esconde, es decir, si una tarjeta ya es por defecto, debería de no mostrarse y mostrar un indicador que le indique al usuario que esta es la tarjeta por defecto; el componente No. 3 tiene una relevancia mayor de la que debería tener, se debe priorizar el ver los detalles de la tarjeta por encima de la funcionalidad que permite eliminarla.

Cuadro 2. Elementos gráficos de detalles de tarjeta

Vista	Descripción de los elementos
	**1:** Representación grafica de la tarjeta que contiene la marca y los últimos 4 dígitos. **2:** Botón que permite establecer la tarjeta como predeterminada. **3:** Botón que permite eliminar la tarjeta.

Después de identificar los problemas que posee la implementación actual, se realiza el nuevo diseño tomando como énfasis la solución de estos, la propuesta realizada se presenta en la **Figura 37**. Para el caso del elemento No. 1 se elimina la imagen del fondo y se aplica un color solido para generar simplicidad y limpieza al diseño, se alinea a la derecha la marca de la tarjeta y se centra el numero de la tarjeta al centro para que gane protagonismo; para solucionar el problema del elemento No. 2, se agrega un indicador justo por encima de la tarjeta para que el usuario entienda que esa en su tarjeta por defecto; por último, el elemento No. 3 fue modificado por un símbolo y ubicado en la parte superior derecha para quitarle protagonismo a esta acción siguiendo así el estilo grafico del listar tarjetas.

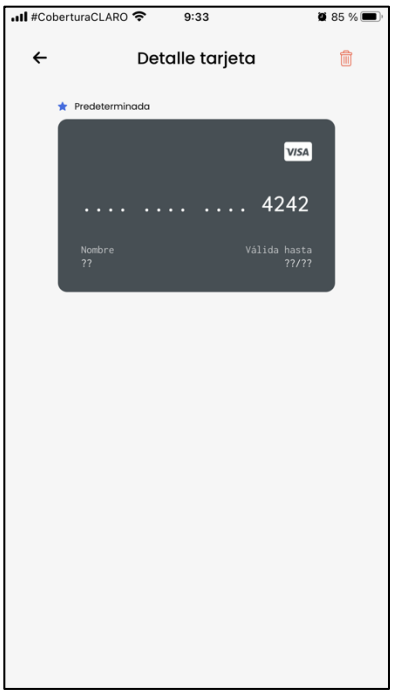


Figura 37. Rediseño detalle de tarjeta

Cabe resaltar que el rediseño del elemento No. 1 fue realizado para ser reutilizado en donde sea requerido. Un ejemplo de esto se puede apreciar en el agregar tarjeta dentro de la siguiente subsección.

4.2.2.3. Agregar Tarjeta

El **Cuadro 3** presenta la captura de pantalla que contiene la implementación actual del formulario que permite registrar una tarjeta junto con la identificación y descripción de los elementos gráficos que la componen.

Si se analizan los elementos de la implementación actual, se puede apreciar que el elemento No. 1 no tiene problemas en cuanto a la función que debe proveer, el único cambio que la empresa solicita es la adición del campo del nombre que aparece en la tarjeta; el elemento No. 2 debe ser removido ya que la empresa no quiere que el usuario sepa quien provee el servicio y cada vez son más las marcas soportadas por la plataforma; el elemento No. 3 tiene la misma relevancia que el elemento No. 4, por lo tanto, eso debe ser modificado para que la función que permite registrar una nueva tarjeta tenga un mayor protagonismo que el escanear.

Cuadro 3. Elementos gráficos del agregar tarjeta

Vista	Descripción de los elementos
	**1:** Formulario que el usuario debe llenar. Contiene el numero de tarjeta, fecha de expiración y CVV (código de seguridad) **2:** Contenedor que muestra las franquicias aceptadas por Rinn, el proveedor del servicio y el cobro que se realizará. **3:** Botón que permite escanear la tarjeta. **4:** Botón que permite agregar la tarjeta.

Una vez identificados los problemas que posee la implementación actual, se modifica el diseño tomando como énfasis la solución de estos, la propuesta realizada se presenta en la **Figura 38**. Para el caso del elemento No. 1, se cambia el estilo de los campos de texto, siguiendo con un estilo mas moderno y se agrega el campo del nombre; para el elemento No. 3 se realiza un redimensionamiento del botón para restarle relevancia; por ultimo, el elemento No. 4 tiene ajustes menores en cuanto al color y tipografía.

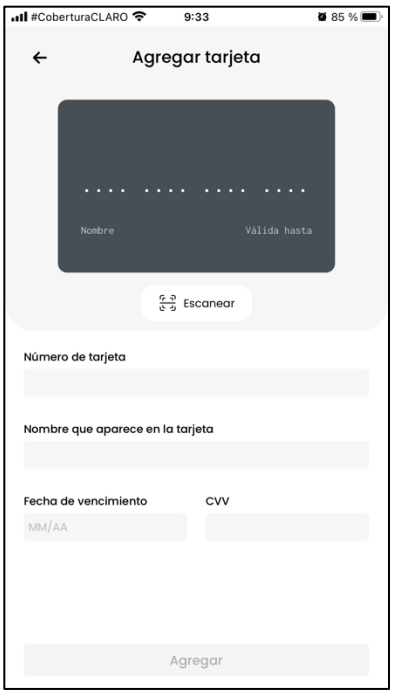


Figura 38. Rediseño de agregar tarjeta

Es importante resaltar la mejora en cuanto a usabilidad que tiene esta vista, en tanto que es mucho mas claro para el usuario saber qué dato ha ingresado en el formulario.

4.2.3. FUNCIONALIDAD 3: REDISEÑO DE BILLETERA

Cuando se habla de rediseñar la billetera, se hace referencia a un rediseño completo del modulo (vista principal de la billetera, historial de movimientos, recargas y detalles de movimiento) ya que el diseño actual, al igual que el modulo de tarjetas, se ha quedado atrás con respecto al de su competencia. Esta funcionalidad se rige por el mismo proceso de tres partes utilizado en el rediseño del modulo de tarjetas. Además, se debe tener en cuenta dos requisitos solicitados por la empresa: la creación de una nueva pantalla para ver el detalle de un movimiento y colocar los últimos 5 movimientos dentro de la vista principal de la billetera.

A continuación, se documenta el proceso realizado para cada uno de los rediseños de los componentes que componen el módulo de billetera.

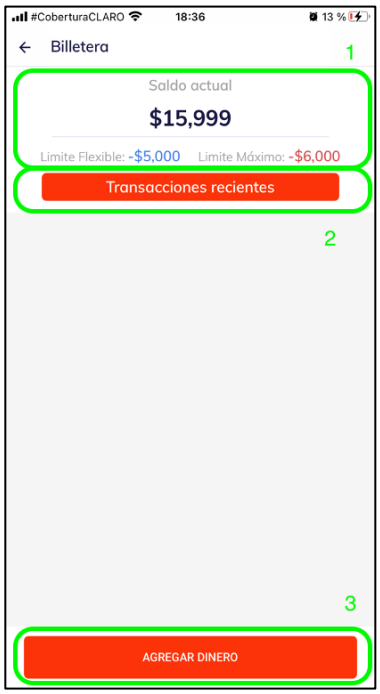
4.2.3.1. Pantalla Principal

El **Cuadro 4** presenta la pantalla contiene la implementación actual del detalle de las tarjetas registradas por el usuario junto con la identificación de los elementos gráficos que la componen.

Al analizar los elementos, se observa que el elemento No. 1 muestra de forma adecuada los datos requeridos, pero la empresa solicita prescindir del limite flexible y el limite máximo, en tanto que se necesita que estos datos sean solamente conocidos por los

administradores de la plataforma; los elementos No. 2 y No. 3 poseen un problema y es que tiene una relevancia mayor de la que debería tener dentro de la pantalla.

Cuadro 4. Elementos gráficos de la billetera

Vista	Descripción de los elementos
	**1:** Contenedor que muestra el saldo actual de la billetera, el limite flexible y el limite máximo. **2:** Botón que permite accede al historial de movimientos. **3:** Botón que permite acceder a la pantalla de selección de método de recarga.

Después de identificar los problemas que posee la implementación actual, se procede a realizar el nuevo diseño tomando como énfasis la solución de estos, la propuesta realizada se presenta en la **Figura 39**. Para los problemas de los elementos No. 1 y No. 3, se opta por unirlos en un nuevo elemento donde se muestre el balance y la opción de recargar.

