UCAB Universidad Católica ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMA DE PAGOS A TRAVÉS DE UN PIN ÚNICO PARA SERVICIOS CON ACCESIBILIDAD REDUCIDA A INTERNET PARA CITYWALLET.

Este Jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado: DIECINUEVE (19) PUNTOS

JURADO EXAMINADOR

Firma:	Firma:	Firma:
Nombre: BIAGIO CANTE	Nombre: Wilmer Pereira	Nombre:

REALIZADO POR	**German Javier Pérez.**
TUTOR EMPRESARIAL	**Ing. Félix Armando Fernández.**
TUTOR ACADÉMICO	**Ing. Biagio Cante.**
FECHA	**Julio 2017.**

UCAB Universidad Católica ANDRÉS BELLO

FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMA DE PAGOS A TRAVÉS DE UN PIN ÚNICO PARA SERVICIOS CON ACCESIBILIDAD REDUCIDA A INTERNET PARA CITYWALLET.

TRABAJO INSTRUMENTAL DE GRADO
Presentado ante la
UNIVERSIDAD CATÓLICA ANDRÉS BELLO
Como parte de los requisitos para optar al título de
INGENIERO EN INFORMÁTICA

REALIZADO POR	German Javier Pérez
TUTOR EMPRESARIAL	Ing. Félix Armando Fernández Pérez
TUTOR ACADÉMICO	Ing. Biagio Cante.
FECHA	Julio 2017.

DEDICATORIA

A mis padres German Pérez y Silvia González, por el apoyo y los buenos consejos que me han dado.

A mis sobrinos Ethan y Matías que me alegran la vida día a día, los mejores regalos que me han dado.

A mis hermanos Lucero Pérez y Luis Pérez, y a toda mi familia.

AGRADECIMIENTOS

Quisiera agradecer primero a dios por la salud, las buenas oportunidades a lo largo de mi vida y sobre todo por permitirme realizar una de las metas más anheladas en mi plan de vida.

A mis padres, hermanos y sobrinos que están en las buenas y malas con sus consejos y regaños.

A la familia que vas ganando a lo largo de la vida como lo son, Jenny Sayago, Solsire Torres, Graciela Lucena e Israel Osuna. Quiero que sepan que son mis ingenieros favoritos☺.

A los profesores que a lo largo de la carrera nos fueron forjando y exigiendo para ser excelentes profesionales. En el momento que estas en los pupitres no te das cuenta que las exigencias que nos piden son necesarias, pero una vez terminas te das ves lo valioso que fue cada una de sus lecciones. Quisiera extender este agradecimiento Wilmer Pereira, Biagio Cante, Valeria León, Susana García, Gloria Tarrío y a los demás profesores de la escuela de ingeniería informática.

Y por supuesto a la gran familia de CityWallet de la cual me siento orgulloso de pertenecer. Atilana Piñón, Ramón Ginez, Félix Fernández, Liesl Hros, July Marval, Alejandro García, Carlos Unda, Manuel Pacheco, Santiago Bernal, Fernando Marcano y Hernán García. Y a los CityWalleños que están con nosotros por adopción: Gabriela Sotelo, María Unda, Graciela Sosa y Octavio Londoño.

No seremos el equipo más grande, pero si somos el más diverso, tostado y divertido.

¡Muchas gracias a todos!

ÍNDICE

Índice de tablas

Índice de figuras

SINOPSIS

El presente trabajo de grado intitulado" Sistema de pagos a través de un pin único para servicios con accesibilidad reducida a internet para CityWallet" tiene como finalidad permitirle a los usuarios de la plataforma poder enviar o recibir transacciones monetarias por medio de un identificador único. Estas transacciones son posibles a través de datos móviles o SMS en caso de no disponer de los mismos. Adicionalmente, el desarrollo de una aplicación de reportes de interés administrativo de CityWallet y la automatización de recargas por depósitos bancarios dentro de la plataforma. El marco de trabajo implementado durante el proceso de desarrollo se basó en Scrum, siendo necesarios 12 sprints para su culminación. Como resultado se obtuvo una aplicación que permite la transferencia de dinero, consulta de transferencias realizadas y un buzón de mensajes. De igual forma se obtuvo una aplicación web donde se pueden visualizar los reportes de dichas transferencias y de usuarios.

CAPÍTULO I – EL PROBLEMA

1. NECESIDADES DE LA EMPRESA

Citywallet es un start up venezolano que desde el 2015 ofrece a sus usuarios el pago de estacionamientos a través de calcomanías NFC y puntos lectores de NFC. Cada usuario afiliado al servicio puede acceder a los estacionamientos de la red de Citywallet simplemente presentando su calcomanía NFC sobre el lector.

Actualmente esta solución se encuentra implementada en la Torre Xerox, en el Centro Empresarial La Lagunita y próximamente en el Centro Comercial Ciudad Tamanaco. Este servicio ha sido beneficioso para los usuarios reduciendo el uso de efectivo y largas colas de espera tanto en los estacionamientos como en los cajeros automáticos.

Citywallet cuenta con un método de recarga a través de la integración con la plataforma de pagos de Instapago. Con este módulo los usuarios pueden realizar recargas con sus tarjetas de créditos y el monto correspondiente se abona a su cuenta.

Debido a su buena receptividad en el mercado venezolano y a su rápida captación de clientes, Citywallet ha decidido expandirse a otros mercados que requieren manejo de efectivo.

Estos mercados son más dinámicos que los estacionamientos, donde no se cuenta con una conexión directa a internet y tampoco se puede disponer de dispositivos de lectura NFC. Por ejemplo, podemos ver el pago de un servicio de entregas o una carrera de taxi, en este tipo de servicios sería engorroso y poco funcional implementar la solución que posee actualmente Citywallet para el pago de estacionamientos.

2. SOLUCIÓN PROPUESTA

La solución que se propone para la expansión de Citywallet hacia otros mercados es desarrollar una plataforma de pagos a través de un número hexadecimal único que identificará a cada usuario del sistema y estará compuesto de 5 dígitos.

Los clientes de CityWallet dispondrán de una aplicación móvil donde se podrá realizar o recibir pagos. Estos pagos se realizarán de la siguiente forma: el usuario inicia sesión en la aplicación móvil, luego solicita al comercio u otro usuario el identificador único. El usuario ingresa el nombre del receptor, su clave, el monto y un concepto de pago. Una vez realizada la transacción se les notifica a ambos el estado de la misma.