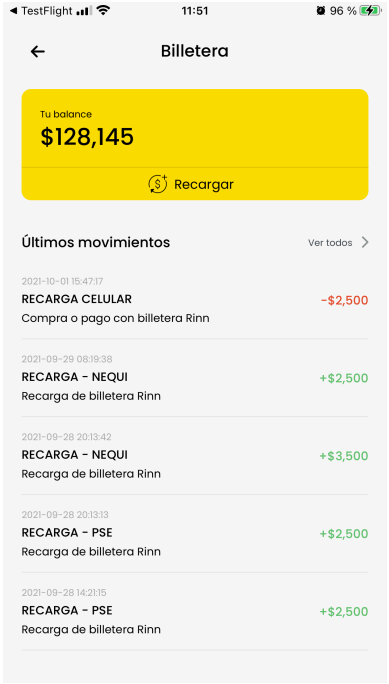


Figura 39. Rediseño de pantalla principal de billetera

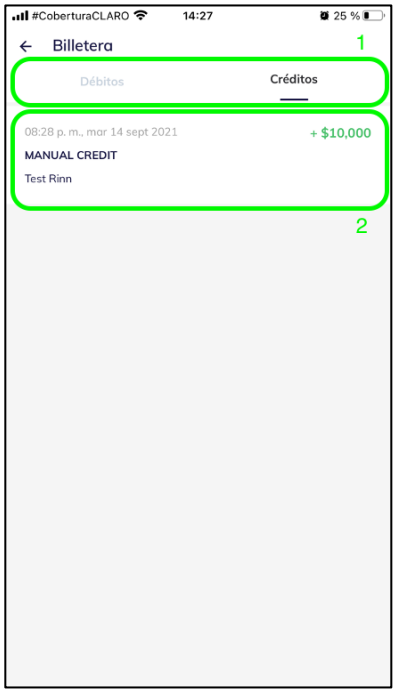
Se puede apreciar la adición de un nuevo elemento grafico ubicado en la parte inferior de la pantalla que muestra los últimos 5 movimientos de la billetera junto con un botón ubicado a la derecha del subtitulo que permite acceder al historial de los movimientos. De esta forma se soluciona el problema del componente No. 2 y se cumple con uno de los dos requerimientos con respecto al rediseño de este módulo. Cabe aclarar que se tomó como referencia las billeteras móviles *Movii* y *Tpaga* para desarrollar el diseño.

4.2.3.2. Historial de Movimientos

El **Cuadro 5** presenta la pantalla contiene la implementación actual del historial de los movimientos junto con la identificación de los elementos gráficos que la componen.

Una vez analizado el componente, se identifican dos problemas: el primero, al dar clic en un movimiento, no se muestra el detalle del mismo; el segundo, la paginación no funciona. Además, para esta pantalla, la empresa solicita remover el elemento No. 1 y combinar las listas de débitos y créditos en una sola.

Cuadro 5. Elementos gráficos del historial de la billetera

Vista	Descripción de los elementos
	**1:** Pestañas que permiten navegar entre los débitos y créditos. **2:** Contenedor que muestra la fecha del movimiento, su titulo, su descripción y el valor del movimiento.

Ahora bien, una vez identificados los problemas con los que cuenta la implementación actual, se procede a realizar el nuevo diseño tomando como énfasis la solución de estos, la propuesta realizada se presenta en la **Figura 40**. El elemento No. 1 es suprimido y las dos listas combinadas en una sola, la cual se encuentra dentro de un contenedor cuyo color es mucha mas claro que el fondo, permitiendo así darle una mayor relevancia al contenedor y limpieza al diseño en general. El elemento No. 2 fue levemente modificado, en tanto que solo se reubicaron los datos para que gane protagonismo el titulo y el valor del movimiento.

Figura 40. Rediseño del historial de billetera

La pantalla generada al dar clic en un movimiento de la lista se encuentra en dentro del componente “Detalles de movimiento”.

4.2.3.3. Rediseño De Recargas De La Billetera

Una de las grandes preocupaciones dentro de la organización para este componente es la escalabilidad ya que, a futuro, se pretende tener una gran variedad de métodos de recarga que puedan ser llamados desde cualquier lugar (modulo) de la aplicación, es decir, se debe diseñar un modulo que permita recargar la billetera completamente desacoplado del resto.

Como se puede apreciar en la **Figura 41**, se modifica el flujo de una recarga para que pueda ser utilizado por cualquier modulo del aplicativo de tal manera que el componente de entrada es la selección del método de recarga, también se muestra el *Modulo xxx* que representa al modulo que requiere de esta funcionalidad. Además, se puede ver la adición de un elemento nuevo *Resultado recarga* que se encargará de mostrarle al usuario el resultado de su recarga, es decir, mostrar al usuario su nuevo saldo de su billetera, el valor recargado, el método utilizado para pagar, el estado de la transacción, etc.

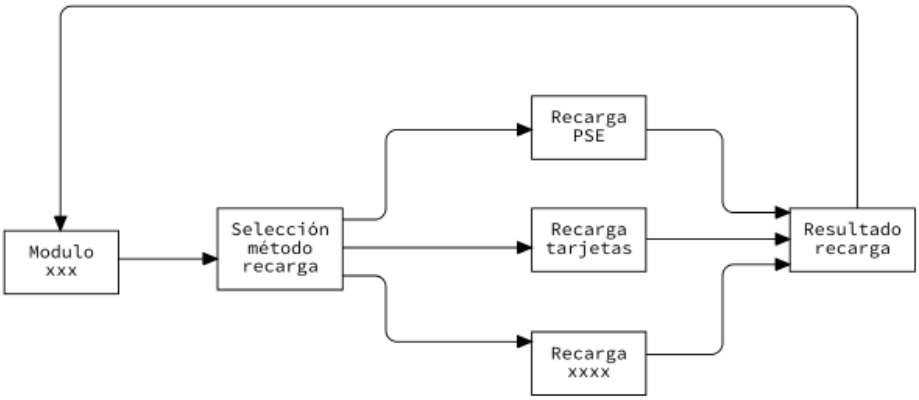


Figura 41. Nuevo flujo de recarga

Para poder comprender mejor este flujo se procede a realizar un ejemplo, a través de simples pasos, donde se realiza una recarga desde la billetera móvil del aplicativo (**Figura 39**) mediante el método de PSE. A continuación, se documenta este ejemplo:

1. Ingresar a la billetera móvil.
2. Dar clic en “Recargar”.
3. Dentro de la pantalla de selección del método de recarga, seleccionar un método.
4. Dentro del formulario de PSE, ingresar los datos correspondientes para poder crear una transacción.
5. Realizar el pago en la plataforma web de la entidad bancaria.
6. Dentro de la pantalla de Resultado de recarga, dar clic en “Regresar” para volver a la billetera del usuario.

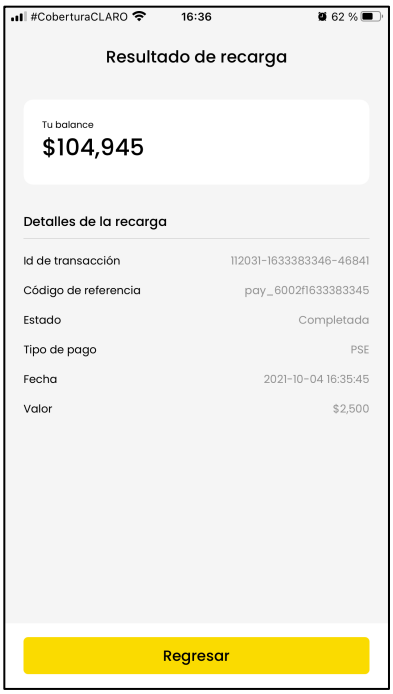
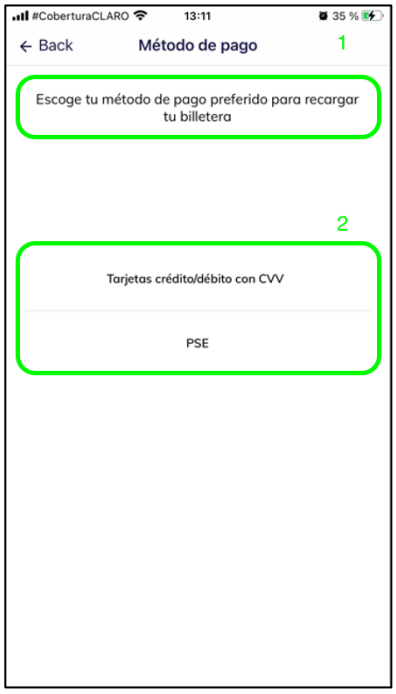


Figura 42. Pantalla que muestra el resultado de una recarga

Para poder continuar con el rediseño de cada uno de los métodos de recarga, se debe adaptar el diseño actual de la pantalla de los métodos de recarga al nuevo estilo grafico utilizado para los demás módulos. En el **Cuadro 6**, se presenta la pantalla que contiene la implementación actual de los métodos de recarga junto con la identificación de los elementos gráficos que la componen.

Al analizar el componente, se encontraron dos problemas con los elementos: el primero, el subtitulo es demasiado largo; el segundo, cada una de las filas de la lista de métodos de recarga debe representar un botón para que el usuario de inmediato sepa que ahí puede dar clic para que suceda una acción.

Cuadro 6. Elementos gráficos de los métodos de recarga

Vista	Descripción de los elementos
	**1:** Subtitulo de la vista. **2:** Lista que contiene los distintos métodos de recarga disponibles en la plataforma.

Ahora bien, una vez identificados los problemas que posee la implementación actual, se procede a realizar el nuevo diseño tomando como énfasis la solución de estos, la propuesta realizada se presenta en la **Figura 43**. Para diseñar la lista de métodos de recarga se sigue el estilo utilizado para la lista de tarjetas (**Figura 36**) dado que ambos elementos gráficos poseen una funcionalidad similar y también para tener una consistencia grafica a través de todo el aplicativo.