En cuanto a la disponibilidad de internet en los dispositivos se desarrollará una capa de traducción para todo el sistema de datos móviles a mensajes de textos y viceversa. Si una petición de transferencia de fondos o una petición de pago no pudiese realizarse por falta de datos móviles, este se enviará a través de mensajes de texto. Con esta solución se garantiza la alta disponibilidad del servicio en caso de que el usuario no posea datos móviles o se encuentre fuera del rango de cobertura del servicio.

3. OBJETIVO GENERAL

Desarrollar un sistema de pagos a través de un pin único para servicios con accesibilidad reducida a internet para Citywallet.

4. OBJETIVOS ESPECÍFICOS

1. Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
2. Desarrollar un webservice basado en REST para el manejo de peticiones.
3. Desarrollar el núcleo del sistema.

4. Desarrollar aplicación web para gestión administrativa.

5. Desarrollar proceso batch para recargas de depósitos en cuentas bancarias.

6. Desarrollar aplicación móvil para usuarios de Citywallet.

7. Diseñar e implementar un proceso de transferencias de fondos monetarios a través de un pin único.

5. APORTE TECNOLÓGICO

8. Diseñar e implementar una capa de traducción entre SMS y datos móviles.

6. APORTE FUNCIONAL

9. Diseñar proceso de pagos.

7. ALCANCE

7.1 OBJETIVOS ESPECÍFICOS

1. **Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.**

   Se realizará el levantamiento de la información con el personal de área de ventas de la empresa. Se procederá a modelar dicha información para cumplir con los requerimientos solicitados. En base a esto se desarrollará la base de datos no relacional.

2. **Desarrollar un webservice basados en REST para el manejo de peticiones**

   Se desarrollará el webservice que reciban las peticiones

desde la aplicación de web y aplicación de móvil. Estos servicios se desarrollarán bajo la arquitectura REST y será el intermediario entre las aplicaciones y el núcleo del sistema. Los servicios serán publicados a través de un API para la interacción.

3. **Desarrollar el núcleo del sistema.**

   Se desarrollará el núcleo del sistema con la capacidad de recibir y manejar las peticiones de consulta, actualización, eliminación y adición de información recibidas por el webservice. También se deberá realizar el manejo de excepciones.

4. **Desarrollar aplicación web para gestión administrativa.**

   Se realizará el levantamiento de información con el apoyo del departamento de ventas para generar los reportes que requieran. Esta aplicación web deberá generar los reportes de movimientos por usuarios, transacciones fallidas o exitosas de un usuario y recargas realizadas por un usuario.

5. **Desarrollar proceso batch para recargas de depósitos en cuentas bancarias.**

   El proceso de recargas manuales por usuarios puede ser muy tedioso, por lo cual se requiere un proceso que sea capaz de revisar las transacciones realizadas todos los días y verificar si la recarga ya se hizo efectiva al usuario correspondiente. En caso de no haberse realizado se debe agregar dicho monto a la cuenta del usuario. De igual modo se requiere que este proceso verifique si no hay duplicidad de números de transferencias.

6. **Desarrollar aplicación móvil para usuarios de Citywallet**

Los usuarios de CityWallet dispondrán de una aplicación móvil donde una vez iniciada la sesión podrán solicitar o enviar dinero a otros usuarios del sistema a través de un identificador único. De igual forma podrán visualizar los movimientos de dinero realizados a través de la plataforma, gestionar sus datos de perfil, cuentas bancarias y fondos disponibles. Cada cuenta contará con un buzón de mensaje donde se recibirán mensajes del sistema, solicitudes de cobro de otros usuarios o confirmaciones de envío de dinero. Igualmente el usuario podrá realizar peticiones de transferencias de los fondos monetarios en el sistema a sus cuentas bancarias. Si el usuario al momento de enviar dinero a otro usuario no posea acceso a internet se enviará la información a través de SMS. Se implementará un sistema de monitoreo de la aplicación con el cual se podrá detectar cualquier error dentro de ella.

7. **Diseñar e implementar un proceso de transferencias de fondos monetarios a través de un pin único.**

Cada usuario dispondrá de un identificador único. Los usuarios que deseen realizar transferencias a otro usuario deberán introducir el identificador del destinatario, monto, clave y concepto. Una vez realizada esta transferencia se le notificará tanto al emisor como al receptor del estatus de la transferencia. En caso de no disponer datos móviles en el dispositivo dichas transferencias se realizarán por medio de la capa de traducción de datos.

7.2 APORTE TECNOLÓGICO

1. DISEÑAR E IMPLEMENTAR UNA CAPA DE TRADUCCIÓN ENTRE SMS Y DATOS MÓVILES.

Debido a lo intermitente del servicio de internet de Venezuela, se requiere desarrollar una capa de traducción que permita la conversión entre datos móviles y mensajería de texto. Con esta conversión se podrá mantener un servicio altamente disponible para la realización de transacciones. Esta capa será capaz de recibir mensajes de texto y traducirlos a datos que sean entendidos por el webservice para el registro de transacciones.

7.3 APORTE FUNCIONAL

1. Diseñar proceso de pagos

Actualmente Citywallet lleva los procesos de pago a través de calcomanías NFC y lectores NFC. El usuario que desee realizar una transacción a través de la plataforma debe dirigirse con una calcomanía de forma física a los lectores NFC, una vez se encuentre allí debe presentar su calcomanía en el lector NFC y esperar la respuesta de transacción exitosa o no.

Este proceso requiere la presencia física del lector NFC, de la calcomanía y del usuario en el establecimiento para poder efectuar un pago. Por esta razón se diseñará un nuevo proceso de pago que consistirá en que cada usuario dispondrá de un identificador único y una clave secreta de 4 dígitos. El usuario que desee realizar un pago solicitará al receptor su nombre de usuario.

Una vez obtenido el nombre de usuario, el emisor lo introduce en una aplicación móvil junto con el monto de la transacción, concepto y clave secreta. Esta solución no requiere la presencia física del usuario para realizar el pago, puntos de lectura NFC, calcomanías NFC o la inversión de hardware cómo se maneja actualmente.