Figura 43. Nuevo diseño de los métodos de recarga

Además del cambio en la interfaz gráfica, se realizó un cambio en la lógica de esta pantalla. En la implementación actual (**Figura 30**) los métodos de recarga se encuentran dentro del código del aplicativo lo cual impide deshabilitar cualquiera de ellos sin sacar una nueva versión a la tienda de Apple, es por esto por lo que se decidió, al entrar a esta pantalla, listar los métodos de recarga disponibles desde el servidor. De esta manera se tiene un mayor control y se impiden futuros errores, en caso de que, por ejemplo, para el caso de PSE, si la pasarela de pagos presenta fallas, solo se deshabilita este servicio desde el servidor mientras el error persista.

Ahora bien, una vez diseñado el nuevo flujo de recarga, se procede a realizar el nuevo diseño de las recargas mediante PSE y mediante tarjetas debito/crédito siguiendo el mismo proceso utilizado para el rediseño del CRUD de tarjeta y del modulo de billetera.

4.2.3.3.1. Rediseño de Recarga Mediante PSE

Para la recarga mediante PSE se decide mantener el flujo desarrollado inicialmente. Los cambios que se realizan en esta funcionalidad son únicamente estéticos sobre el formulario que permite crear la transacción (**Figura 33**), siguiendo con el mismo proceso de rediseño de los módulos anteriores.

El **Cuadro 7** presenta la pantalla que contiene la propuesta inicial del formulario de PSE junto con la identificación de los elementos gráficos que la componen.

Cuadro 7. Elementos gráficos del formulario de PSE

Vista	Descripción de los elementos
	**1:** Titulo y subtitulo del formulario. **2:** Formulario que contiene los campos “Cantidad”, “Banco”, “Tipo de documento” y “Numero de documento”. **3:** Botón que permite crear la transacción.

Después de analizar los elementos gráficos, se llega a la conclusión de que no existen problemas en cuanto a que es claro la función de cada elemento dentro del componente. La única modificación que se propone es el cambio de estilo grafico del formulario y, a requisito de la empresa, se suprime el subtitulo del mismo. Lo anterior, se presenta en la **Figura 44**.

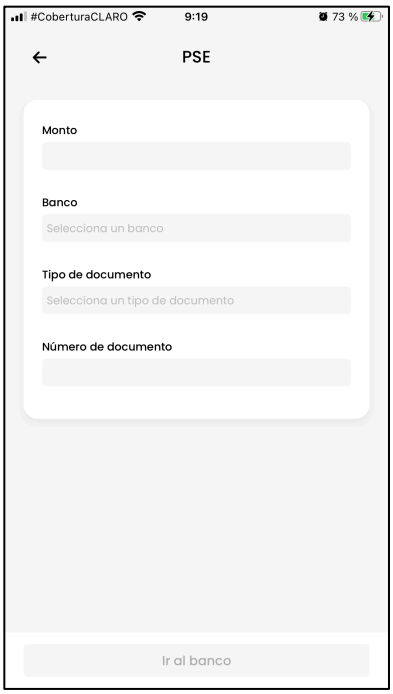


Figura 44. Rediseño del formulario de PSE

De esta forma, se utilizan elementos de módulos anteriores, como los campos de texto y validaciones de estos. Esto para tener uniformidad en el nuevo diseño.

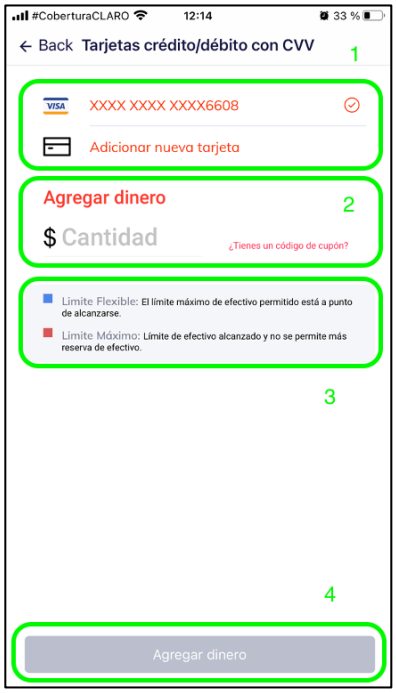
4.2.3.3.2. Rediseño de Recarga Mediante Tarjeta

Para la recarga de billetera con tarjeta, se decide rediseñar el flujo con el que contaba este tipo de recarga y adaptarlo al nuevo flujo principal de una recarga de billetera presentado en la **Figura 41**. Para esto se sigue el mismo proceso que los demás rediseños presentados con anterioridad.

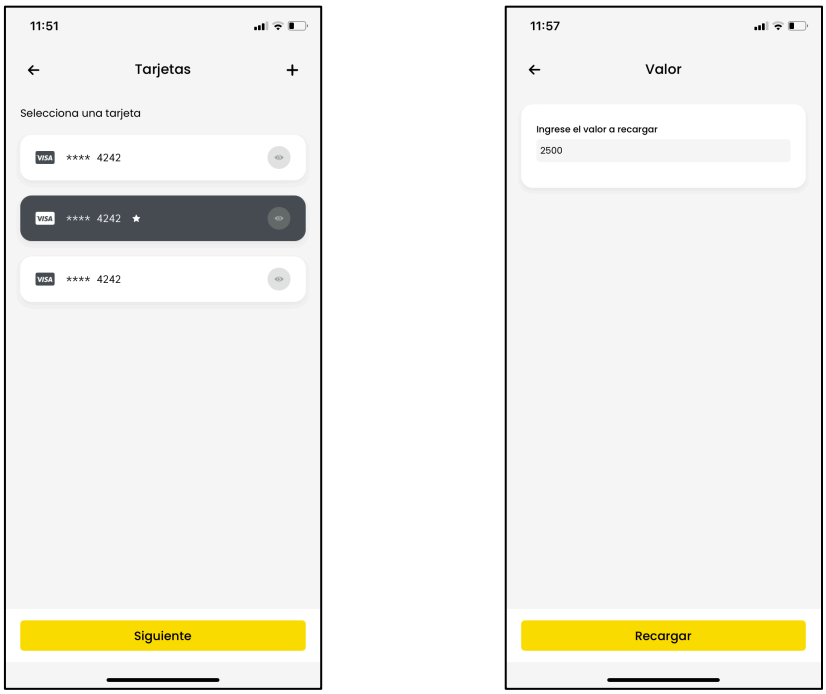
El **Cuadro 8**, presenta la pantalla que contiene la implementación actual de recarga con tarjeta junto con la identificación de los elementos gráficos que la componen.

Después de analizar los elementos gráficos, se encuentran una serie de problemas: el elemento No. 1 funciona bien si la cantidad de tarjetas son pocas o ninguna, pero, ¿qué pasa si, por ejemplos, el usuario ha ingresado 100 tarjetas? En ese caso, solamente la lista de tarjetas seria visible y el formulario y el contenedor serian invisibles para el usuario. Además, como se había explicado con anterioridad, se debe eliminar el elemento No. 3.

Cuadro 8. Elementos gráficos del la recarga con tarjeta

Vista	Descripción de los elementos
	**1:** Lista de tarjetas registradas por el usuario y opción que permite agregar una nueva tarjeta. **2:** Formulario que permite ingresar el valor a recargar. **3:** Contenedor que explica el significado del limite flexible y máximo. **4:** Botón que lanza la transacción.

Para solucionar los problemas encontrados, se propone dividir este componente en un formulario que tiene dos pasos: el primero para mostrar la lista de tarjetas y que además permita seleccionar la tarjeta requerida, y el segundo para ingresar el valor a recargar desde donde se lanzará la transacción. Esto se encuentra dentro de la **Figura 45**.



Paso 1 ***Paso 2***

Figura 45. Rediseño de recarga con tarjeta

Cabe aclarar que, para el paso 1, se reutiliza el componente que lista las tarjetas registradas por el usuario (presentado en la **Figura 36**) y se le agrega la funcionalidad que permite seleccionar la tarjeta requerida.

4.2.3.4. Detalles de Movimiento

Como se dijo con anterioridad, uno de los problemas que posee el componente de historial de movimientos, es que no se tiene una pantalla para que el usuario pueda ver el detalle de un movimiento realizado con su billetera. Para solucionar este problema lo primero que se realiza es una verificación de los datos que se le deben mostrar al usuario, según la empresa, los datos a mostrar (ordenados por relevancia) son los siguientes:

- Valor de la transacción (en caso de ser debito, se debe mostrar “-”, caso contrario un “+”, justo antes del valor)
- Motivo de la transacción.
- Referencia.
- Tipo de transacción.
- Descripción.
- Fecha.

Una vez identificados los datos a mostrar, se presenta el diseño realizado en la **Figura 46**, donde se pretende emular una factura de la vida real, es por esto por lo que se omiten

los bordes redondeados y se agregan unas líneas punteadas entre el valor y la tabla de datos y entre la tabla de datos y la fecha del movimiento.

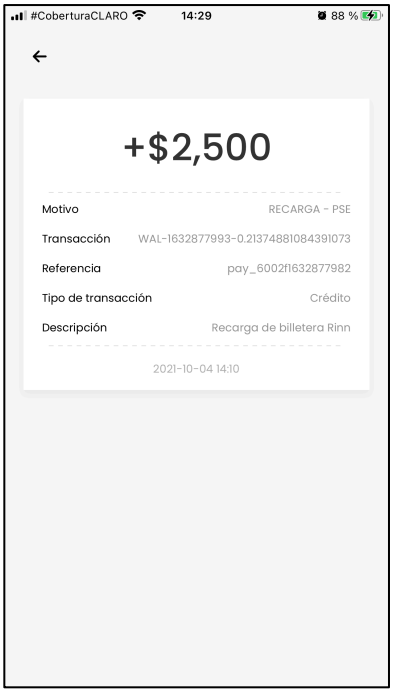


Figura 46. Pantalla del detalle de un movimiento de billetera

Cabe resaltar que se organiza de arriba abajo lo datos requeridos siguiendo su nivel de importancia. Además, el valor del movimiento toma protagonismo dentro del diseño, al igual que una factura de la vida real.

4.2.4. FUNCIONALIDAD 4: MODULO DE PAGOS Y RECARGAS

Esta funcionalidad es una de las funcionalidades más grandes en cuanto a tamaño y complejidad dentro de la practica profesional en tanto que abarca diseño y arquitectura de nuevos componentes, principios de usabilidad, integración con servicios *backend*, entre otros. La empresa solicita el desarrollo de un nuevo módulo dentro del aplicativo iOS que permita al usuario realizar distintos tipos de pagos utilizando únicamente su billetera móvil. Los pagos soportados son: recarga de celulares, pago de servicios públicos y compra de SOAT (para carro y moto), a futuro se piensa implementar la compra de pines de entretenimiento, pero este tipo de pago no se contempló dentro del presente trabajo de grado.

La empresa indica que, al ser la billetera quien contiene a este nuevo modulo, se tomen estos tipos de pagos como productos dentro de la billetera de cada usuario, ese decir, ahora la billetera de Rinn tendrá productos que podrán ser adquiridos utilizando el balance contenido en la misma. Además, se deja en claro que el usuario solamente podrá adquirir un único producto a la vez y que se debe diseñar este nuevo modulo de tal forma que pueda ser llamado desde cualquier lugar de la aplicación.

Desde una perspectiva general, se propone el flujo presentado en la **Figura 47** que presenta la interacción que el usuario realizará para adquirir un producto de la billetera desde un modulo cualquiera, donde indica que se deberá ingresar al flujo especifico del producto a comprar, una vez finalizada esa interacción, se le mostrará la pantalla de confirmación que contiene los detalles del producto a comprar (puede ser vista como un *checkout*), de ahí la lanzará la transacción y mientras se completa se le muestra una pantalla de carga que le indica que se está procesando el pago, una vez realizado, se le mostrará la pantalla que le muestra los detalles de su compra y la opción de regresar al módulo desde donde se inicio la interacción. Este flujo permite que la adición de nuevos productos sea muy simple en tanto que no se debe modificar la lógica de las demás pantallas del módulo.

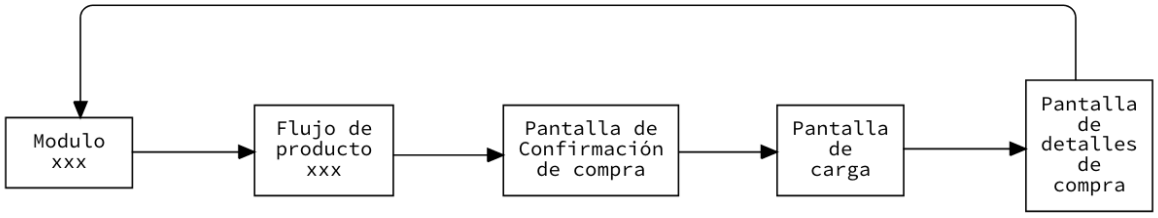


Figura 47. Flujo del nuevo módulo de pagos y recargas

Como se puede apreciar en el flujo del módulo, al finalizar cualquiera de los flujos de los productos de la billetera, el usuario será redirigido a la pantalla de confirmación presentada en la **Figura 48**, donde se le presentará el valor que está a punto de pagar, el tipo de pago, el producto seleccionado, la referencia del producto (para el caso de una recarga celular es el numero de celular, para el caso de SOAT es la placa del vehículo y para el caso de los servicios públicos es el numero de referencia de la factura) y un texto que le indica al usuario que el dinero saldrá de su billetera.

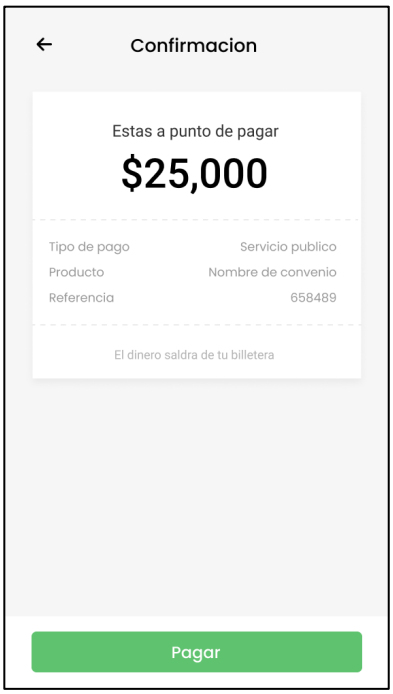


Figura 48. Pantalla de confirmación de pago

Al dar clic en “Pagar”, se valida que el valor actual de la billetera sea mayor o igual al valor del producto a pagar, en caso de que no sea así se le mostrará al usuario un mensaje indicándole que debe recargar su billetera junto con una opción para realizar la recarga (**Figura 49**) para poder continuar con el flujo normal de un pago.

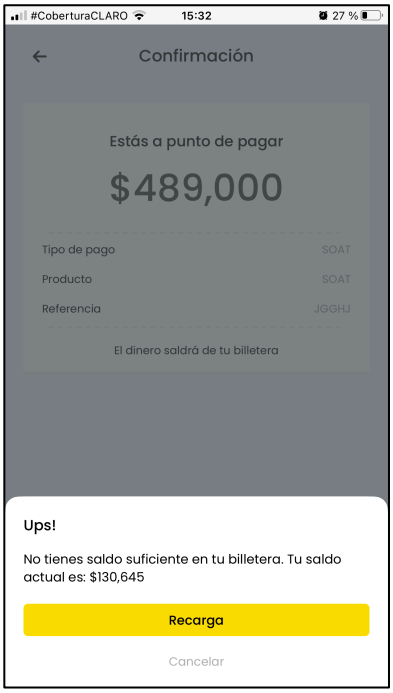
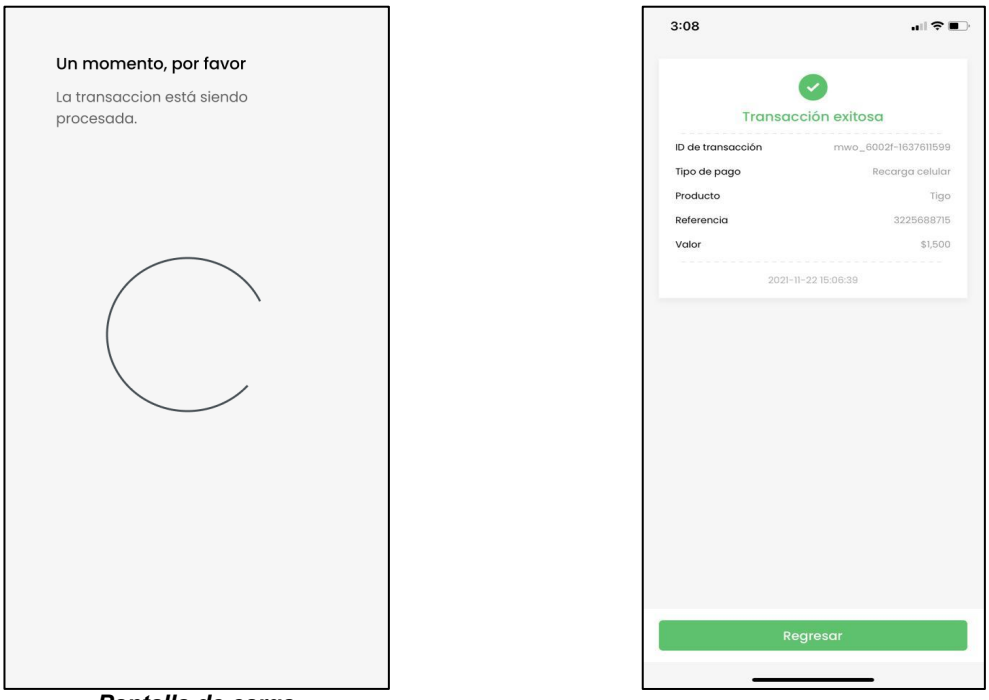


Figura 49. Validación valor de billetera

Una vez realizada esta validación se crea la orden y se envía la información al servidor de Rinn para que se efectúe el pago correspondiente. Mientras se espera la respuesta, se muestra la pantalla que indica que la transacción está siendo procesada y en el momento en que se termina de procesar, se muestra la pantalla que contiene el detalle de la orden presentada dentro de la **Figura 50**. En ella se muestra su identificador asignado para la transacción, el tipo de pago que realizó, el producto comprado, la referencia del producto, el valor del pago y la fecha en que se realizó.