8. LIMITES

Para el desarrollo del sistema se utilizarán las herramientas ofrecidas por la plataforma de Amazon AWS. Entre las cuales se tienen:

AWS Amazon Dynamodb para el manejo de las instancias de la base de datos no relacionales, esta herramienta permite que la base de datos sea escalables en unos minutos con una capacidad de hardware rentable y redimensionable. (Amazon Dynamo, 2016)

AWS Cognito que permite el inicio manejo de sesiones a aplicaciones web y móviles, al igual que la autenticación de los usuarios. (Amazon Cognito, 2016)

AWS Amazon Mobile Analytics. Permite medir el uso e ingreso a la aplicación móviles. También permite analizar el comportamiento de los usuarios en la navegación de las aplicaciones (AWS Amazon mobile Analytics, 2016)

AWS Amazon Device Farm: Permite la probar las aplicaciones móviles en gran variedad de dispositivos físicos que se encuentran en la nube (Aws Amazon Device Farm,2016)

La aplicación móvil será desarrollada bajo el framework de Ionic 2 con Angular 2 y será para la plataforma Android versión 5 o superior.

La aplicación web se desarrollará bajo el framework de Ionic 2 con angular 2 y deberá ser conectada por medio de los webservices desarrollados en este trabajo instrumental de grado.

El archivo de entrada que correrá el batch debe de ser csv y deberá contener al menos un dato único del usuario para realizar el match en el sistema.

Las peticiones al webservice deben de seguir los estándares de la arquitectura REST y las respuestas del mismo se realizarán por medio de Json.

La traducción de SMS podrá ser realizada siempre y cuando el teléfono posea sistema operativo Android, tenga compatibilidad para la recepción de SMS, tenga el servicio activo y disponible.

9. JUSTIFICACIÓN

El desarrollo de la plataforma de transferencia de pagos entre usuarios viene dada a la necesidad de CityWallet de expandirse a nuevos mercados. Estos nuevos mercados no son posibles atenderlos cubrir debido a costos del hardware a utilizar, lo incomodo de movilizar los equipos actuales y la limitación de señal de internet. También viene dado a la cantidad de efectivo utilizado en el día a día de los venezolanos, la lentitud de las redes para realizar pagos a través de instrumentos bancarios y los escases de puntos de venta en el país.

CAPÍTULO II – MARCO REFERENCIAL

En este capítulo, se describen los fundamentos teóricos que basan el presente trabajo para lograr el completo entendimiento del lector, mencionando las definiciones básicas que abarcan los objetivos principales de investigación y fundamentan el desarrollo realizado.

1. FINTECH

Es una industria compuesta por empresas que usan la tecnología para hacer los servicios financieros más eficientes. Las Fintech, corresponden a un grupo de empresas que se dedican a ofrecer servicios financieros innovadores preferentemente a través de nuevos canales de distribución, mejorando la oferta de los mercados financieros [1].

Una de las principales características de la aparición de las Fintech ha sido su capacidad de innovación para ofrecer nuevos enfoques sobre la cobertura de las necesidades financiera y el clásico modo de operar de la banca.

El Fintech forma parte del modelo de economía colaborativa que ofrece el sector financiero, Según David Igual "la denominada economía colaborativa es un nuevo modelo económico basado en comunidades de personas que, organizadas alrededor de plataformas Fintech, pueden obtener lo que desean las unas de las otras, con o sin intercambio de dinero. Este enfoque rompe con la idea de la propiedad como único camino para disfrutar de productos o servicios, y en cambio, sitúa el concepto de compartir entre iguales como práctica emergente" [2].

Según la Asociación Española de Fintech, las categorías dentro del sector son [3]:

- Asesoramiento/Gestión Patrimonial.
- Finanzas Personales.

- Finanzas Alternativas: préstamos, préstamos colaborativos, compensación, deudas, créditos.
- Inmobiliario.
- Servicios Transaccionales/Divisas.
- Medios de Pagos.
- Pagos entre personas.
- Tecnología para aseguradoras.

2. P2P PAYMENTS

En la actualidad la conexión crece exponencialmente, y con ello las personas conectadas toman cada vez más relevancia que no tiene precedentes. Se habla de redes P2P para definir aquellos procesos en que los usuarios (personas) son los protagonistas de los intercambios en la red, sea para el intercambio de información, de conocimiento, de productos o servicios. De usuario a usuario creando una economía colaborativa [4].

Se habla de economía colaborativa financiera como bloque que reúne a las empresas que poseen modelos de negocios enfocados a descentralizar las finanzas, tales como: bancos de personas a personas, modelos de inversión colaborativos, pagos entre personas, entre otros.

Dentro de este modelo de negocios se encuentran los pagos peer-to-peer (P2P) o bajo el término comúnmente denominado los P2P payments (pagos P2P en su término en español), cuya definición establece que es dinero que puedes enviar a otras personas a través de una App móvil, mensaje de texto (SMS) o correo electrónico a través de una tarjeta de débito, crédito, una cuenta bancaria vinculada, entre otras opciones [5].

3. NFC

Near Field Communication (NFC) o comunicación de campo cercano, es una tecnología basada en chips que fomenta la comunicación entre dos (2) dispositivos sin realizar contacto, actualmente la mayor parte de teléfonos inteligentes o tabletas lo contienen, sin embargo, también existen dispositivos como tarjetas, calcomanías, brazaletes, entre otros, que cuentan con esta tecnología. El chip actúa como un enlace inalámbrico, el cual se activa al entrar en contacto con otro, por lo que no consume energía ni requiere una fuente de poder como una batería. Adicionalmente, cada etiqueta NFC posee un identificador único, basado en el estándar ISO 14443 [6].

La interconexión entre dispositivos ocurre con una frecuencia de 13,56Mhz, el NFC funciona en distancias inferiores a 10cm con velocidades de transmisión entre el rango 106 y 424Kb/s, a nivel de operación es sencilla y simple dado que no necesita licencias administrativas para transmitir los datos [7].

La tecnología NFC puede funcionar en modo activo o pasivo. El modo activo, ocurre cuando ambos chip NFC generan un campo electromagnético para intercambiar datos, y el pasivo, ocurre cuando sólo uno de los dispositivos genera el campo electromagnético, de forma que, el otro aprovecha ese campo para intercambiar la información [7].

4. CLOUD COMPUTING

El cloud computing (o informática en la nube en español) consiste en obtener acceso a servidores, almacenamiento, bases de datos y una amplia gama de servicios de aplicaciones a través de una plataforma de servicios en Internet [8].