Pantalla de carga

Detalle de orden de pago

Figura 50. Compra de un producto de billetera

Ahora bien, como se puede apreciar en la **Figura 50** y más específicamente en la pantalla de detalle, una orden de compra de un producto siempre tendrá unos atributos comunes a cualquier tipo de producto y estos son: el identificador de transacción, el estado actual de la orden, el tipo de pago, el producto que se adquirió, la referencia de pago, el valor y la fecha en que se realizó.

Una vez diseñados los componentes gráficos comunes a todos los productos, se realiza la documentación de los flujos específicos para cada producto que soportará la billetera. Cabe resaltar que el proceso que realizó para cada el diseño de los flujos de los productos se divide en tres (3) pasos: identificación de los datos que el usuario debe ingresar, identificación de los componentes gráficos y diseño e implementación de las pantallas finales.

4.2.4.1. Recarga Celular

Para este producto, la empresa deja en claro que son tres (3) los datos que el usuario debe ingresar a la aplicación: el operador móvil (el cual debe poder ser seleccionado a partir de una lista provista por un servicio desde el *backend*), el número de celular y el valor a recargar. Además, se debe seguir con el estilo grafico utilizado para los rediseños presentados con anterioridad.

Una vez identificados los datos que el usuario debe ingresar, se procede a agruparlos para poder identificar los componentes gráficos necesarios. Al ser tan solo tres (3) datos se propuso separar cada dato por componente gráfico como se puede apreciar en la **Figura 51** que presenta el flujo para este tipo de producto, donde cada contenedor representa una pantalla dentro del aplicativo.

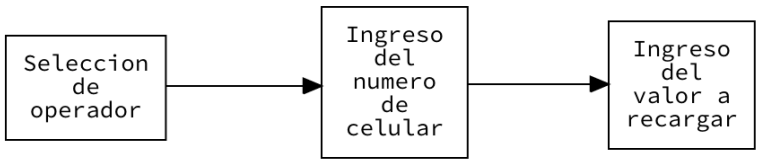
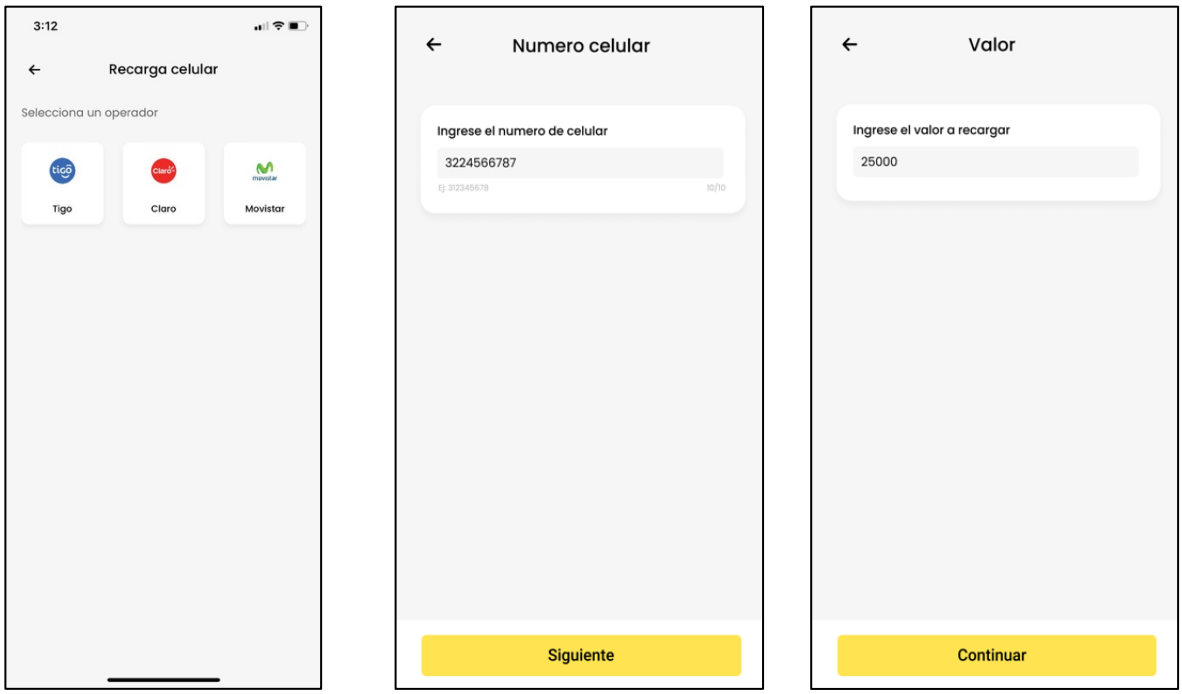


Figura 51. Flujo de una recarga celular

Ahora bien, una vez identificados los componentes gráficos, se realiza el diseño e implementación de cada uno de ellos (**Figura 52. Recarga de celular**). Para el caso de la selección de operadores se opta por el uso de un *grid* donde cada uno de sus elementos cuenta con un titulo y una imagen, este componente es diseñado de tal forma que pueda ser utilizado por cualquier otro modulo de la aplicación. El componente donde el usuario debe ingresar el numero de celular y el valor a recargar utilizan los mismos elementos gráficos para poder cumplir con uno de los requerimientos principales de la empresa: la reusabilidad.



Selección de operador *Número de celular* *Valor a recargar*

Figura 52. Recarga de celular

Cabe aclarar que, una vez finalizada la compra de este producto, el proveedor del servicio se encargará de realizar la recarga y de notificar al usuario.

4.2.4.2. Servicios Públicos

Para este producto, la empresa deja en claro que son dos (2) los datos que el usuario debe ingresar: la empresa proveedora del servicio (se debe poder buscar mediante el nombre y seleccionar la requerida) y el numero de referencia de la factura. Además, como en la recarga de celular, debe seguir el mismo estilo gráfico utilizado para los rediseños presentados con anterioridad.

Una vez identificados los datos que el usuario debe ingresar, se procede a agruparlos para poder identificar los componentes gráficos necesarios. Al ser tan solo dos (2) los datos, se propuso, al igual que con la recarga de celulares, separar cada dato por componente grafico como se puede apreciar en la **Figura 53** que presenta el flujo para este tipo de producto, donde cada contenedor representa una pantalla dentro del aplicativo.

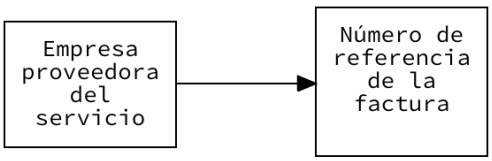
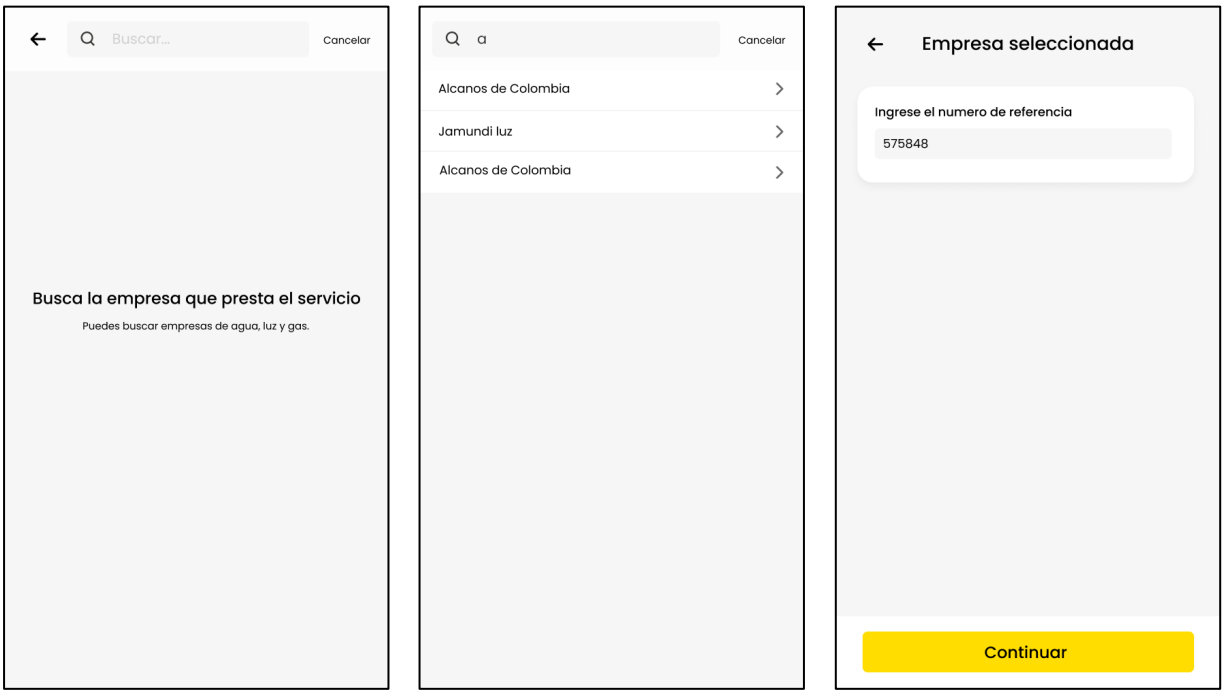


Figura 53. Flujo de pago de servicios públicos

Ahora bien, una vez identificados los componentes gráficos, se procede a realizar el diseño e implementación de cada uno de ellos. Para el caso de la selección de la empresa proveedora del servicio se opta por diseñar una pantalla de búsqueda cuyos resultados sean seleccionables, este componente es diseñado de tal forma que pueda ser utilizado por cualquier otro modulo de la aplicación. Además, el componente donde el usuario debe ingresar el numero de referencia de la factura utiliza los mismos elementos gráficos que el valor a recargar en la recarga de celulares. Lo anterior, se presenta en la **Figura 54**.



Pantalla de búsqueda vacía ***Pantalla de búsqueda con elementos encontrados*** ***Pantalla que permite ingresar la referencia de la factura***

Figura 54. Pago de servicios públicos

Es importante aclarar dos cosas: la primera que, al realizar una búsqueda y el usuario le da clic a un resultado, es redirigido a la pantalla que permite ingresar el número de referencia de la factura; la segunda, una vez realizado el pago del servicio publico, es el proveedor de servicio quien se encarga de realizar el pago con la empresa prestadora del servicio y de notificar al usuario.

4.2.4.3. SOAT

El caso de la compra de SOAT es particular en tanto que son bastantes los datos que el usuario debe ingresar, estos son: placa del vehículo, tipo de identificación del propietario, numero de identificación del propietario, dirección, email, tipo de póliza, departamento, municipio, etc. Una vez identificados los datos que el usuario debe ingresar, se agrupan para poder identificar los componentes gráficos necesarios. Se propuso agrupar los datos en tres grupos diferentes: datos de búsqueda del vehículo, tipo de póliza e

información del vehículo y propietario. Esto se puede apreciar en el flujo presentado en la **Figura 55**.

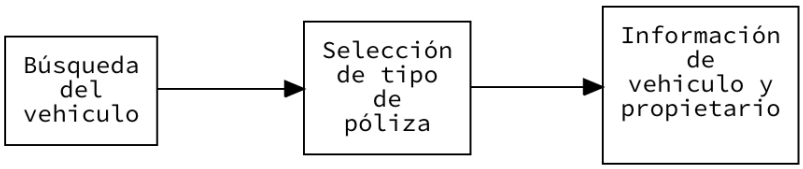
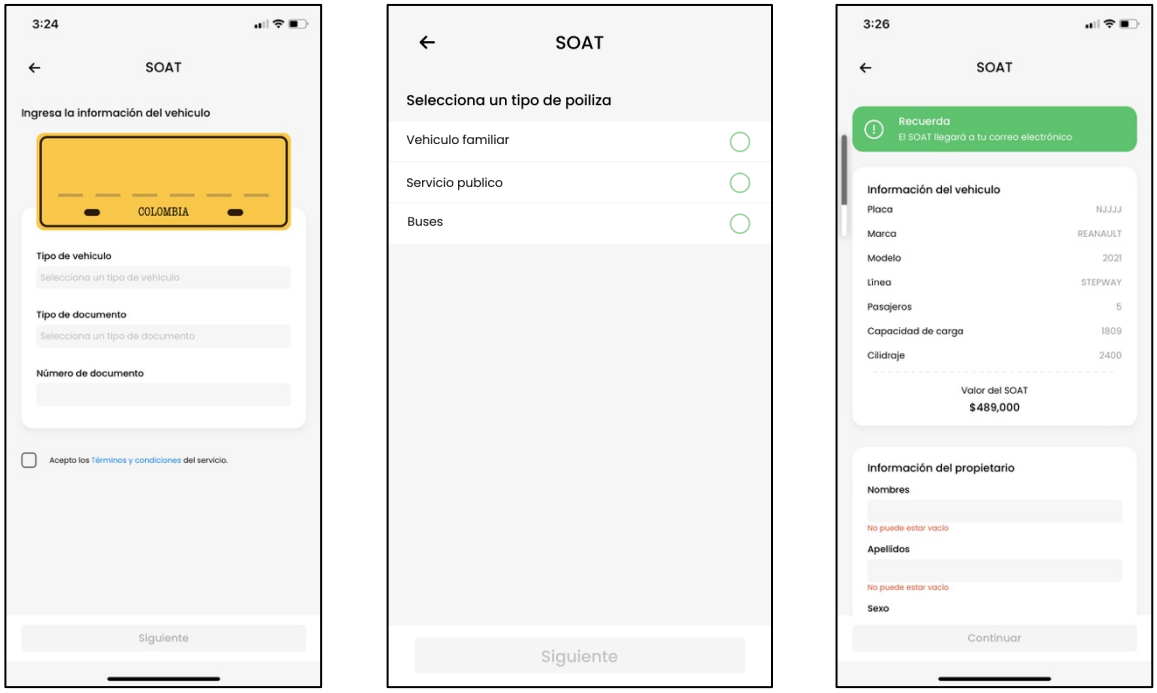


Figura 55. Flujo de compra de SOAT

Una vez diseñado el flujo especifico del producto y al revisar la documentación del proveedor del servicio, se observa que para poder cotizar un SOAT se debe buscar el vehículo mediante su placa y el numero de identificación del propietario (esto se hace para poder verificar que es el propietario quien se encuentra realizando la cotización), por lo tanto, el primer componente grafico (contiene un formulario para que el usuario ingrese esos datos, lo que se propone con este componente es emular las placas de los vehículos de la vida real para que el usuario encuentra la interfaz mucho más amigable y cercana. Es posible que un vehículo tenga más de un tipo de póliza, ejemplo: bus y vehículo público, en caso de que sea así, el usuario debe seleccionar el tipo de póliza que desea comprar. Después de seleccionar el tipo de póliza, se le muestra la información del vehículo junto con el valor del SOAT y el formulario de la información del propietario.



Búsqueda de vehículo ***Tipo póliza*** ***Información del vehículo***

Figura 56. Compra de SOAT

Al finalizar la compra del SOAT, el proveedor del servicio se encarga de enviar la copia de la póliza generada al correo y un SMS con el enlace de esta al celular ingresado por el usuario.

4.2.5. FUNCIONALIDAD 5: VALIDACION DE TARJETAS

Esta es una de las funcionalidades más pequeñas que se realizaron dentro de toda la practica profesional, sin embargo, es una de las mas importantes. La empresa requería la adición de una nueva capa de seguridad dentro del componente agregar tarjeta de la plataforma ya que algunos intentos de posible fraude fueron reportados.

Para solucionar este problema se añade una nueva validación en el momento que el usuario ingresa los datos de la tarjeta y se le realiza el cobro. Esta validación consiste en pedirle al usuario que ingrese el valor aleatorio que fue cobrado a su tarjeta (presentado en la **Figura 57**) para que de esta manera se demuestren dos cosas: que la tarjeta existe y que pertenece al usuario en cuestión. Cabe aclarar que se dispondrán de hasta 3 intentos (este limite es variable) para que pueda ser validada, en caso de que no pueda ser efectuada la validación se deberá volver a ingresar la tarjeta e iniciar el proceso nuevamente.

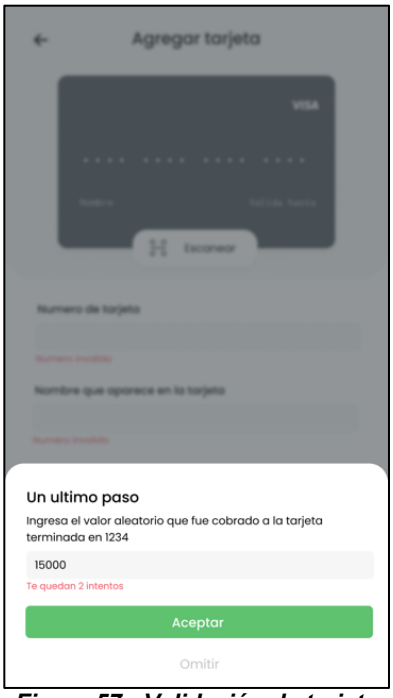


Figura 57 - Validación de tarjeta

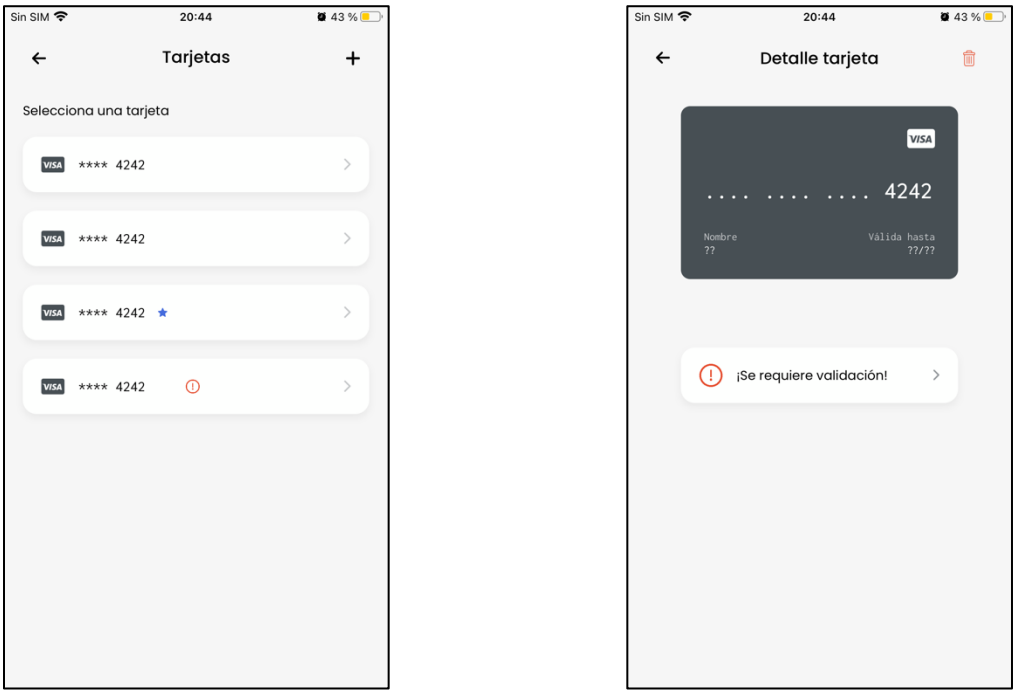
A continuación, en la **Tabla 23**, se listan los estados de validación que tiene una fuente de pago (tarjetas de crédito / débito):