Además de esto, proporciona a los desarrolladores y departamentos de TI (tecnología de información) la capacidad de concentrarse en lo que más importa y evitar

tareas como el aprovisionamiento, el mantenimiento y la planificación de capacidad. A medida que ha incrementado la popularidad de la cloud computing, se han desarrollado varios modelos y estrategias de implementación para satisfacer las necesidades de los distintos usuarios. Cada tipo de servicio en la nube y método de implementación le aporta distintos niveles de control, flexibilidad y administración. Entender la diferencia entre la Infraestructura como servicio, la Plataforma como servicio y el Software como servicio, además de las estrategias de implementación disponibles, puede ayudarle a determinar el conjunto de servicios que más se adapta a sus necesidades [9].

La informática en la nube se compone de tres tipos principales, que se denominan comúnmente infraestructura como servicio (IaaS), plataforma como servicio (PaaS) y software como servicio (SaaS). A continuación se resume cada uno de estos tipos:

- Infraestructura como servicio (IaaS): contiene los bloques de creación fundamentales para la tecnología informática (TI) en la nube y proporciona acceso a las características de redes, a los equipos (virtuales o en software dedicado) así como espacio de almacenamiento de datos.
- Plataformas como servicio (PaaS): administran la infraestructura subyacente (normalmente hardware y sistemas operativos) de las compañías, permitiéndoles centrarse en la implementación y la administración de sus aplicaciones.
- Software como servicio (SaaS): proporciona un producto completo que el proveedor del servicio ejecuta y administra. En la mayoría de los casos, el Software como servicio se refiere a aplicaciones de usuario final.

5. SERVICIOS AWS

Amazon Web Services (AWS) o en su término en español "Servicios Web Amazon" son un conjunto de productos globales basados en la nube, incluidas aplicaciones de informática, almacenamiento, bases de datos, análisis, redes, móviles, herramientas para desarrolladores, herramientas de administración, IoT, seguridad y empresariales [10].

5.1 AMAZON DYNAMODB

Amazon DynamoDB o en su término en español "Servicios de Bases de Datos no relacionales de Amazon" es un servicio administrado que facilita las tareas de configuración, utilización y escalado de una base de datos no relacional en la nube. Proporciona capacidad rentable y de tamaño modificable y, al mismo tiempo, administra las tediosas tareas de administración de la base de datos, lo que le permite centrarse en sus aplicaciones y su negocio [11].

5.2 AMAZON COGNITO

Es la herramienta que permite agregar de forma sencilla el registro y el inicio de sesión a sus aplicaciones web y móviles. Amazon Cognito también permite autenticar a los usuarios mediante un proveedor de identidad externo y ofrece unas credenciales de seguridad temporales para obtener acceso a los recursos back-end de la aplicación en AWS o en cualquier servicio que se encuentre detrás de Amazon API Gateway. Amazon Cognito funciona con proveedores de identidad externos que admiten OpenID Connect o SAML y con proveedores de identidad social (como Facebook, Twitter, Amazon). También podemos integrar nuestro propio proveedor de identidad [12].

Mediante esta herramienta, se pueden sincronizar datos en distintos dispositivos de usuario para ofrecerles una experiencia uniforme cuando cambien de un dispositivo a otro o cuando actualicen a uno nuevo. La aplicación puede guardar datos localmente en los dispositivos de los usuarios, lo que permite que las aplicaciones funcionen cuando los dispositivos no tengan conexión y que luego se sincronicen los datos automáticamente cuando vuelvan a estar en línea. En la Figura 1, se muestra como ocurre la administración, autenticación y sincronización de los usuarios entre dispositivos [13].

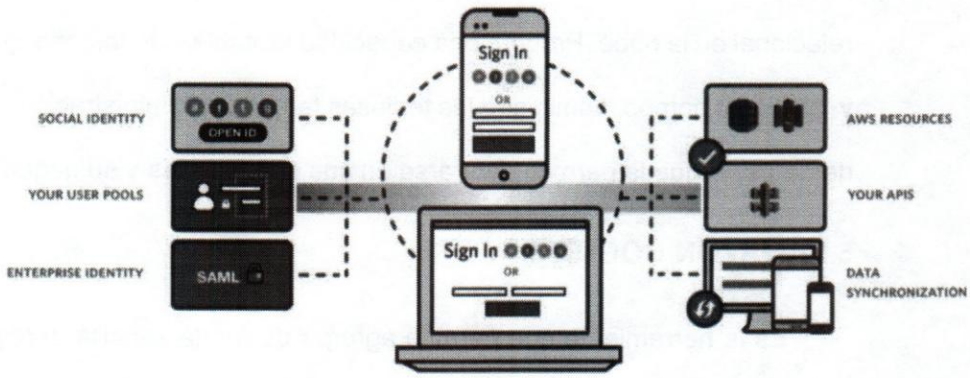

Figura 1. Administración, autenticación y sincronización con Amazon Cognito.

Fuente: Amazon Web Services 2017 [13].

5.3 AMAZON SIMPLE STORAGE SERVICES S3

Es un servicio que permite recopilar, almacenar y analizar datos de forma cómoda y sencilla, independientemente de su formato y a escala masiva. S3 es un almacenamiento de objetos creado para almacenar y recuperar cualquier cantidad de datos desde cualquier ubicación: sitios web y aplicaciones móviles,

aplicaciones corporativas y datos de sensores o dispositivos IoT [14].

5.4 AMAZON DEVICE FARM

Es un servicio de pruebas de aplicaciones que le permite probar sus aplicaciones Android, iOS y web e interactuar con ellas en numerosos dispositivos al mismo tiempo o reproducir errores en un dispositivo en tiempo real. Adicionalmente, permite ver vídeos, capturas de pantalla, logs y datos de desempeño para identificar y solucionar errores antes de publicar su aplicación [16].

Permite a los desarrolladores pueden cargar su aplicación y scripts de pruebas y ejecutar pruebas automatizadas simultáneamente en cientos de dispositivos reales. También permite depurar y reproducir los errores de los clientes deslizando los dedos por la pantalla, haciendo gestos e interactuando con un dispositivo a través de su navegador web [17].

5.5 AMAZON MOBILE ANALYTICS

Es un servicio que permite almacenar, visualizar y comprender los datos de uso de las aplicaciones a escala. Fue diseñado para entregar informes de uso en los 60 minutos posteriores a la recepción de los datos desde una aplicación, permitiendo actuar sobre los datos con mayor rapidez [18].

Su objetivo es escalar al ritmo de la aplicación, lo que le permite recopilar y procesar miles de millones de eventos al día de millones de usuarios. Para usar Amazon Mobile Analytics se debe añadir el SDK para móviles de AWS opcional a la aplicación y publicar mediante el mecanismo de distribución existente (como iTunes Store, Google Play o Amazon Appstore); a partir de

ese momento se inicia automáticamente la recopilación de métricas, la herramienta calcula y actualiza de forma automática las siguientes métricas a medida que se reciben los datos [19]:

- Usuarios activos diarios (DAU), usuarios activos mensuales (MAU) y usuarios nuevos.
- Factor sticky (DAU dividido entre MAU).
- Recuento de sesiones y promedio de sesiones por usuario activo diario.
- Ingresos medios por usuario activo diario (ARPDAU) e ingresos medios por usuario activo diario que paga (ARPPDAU).
- Retención de 1, 3 y 7 días, y retención de 1, 2 y 3 semanas.
- Eventos personalizados.

6. ARQUITECTURA DE SOFTWARE

Se define como la estructura del sistema, que expresa un detalle general o "macro" de los principales elementos que componen el mismo, su conducta y su interacción para alcanzar el objetivo por el cual fue creado el sistema [20].

6.1 Back-End

6.1.1 JAVA

Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general [21].

Una de las principales características de Java, es que es un lenguaje

independiente de la plataforma, es decir, un programa hecho en Java podrá funcionar en cualquier computadora del mercado. Esto lo convierte en una ventaja significativa para los desarrolladores de software, dado que antes debían hacer un programa para cada sistema operativo, por ejemplo Windows, Linux, Apple, entre otros; esto es posible porque se ha creado una Máquina de Java para cada sistema que hace de puente entre el sistema operativo y el programa de Java y posibilita que este último se entienda perfectamente [22].

6.2 Front-end

6.2.1 Framework

El concepto framework se emplea en muchos ámbitos del desarrollo de software y aplicaciones web. En general, con el término framework se hace referencia a la estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación; un framework se considera como una aplicación genérica incompleta y configurable a la que se añaden piezas para construir la aplicación deseada [23].

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones; dentro de las ventajas de usar frameworks tenemos [24]:

- Facilita la colaboración.
- El desarrollador no necesita plantearse una estructura global

de la aplicación, sino que el framework le proporciona un esqueleto que hay que "rellenar".

6.3.1 Angular 2

Es un framework que permite separar: las capas de presentación, lógica y componentes de una aplicación.

Mediante Angular 2 se pueden crear aplicaciones web tan robustos como sea necesario, es decir, se puede crear desde un blog hasta un portal con tráfico similar a Amazon. Con esta herramienta se puede ampliar el HTML de la forma que se necesite, permitiendo crear etiquetas personalizadas que encapsulan el contenido en HTML, el comportamiento en JavaScript y el estilo en CSS, puesto que el framework posee las siguientes especificaciones [25]:

- HTML Imports: usa en la distribución de las librerías de componente web, esta tecnología posibilita importar un código HTML en otro HTML.
- Custom Elements: esto conlleva a la creación de etiquetas semánticamente correctas para la función que se requiera.
- Templates: posibilitan la creación de una estructura visual del componente web, sin afectar la página, eso quiere decir que no es incluido en el DOM hasta ser procesado por cualquier elemento, haciendo este una copia para incluirla posteriormente en el DOM.

6.4.1 Ionic 2

Ionic 2 es un framework para el desarrollo de aplicaciones híbridas pensado para móviles y tablets, el cual fue evolucionado para ser capaz de implementar aplicaciones web y aplicaciones de escritorio multiplataforma. Su característica fundamental es que usa por debajo Angular 2 y una cantidad de componentes enorme, que facilita mucho el desarrollo de las aplicaciones [26].

Esta herramienta capaz de crear de aplicaciones, permite obtener resultados de una manera rápida y con una menor inversión económica, ya que permite crear aplicaciones para distintas plataformas móviles con una misma base de código [27].

6.3 Cliente-servidor

En el mundo de TCP/IP las comunicaciones entre computadoras se rigen básicamente por lo que se llama modelo Cliente-Servidor, éste es un modelo que intenta proveer usabilidad, flexibilidad, interoperabilidad y escalabilidad en las comunicaciones.

Desde el punto de vista funcional, se puede definir la computación Cliente/Servidor como una arquitectura distribuida que permite a los usuarios finales obtener acceso a la información en forma transparente aún en entornos multiplataforma [28].

Las características básicas de una arquitectura Cliente/Servidor son [29]:

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos compartidos. El proceso del cliente

proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, módems, etc.

- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco y input-output devices.
- Los servidor puede dar servicio a muchos clientes, regulando su acceso a recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son éstos los que hacen peticiones de servicios a los servidores. Estos últimos tienen un carácter pasivo ya que esperan las peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicio.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre la misma. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente/Servidor. La escalabilidad

horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

6.4 Web Services

El término web services (o servicios web en español) hace referencia a la tecnología que permite que las aplicaciones se comuniquen en una forma que no depende de la plataforma ni del lenguaje de programación. Un servicio web es una interfaz de software que describe un conjunto de operaciones a las cuales se puede acceder por la red a través de mensajería XML estandarizada. Usa protocolos basados en el lenguaje XML con el objetivo de describir una operación para ejecutar o datos para intercambiar con otro servicio web. [30].

De una manera más clara y resumida se podría decir que un web service es una función que diferentes servicios o equipos utilizan; es decir, solo se envían parámetros al servidor (lugar donde está alojado el web service) y éste responderá la petición. Entre algunas que se manejan de utilizar servicios webs en las aplicaciones destacan las siguientes [31]:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su

funcionamiento.

- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.

- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

- Permiten la interoperabilidad entre plataformas de distintos fabricantes por medio de protocolos estándar y abiertos. Las especificaciones son gestionadas por una organización abierta, la W3C, por tanto no hay secretismos por intereses particulares de fabricantes concretos y se garantiza la plena interoperabilidad entre aplicaciones.

6.4.1 API

Un API es un conjunto de comandos, funciones, protocolos y objetos que los programadores pueden usar para crear software o interactuar con un sistema externo. Proporciona a los desarrolladores comandos estándar para realizar operaciones comunes para que no tengan que escribir el código desde cero.