Tabla 23 - Estados de una fuente de pago

Estado	Descripción
PENDING	Estado inicial de todas las fuentes de pago al momento de ser registradas en la plataforma.

VALIDATED	La fuente de pago ha sido validada exitosamente por el usuario.
NOT_VALIDATED	La fuente de pago no pudo ser validada.

Como se puede apreciar en la **Tabla 23**, al agregar una tarjeta, esta tiene un estado de *PENDING* puesto que no se ha validado, para que el usuario entienda esto se propone una variante al diseño del componente de la lista de tarjetas presentada en la **Figura 58**, donde se puede apreciar un indicador en a ultima tarjeta que le indica al usuario que esa tarjeta requiere atención, también se modifica el diseño del componente de detalles de tarjetas presentado en la **Figura 58**, donde se muestra un nuevo elemento grafico que le indica al usuario que la tarjeta requiere ser validada.



Lista de tarjetas que contiene una tarjeta no validada

Detalles de tarjeta no validada

Figura 58. Variante al diseño de tarjetas pendientes de validación

Por ultimo, cabe resaltar que el usuario solamente podrá utilizar las tarjetas que hayan pasado con éxito este nuevo proceso de validación.

4.2.6. FUNCIONALIDAD 6: METODO DE PAGO NEQUI

La ultima funcionalidad realizada dentro de la empresa fue la adición de un nuevo método de pago utilizando Nequi. Esto hace referencia a una nueva forma con la cual el usuario podrá realizar recargas a su billetera móvil y pagar sus ordenes dentro de la plataforma. Para esto se basó en el diseño implementado para el método de pago con tarjeta. Además, la empresa deja en claro que un usuario puede tener cero o muchos Nequi vinculados a su cuenta.

4.2.6.1. CRUD de Nequi

Una de las sub-funcionalidades que componen esta nueva funcionalidad es el CRUD de Nequi, con el cual el usuario podrá gestionar sus cuentas vinculadas. Podrá agregar, listar y ver los detalles.

4.2.6.1.1. Agregar

Para poder adicionar o vincular una cuenta de Nequi a Rinn, el usuario debe ingresar el nombre, correo electrónico, número de documento y el numero de celular con el cual se encuentra registrada la cuenta. Esto se traduce en un formulario (**Figura 59**) que contiene los datos anteriormente mencionados y con el mensaje que indica que se debe aprobar la cuenta desde Nequi.

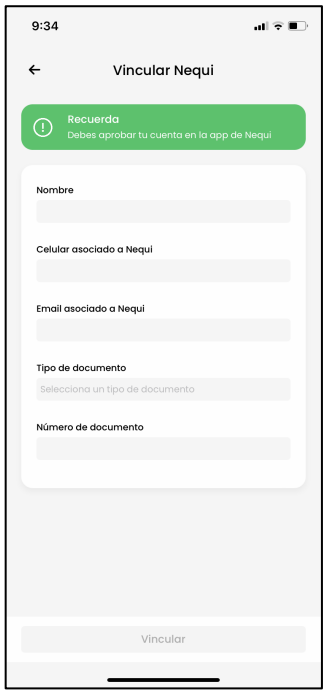


Figura 59. Formulario que permite agregar un Nequi

Una vez se haya vinculada exitosamente la cuenta, al igual que con las tarjetas, la cuenta de Nequi debe ser validada para poder ser utilizada dentro de la plataforma. Esta validación, a diferencia de las tarjetas, la realiza el usuario dentro del aplicativo de Nequi.

4.2.6.1.2. Listar

Como se había dicho con anterioridad, un usuario puede tener varias cuentas de Nequi vinculadas, es por esto por lo que se decide implementar una pantalla que contenga todas las cuentas del usuario que en encuentren validadas o pendientes de validación (**Figura 60**). Desde ahí se podrán registrar nuevos Nequi y acceder al detalle de cada uno.

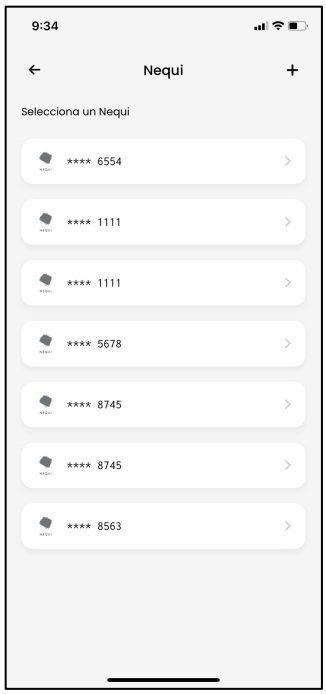
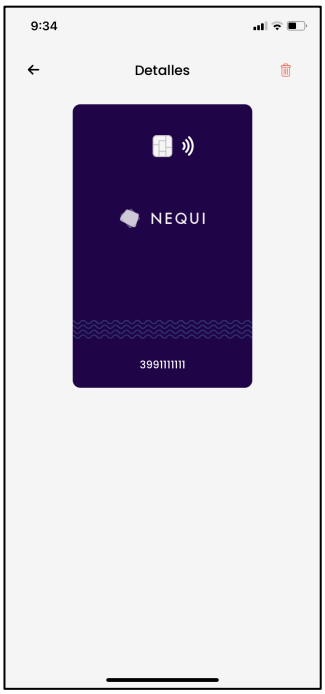


Figura 60. Lista de Nequi vinculados

Como se puede apreciar, para esta pantalla, se reutilizaron los elementos gráficos del rediseño de la lista de tarjetas (**Figura 36**), ya que el problema a solucionar era muy similar y el estilo gráfico debía ser el mismo.

4.2.6.1.3. Detalles

Para acceder a los detalles de un Nequi, el usuario deber dar clic sobre la cuenta dentro de la lista de Nequi vinculados. Según la empresa, se debe mostrar el numero de teléfono de la cuenta y su estado de validación actual, para esto se diseña una pantalla que contenga los datos solicitados contenidos en un elemento gráfico que emula a las tarjetas debito físicas de Nequi. Además, se muestra el detalle de un Nequi pendiente por ser validado. Lo anterior se presenta en la **Figura 61**.



Detalle de un Nequi validado

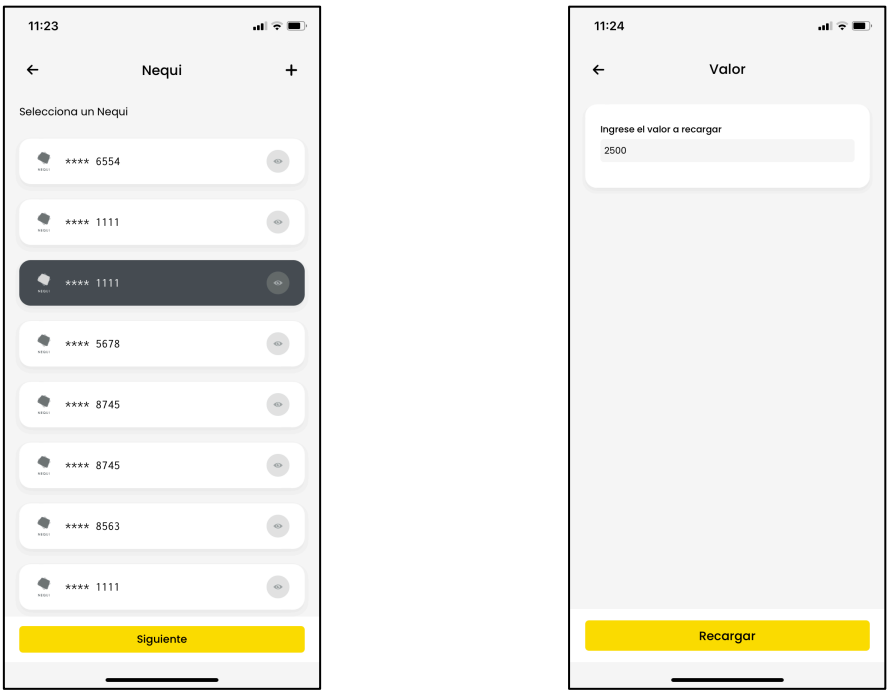
Detalle de un Nequi pendiente por ser validado

Figura 61. Detalle de un Nequi

Cabe resaltar que, al igual que el detalle de tarjeta, el usuario puede eliminar su Nequi al dar clic sobre el icono ubicado en la parte superior derecha.