Los API están disponibles tanto para sistemas operativos de escritorio como móviles. Los API de interfaz, por ejemplo, proporciona a los desarrolladores controles y elementos de usuario, como ventanas, barras de desplazamiento y cuadros de diálogo. También proporciona comandos para acceder al sistema de archivos y realizar operaciones

de los mismos, a su vez, como crearlos y eliminarlos. Además, incluye comandos de red que se pueden utilizar para enviar y recibir datos a través de una red local o de Internet.

6.5 Formato de envío y recepción de información

En la actualidad, XML y JSON son los tipos de formatos para el envío y recepción de la información a través de los web services, en el presente trabajo se utilizó JSON.

6.5.1 JSON

JSON (JavaScript Object Notation o notación de objeto de JavaScript en español) es un formato liviano de intercambio de data, de fácil lectura y escritura, y simple para las máquinas de generar y procesar. Está basado en el lenguaje de programación de JavaScript y es un formato de texto independiente de cualquier lenguaje de programación el cual está construido bajo dos estructuras: 1) Una colección de pares de nombre y valor; y 2) Una lista ordenada de valores. La forma del JSON viene definida por: un objeto, que es una lista desordenada de pares de nombre y valores, el cual comienza con el símbolo "{" y termina con "}", cada nombre es seguido por ":" y cada par de nombre y valor es separado por ",", [32]. En la Figura 2 se muestra un ejemplo ilustrativo.

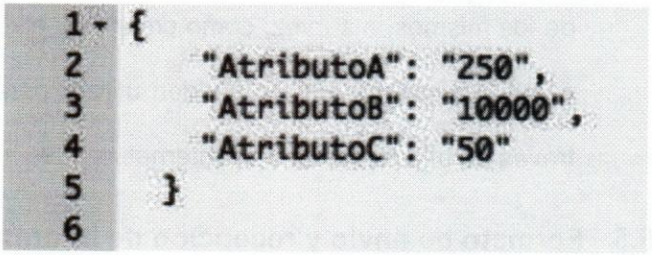

Figura 2. Ejemplo de JSON

Fuente: Elaboración propia

7. CRIPTOGRAFÍA

La palabra Criptografía proviene del griego "kryptos" que significa oculto, y "graphia", que significa escritura, y consiste en una técnica (o el conjunto), que tratan sobre la protección o el ocultamiento de la información frente a observadores no autorizados. Entre las disciplinas que engloba se encuentran [33]: la Teoría de la Información, la Complejidad Algorítmica y la Teoría de números o Matemática Discreta, que trata de estudiar las propiedades de los números enteros.

Mediante la criptografía, la información puede ser resguardada contra el acceso no autorizado, su interceptación, su modificación y la inserción de información extra; o puede ser usada para modo anticipado para prevenir el acceso y uso no autorizado de los recursos de una red o sistema informático y para prevenir a los usuarios la denegación de los servicios a los que sí están permitidos [34]. En la actualidad, la criptografía es la metodología empleada para proveer la seguridad de las redes telemáticas, incluyendo la identificación de entidades y autenticación, el control de acceso a los recursos, la confidencialidad de los mensajes transmitidos, la integridad de los mensajes y su no repudio [35].

La criptografía se divide en dos ramas, clave privada o simétrica y clave pública o asimétrica [36]:

La criptografía de clave privada o simétrica, se refiere al conjunto de métodos que permiten una comunicación segura entre las partes siempre que, con anterioridad, se intercambie la clave correspondiente, que se denomina clave simétrica. La simetría se refiere a que las partes tienen la misma llave, tanto para cifrar como para descifrar.

La criptografía de clave pública o asimétrica, también denominada RSA por las siglas de los apellidos de sus inventores Rivest Shamir y Adelman, es por definición aquella que utiliza dos claves diferentes para cada usuario, una para cifrar que se llama clave pública y otra para descifrar que es la clave privada. El nacimiento de la Criptografía asimétrica ocurrió como resultado de la búsqueda de un modo más práctico de intercambiar las llaves simétricas.

7.1 MD5

Es un algoritmo que proporciona un código asociado a un archivo o un texto concretos. De esta forma, a la hora de descargar un determinado archivo, el código generado por el algoritmo, también llamado hash, viene "unido" al archivo [37]. Los requisitos que deben cumplir las funciones hash son [38]:

- Imposibilidad de obtener el texto original a partir de la huella digital.
- Imposibilidad de encontrar un conjunto de datos diferentes que tengan la misma huella digital (aunque como hemos visto anteriormente es posible que este requisito no se cumpla).
- Poder transformar un texto de longitud variable en una huella de tamaño fijo (como el SHA-1 que es de 160bits).
- Facilidad de empleo e implementación.

El MD5 inicia completando el mensaje a una longitud congruente en módulo 448 mod 512, es decir, la longitud del mensaje es 64 bits menos que un entero múltiplo de 512. El relleno consiste en un bit en 1 seguido por cuentos bits en 0 sean necesarios. La longitud original del mensaje es almacenada en los últimos 64 bits del relleno.

Adicionalmente se inicializa, con un valor fijo, un buffer de 128 bits. Este buffer puede verse como 4 registros de 32 bits (A,B,C,D) y son inicializados con los siguientes valores hexadecimales: A=67452301; B=EFCDAB89; C=98BADCFE; D=10325476.

Durante varias rondas de procesamiento el algoritmo toma bloques de 512 bits de la entrada y los mezcla con los 128 bits del buffer. Este proceso es repetido hasta que todos los bloques de entrada han sido consumidos. El valor resultante en el buffer es el hash del mensaje [38].

8. DISPOSITIVOS MÓVILES

8.1 ANDROID

Durante el transcurso del año 2008, Google lanzó la primera versión del sistema operativo libre denominado "Android", desarrollado como plataforma abierta para implementarse en teléfonos celulares, y gracias a su código abierto, el mismo puede ser modificado y mejorado de acuerdo a las diversas necesidades de los usuarios [39].

Android es un paquete de software que contiene un sistema operativo, middleware, interfaz de usuario y una serie de aplicaciones, todo ello para ser utilizado sobre dispositivos móviles.

8.2 IOS

iOS es un sistema operativo para dispositivos móviles desarrollado por la empresa Apple, el cual se distribuye ya instalado de forma exclusiva en los dispositivos comercializados por esta empresa, no permitiéndose la instalación de este sistema operativo en ningún otro tipo de dispositivo.

El sistema operativo IOS vio la luz en Junio del año 2007 con la salida al mercado del primer teléfono móvil de Apple, el iPhone, un teléfono inteligente o smartphone en inglés, que es un teléfono con capacidades de computadora de bolsillo ya que permite la instalación de aplicaciones informáticas. Posteriormente Apple incluyó este mismo sistema operativo en sus dispositivos iPod Touch, un ordenador de bolsillo sin capacidad de teléfono que fue lanzado al mercado en Septiembre de 2007. En Enero de 2010 también se incluyó este sistema operativo en los dispositivos iPAD, la tableta de Apple y en los iPAD Mini, una tableta con una pantalla más reducida, a finales del año 2012 [40].

CAPÍTULO III – MARCO METODOLÓGICO

1. SCRUM

Scrum es un efectivo enfoque en la gerencia de proyectos que fue pensado para desarrollo de software, pero que tiene aplicabilidad en otras disciplinas incluyendo manufactura y servicios. La orientación de Scrum es sencilla y está enmarcada en el trabajo en equipo, la comunicación y la auto-organización.

Las bondades de Scrum radican en que se crea balance entre aquellos solicitando el trabajo y los equipos produciendo los resultados. Además, con Scrum se establece una línea bien demarcada entre los clientes y los usuarios. Por si fuera poco, Scrum también construye un ambiente de trabajo que empodera a todos los equipos de

una organización y/o a todos los miembros de un equipo.

No en vano muchas famosas startups deben su éxito a haber aplicado las reglas de Scrum. Por ejemplo, Zappo´s, el gigante e-comerce de ropa y zapatos, desarrolló su primera propuesta de producto en tan sólo 12 semanas [41]. El secreto de su éxito radicó en dividir los esfuerzos en períodos de una semana con objetivos concretos. Estos períodos, son lo que Scrum denomina sprints. La misma filosofía y la misma división de tiempo se aplicaron en CityWallet.

Otro gigante que ha probado el éxito de Scrum es Spotify. Para mantenerse adelante de Apple con iMusic, Google con Google Play Music All Access, entre otros; Spotify se divide en pequeños grupos que se manejan como startups individuales y no como departamentos. En consecuencia, la frecuencia en la entrega de nuevos productos o actualizaciones que presenta Spotify es muchísimo más rápida que la de compañías con largas cadenas de aprobación o burocracias. Scrum le da vida y autonomía a los equipos de trabajo para que sean estos los que diseñen la estrategia para alcanzar sus objetivos y, además, sorteen los eventuales bloqueos que puedan presentarse [42].

Con base en lo previamente expuesto se puede decir que haber trabajado con Scrum impulsó a CityWallet a desarrollar en tiempo record varias versiones de la aplicación móvil; variaciones importantes en la arquitectura de la plataforma transaccional; iteraciones en el modelo de negocio y además, mejoras sustanciales en el diseño de los cajetines lectores ubicados en los estacionamientos. Todo esto con un pequeño equipo de 6 personas.

Para garantizar la eficiencia del trabajo, Scrum establece varios pilares, procesos y técnicas que describiremos a continuación.

1.1 Pilares

Scrum es un marco de trabajo que empodera a equipos para que aborden problemas complejos con alta productividad y calidad. Para ello, el trabajo, la documentación, los reportes y las tareas deben cumplir tres principios:

- Transparencia: los aspectos más relevantes del trabajo deben ser visibles para los responsables de los resultados. Para que la transparencia realmente exista, es importante que el equipo determine parámetros para esquematizar la información de forma que sea comprensible por todos. Un concepto que toma gran importancia es el de las tareas "hechas"; consecuentemente, estas tareas deberán cumplir con una serie de requisitos para considerarse como culminadas y con la calidad exigida.
- Inspección: el progreso del desarrollo debe pasar por varias revisiones para garantizar que el ritmo de trabajo no se vea afectado por bloqueos que comprometan la consecución de los objetivos. Estas inspecciones son planificadas y constantes y no deben entorpecer el trabajo.
- Adaptación: es importante comprender cómo se están desarrollando las tareas para detectar de forma temprana potenciales desviaciones o bloqueos. Esto con la idea de ajustar el plan y alcanzar los objetivos planteados [43].

1.2 Eventos

Para poder inspeccionar y adaptar los sprints, Scrum establece varios sucesos o reuniones con duración limitada para revisar el avance del trabajo. La idea es que estos eventos no entorpezcan las labores de desarrollo, sino que sean reuniones dinámicas y eficientes que brinden un diagnóstico de grupal de

las labores.

Todos los eventos suceden alrededor de Sprint, que no es más que el período del tiempo en el cual se supone que el equipo de desarrollo alcanzará metas determinadas. La duración de los sprints varía dependiendo de cada empresa.

Los eventos que establece Scrum son:

- Planificación del sprint: en este evento se determinan los objetivos del Sprint; estos objetivos están directamente asociados con el incremento del producto, es decir, el producto final se desglosa en piezas más pequeñas que se desarrollarán de distintos momentos [44].
- Scrum diario: es una reunión corta y diaria que tiene como fin entender si el ritmo del desarrollo va acorde a la planificación establecida. Durante esta junta se examinan las acciones que el día anterior contribuyeron positivamente con el trabajo; también se determina qué se hará durante ese día para alcanzar las metas y si hay algún impedimento frenando las acciones del equipo [45].
- Revisión del sprint: el evento se celebra al final de cada sprint. En la reunión se junta al equipo de desarrollo con el Maestro Scrum y se hace un cierre formal de todas las tareas. El resultado de esta reunión arroja un estatus claro del desarrollo del producto. Los incrementos registrados en ese sprint se suman a todas las funcionalidades previamente hechas.
- Retrospectiva del sprint: en esta ocasión, el equipo se auto inspecciona en compañía del Maestro Scrum. El objetivo es entender:
- La calidad e interrelación entre las personas, los procesos y las

herramientas.

- Determinar cuáles tareas y/o elementos fueron los más exitosos y cuáles tienen oportunidad de mejora.
- Implementar las mejoras en el próximo sprint [46].

1.3 Artefactos

Este componente de Scrum representa cada tarea o funcionalidad que debe desarrollarse para completar el producto final. Los artefactos son [47]:

- El backlog de producto: es un listado de todo lo que el producto pudiera necesitar. Pueden tener la forma de funcionalidades específicas, asignaciones o tareas. Esta una lista que muta con el tiempo, pues se nutre de la retroalimentación de cliente, del contacto con los usuarios y del criterio de dueño del producto, quien es el responsable de mantener esta lista actualizada y eficiente a efectos conseguir los objetivos planteados por quienes demanden el producto final
- El backlog del sprint: esta lista contiene las tareas y/objetivos del Sprint y la misma se alimenta del Backlog de Producto. La lista se conforma con la colaboración activa del equipo de desarrollo y del maestro Scrum, siendo este último el responsable de hacer que el trabajo del equipo de desarrollo fluya hacia los objetivos.