4.2.6.2. Recarga de Billetera

Para la empresa es relevante brindar muchas facilidades al usuario para poder recargar su billetera móvil, es por esto por lo que Nequi, además de ser una fuente de pago con la cual se puede pagar las ordenes, se podrá también recargar la billetera utilizando las cuentas que se encuentren validadas. Para esto se utiliza el mismo flujo de recarga con tarjeta por pasos, es decir, como primer paso: seleccionar el Nequi, después, en otra pantalla, como segundo paso: ingresar el valor a recargar. Estos pasos se presentan en la **Figura 62**.



Paso 1 ***Paso 2***

Figura 62. Recarga mediante Nequi

Cabe resaltar que este nuevo método de recarga se acopla al nuevo flujo de recarga de billetera que se había presentado con anterioridad.

5. CONCLUSIONES

A partir de los requisitos entregados por la empresa, se pudo observar que la planeación inicial para el desarrollo de la presente práctica profesional fue adecuada para desarrollar todas las funcionalidades solicitadas. Para esto se dividieron los requerimientos en funcionalidades y estas, a su vez, en actividades que permitieron mayor control del trabajo realizado y al momento de la estimación de tiempo y su priorización.

Mediante la caracterización de la aplicación, se pudo inferir que la arquitectura *Model-View-ViewModel* (MVVM) resuelve problemas de escalabilidad y adaptabilidad que posee la arquitectura *Model-View-Controller* (MVC) dentro del desarrollo de aplicativos iOS. Por lo tanto, se verificó su adaptabilidad con mayor facilidad a los cambios de requisitos que son frecuentes dentro del mundo de las aplicaciones móviles actuales.

A medida que se fueron desarrollando las principales funcionalidades del aplicativo mediante el uso de metodologías agiles, y más específicamente *SCRUM*, se encontró que este tipo de enfoques agiles se adaptan satisfactoriamente a los desarrollos móviles dado que poseen alta variabilidad con respecto a sus requerimientos, y *SCRUM* permite adaptarse a este tipo de cambios.

Vale destacar, que las herramientas desarrolladas por la industria, tales como *Firebase* de *Google* y más específicamente *Crashlytics,* facilitaron la correcta ejecución de las actividades puesto que permitieron al equipo de desarrollo observar, entre otras cosas, los errores más comunes que tenían los usuarios de la aplicación en producción. Es así como, se redujo la cantidad de *crashes* de la aplicación, pasando de tener alrededor de 100 por semana a más o menos 20 semanales, es decir, una reducción cercana al 80%.

También se pudo observar, mediante la utilización de *Onesky*, que la capacidad que tiene una aplicación para traducir su contenido es una característica o atributo (*feature)* fundamental para todas las aplicaciones que pretendan atraer más usuarios a sus plataformas digitales, en tanto que esto les permite a los usuarios extranjeros (en este caso, no hispanohablantes) adquirir los servicios ofrecidos por la compañía, lo que se traduce en un mayor ingreso económico para la empresa.

Además, a partir de los objetivos planteados al inicio de la practica profesional y más específicamente el objetivo de la implementación de un nuevo método de pago PSE, se pudo observar que un problema del mundo real no tiene una única solución y que, cualquiera que sea la solución propuesta, siempre existirán mejoras a realizar a partir de la misma.

Durante el diseño e implementación de las funcionalidades de la aplicación y dado que se trabajó dentro de un equipo de desarrollo, se pudo fortalecer algunas habilidades blandas indispensables dentro del mundo empresarial, tales como: el manejo del estrés, gestión del tiempo, comunicación escrita y oral, adaptación al cambio, entre otras.

Por último, el trabajo en equipo fue un factor fundamental para poder llevar a buen termino las funcionalidades requeridas por la empresa. El compañerismo, el trato de igual a igual con el asesor, la constante disponibilidad por parte de la directora del trabajo de grado y el buen ambiente laboral permitieron que la presente práctica profesional haya sido concluida exitosamente en tanto que todos y cada uno de los requerimientos fueron desarrollados acorde a las especificaciones entregadas por la compañía.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] A. Sánchez, «Economía colaborativa: un nuevo mercado para la economía social,» *Revista de Economía Pública, Social y Cooperativa,* nº 88, pp. 231-258, 2016.

[2] S. Negri, «¿Cómo es trabajar en plataforma de delivery? - aproximaciones de una investigación preliminar,» de *XIII Jornadas de Sociología*, Buenos Aires, 2019.

[3] P. S. Jurado, «Comercio electrónico en el Ecuador,» 2018.

[4] P. J. Muñoz, de *Economía colaborativa y plataformas digitales*, Reus, 2020, p. 108.

[5] A. Fernandez, M. C. Sánchez, H. V. Jiménez y R. Hernández, «La importancia de la innovación en el Comercio Electrónico,» *Universia Business Review,* nº 47, pp. 106-125, 2015.

[6] D. M. Vigoya Castillo y F. Zambrano, Identificación de los riesgos financieros asociados al modelo agregador de las pasarelas de pago en Colombia., Corporación Universitaria Minuto de Dios, 2020.

[7] A. Vásquez y M. A. Arístides, Integración de la pasarela de pagos con los facilitadores utilizando protocolo HTTPS y servicios web para comercio electrónico, Universidad Mayor de San Marcos, 2017.

[8] A. Soler Patiño, «¿Hacia donde va el comercio electrónico en Colombia?,» *Ploutus,* vol. 4, nº 1, 2014.

[9] A. M. Prieto y J. D. Torres, Estudio sobre los sistemas de pago bajo valor y su regulación, Unidad de Regulación financiera, 2018.

[10] L. N. Aldana y M. J. M. Moreno, «Di pag "pago al instante",» 2020.

[11] C. Y. K. Ríos, El rol de los intermediarios como agentes generadores de confianza dentro del comercio electrónico colombiano basado en un sistema de ventas de consumidor a consumidor (C2C), 2020.

[12] «Documentación oficial de ePayco,» [En línea]. Available: https://docs.epayco.co. [Último acceso: 16 10 2021].

[13] Bancolombia, «Wompi,» [En línea]. Available: https://wompi.co.

[14] R. Vallon, M. Brüggemann y G. Thomas, «An agile and lean procress model for mobile app development: case study austrian industry,» vol. 10, nº 11, 2015.

[15] T. Cañizares, C. Gómez y C. Pardo, «Hacia el escalamiento de soluciones agiles en grandes empresas de software: un mapeo sistemático,» *INGE CUC,* vol. 16, nº 2, 2020.

[16] B. Molina, H. Vite y J. Davila, «Metodologias agiles frente a las tradicionales en el proceso de desarrollo de software,» *Espirales revista multidisciplinaria de investigación,* 2018.

[17] A. Kuz, M. Falco y R. S. Giadini, «Comprendiendo la Aplicabilidad de Scrum en el Aula: Herramientas y Ejemplos,» *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología,* nº 21, pp. 62-70, 2018.

[18] E. R. Mendez Soliman, Estimación de esfuerzo en proyectos de desarrollo de software con metodologías ágiles, Universitat Politècnica de València, 2018.

[19] T. Dimes, «Conceptos básicos de Scrum: Desarrollo de software agile y manejo de proyectos agile,» *Babelcube,* 2015.

[20] Apple, «Swift,» [En línea]. Available: https://developer.apple.com/swift/.

[21] J. Hoffman, Mastering Swift 5.3, Packt Publishing, 2020.

[22] «Mix panel,» [En línea]. Available: https://mixpanel.com/trends/#report/ios_11. [Último acceso: 17 10 2021].

[23] A. Koval, Comparative analysis of modern iOS architectures in different development stages, Ukrainian Catholic University, 2021.

[24] *ISO 9241-210: ergonomics of human system-interaction Part 210: human-centered design for interactive systems,* 2019.

[25] ISO/IEC 25000, «Systems and Software Quality Requierements and Evaluation,» 2020.

[26] A. Durán, J. Álvarez, M. C. d. Río y C. P. Maldonado, «Economía colaborativa: Análisis de la producción cientifica en revistas académicas,» *Revista de Gestão e Scretariado,* vol. 7, nº 3, pp. 1-20, 2016.

[27] J. M. Sastre Centeno y M. E. Inglada Galeana, «La economía colaborativa: un nuevo modelo,» *Revista de economía publica, social y cooperativa.*

[28] J. M. Rodriguez Antón, M. d. M. Alonso Almeida y L. Rubio Almeida, «La economía colaborativa. Una aproximación al turismo colaboratico en España,» *Revista de Economía Pública, Social y Cooperativa,* nº 88, pp. 259-283, 2016.

[29] J. G. Enríquez y S. I. Casas, «Usabilidad en aplicaciones móviles,» *Informes cientificos técnicos,* vol. 5, nº 2, pp. 25-47, 2014.