1.4 Equipo

El equipo Scrum se conforma por El Dueño del Producto, el Equipo de Desarrollo y el Maestro Scrum, todos, deben contar con dos características indispensables: ser auto-organizativos e interfuncionales. Esta combinación es una apuesta directa al auto-liderazgo como base de la eficiencia. A continuación

se describen sus características [48]:

- El Dueño del Producto: es la mente que maqueta el producto final y, por lo tanto, es quien tiene la capacidad de fragmentarlo en las pequeñas partes que construyen el Backlog de Producto. El Dueño del producto interactúa con otros agentes interesados en el producto, como los clientes y los usuarios.
- El Equipo de Desarrollo: conformado por un grupo de profesionales de distintas áreas. Este equipo tiene la tarea de construir el producto, basándose en su propia visión para esquematizar los sprints de trabajo. Su objetivo es completar la mayor cantidad de tareas posibles durante el sprint.
- El Maestro Scrum: esta figura apoya al Equipo de Desarrollo en la consecución de sus objetivos. Puntualmente, el Maestro Scrum se dedica a asegurar que las tareas puedan completarse en el tiempo determinado y con la calidad exigida. El Maestro Scrum está al servicio del Equipo de Desarrollo.

En el caso de CityWallet, El Ing. Ramón Ginéz fue quien asumió el rol del Dueño del Producto, por otro lado, el Ing. Félix Fernández cumplió el papel del maestro Scrum.

2. MARCO DE TRABAJO ÁGIL DE CITYWALLET

- En CityWallet se trabaja con el framework de trabajo explicado anteriormente, pero con una ligera variación. La organización de las actividades a realizar no se hace a través de Historias de Usuarios si no por medio de épicas, hitos y actividades que se definen como:
- Épica: Es el resultado tangible de un grupo de hitos de un tamaño adecuado

para ser gestionada con los principios y técnicas ágiles con estimación y seguimiento cercano. [35].

- Hito: Se utiliza para rastrar el proceso hacia una épica. Estos marcan puntos específicos de progreso en la planificación de desarrollo, deben ser medibles y debes permitir monitorear la evolución del producto [36]
- Actividad: Son las tareas específicas a realizar para completar un hito. Estas tareas son de corto tiempo y deben estar estructuradas en un orden progresivo y lógico. Las actividades son monitoreadas en cada reunión diaria.

2.1 Mapa

- **Sprint 1**
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.
    - Desarrollar el núcleo del sistema.

- **Sprint 2**
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.
    - Desarrollar el núcleo del sistema.

- **Sprint 3**
    - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.
    - Desarrollar el núcleo del sistema.
    - Diseñar proceso de pagos.
    - Diseñar e implementar un proceso de transferencias de fondos monetarios a través de un pin único.

- **Sprint 4**
    - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.
    - Desarrollar el núcleo del sistema.
    - Diseñar proceso de pagos.
    - Diseñar e implementar un proceso de transferencias de fondos monetarios a través de un pin único.

- **Sprint 5**
    - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.
    - Desarrollar el núcleo del sistema.

- **Sprint 6**
    - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.
    - Desarrollar el núcleo del sistema.

- **Sprint 7**
    - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
    - Desarrollar aplicación móvil para usuarios de Citywallet.
    - Desarrollar un webservice basado en REST para el manejo de peticiones.

- Desarrollar el núcleo del sistema.

- **Sprint 8**
  - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
  - Desarrollar aplicación móvil para usuarios de Citywallet.
  - Desarrollar un webservice basado en REST para el manejo de peticiones.
  - Desarrollar el núcleo del sistema.
  - Diseñar e implementar una capa de traducción entre SMS y datos móviles.

- **Sprint 9**
  - Diseñar e implementar una base de datos que se adapte a los requerimientos del sistema.
  - Desarrollar aplicación móvil para usuarios de Citywallet.
  - Desarrollar un webservice basado en REST para el manejo de peticiones.
  - Desarrollar el núcleo del sistema.

- **Sprint 10**
  - Desarrollar aplicación web para gestión administrativa.

- **Sprint 11**
    - Desarrollar aplicación web para gestión administrativa.
    - Desarrollar proceso batch para recargas de depósitos en cuentas bancarias.

CAPÍTULO IV – DESARROLLO

1. INTRODUCCIÓN

El desarrollo del Trabajo Instrumental de Grado fue realizado en un ciclo de 11 sprint, cada uno de 2 semanas para un total de 22 semanas. El inicio de este proyecto se realizó con una primera fase que se nombró como sprint 0

2. SPRINTS

2.1 Sprint 0

Al iniciar este Sprint, se realizó una serie de reuniones con el SCRUM master, el gerente del producto y el arquitecto de la compañía. En estas reuniones se acordó la forma en la cual serían desarrollados los objetivos específicos planteados en este trabajo instrumental de grado, la arquitectura a implementar y los frameworks a utilizar.

Luego de las reuniones con el arquitecto de software se definió el uso de los servicios de AWS para la plataforma a desarrollar. Los motivos principales para el uso de AWS fueron:

- Crédito monetario que posee Citywallet en estos servicios.

- Alta disponibilidad y escalabilidad.
- Experiencia del equipo de trabajo en el área.
- Fácil integración con todos los servicios prestados por AWS.
- Fácil mantenimiento.

Los servicios de AWS a utilizar son los siguientes:

- AWS Elastic Beanstalk.
- Amazon Dynamodb.
- AWS Cognito.
- Amazon Farm Devices.
- Amazon Mobile Analytics.

Para la arquitectura de este proyecto se definió un servidor principal con el manejo del servicio de AWS Elastic Beanstalk. Una de las ventajas principales de este servicio es su alta compatibilidad con aplicaciones web desarrolladas con Java como en este caso, de igual modo es capaz de manejar el balanceo de carga y escalado automático de recursos en caso de necesitarlos en algún aumento de tráfico de usuarios. El servicio de Amazon Dynamodb fue utilizado para el manejo de la base de datos noSQL.

Quedando estructurada de la siguiente forma:

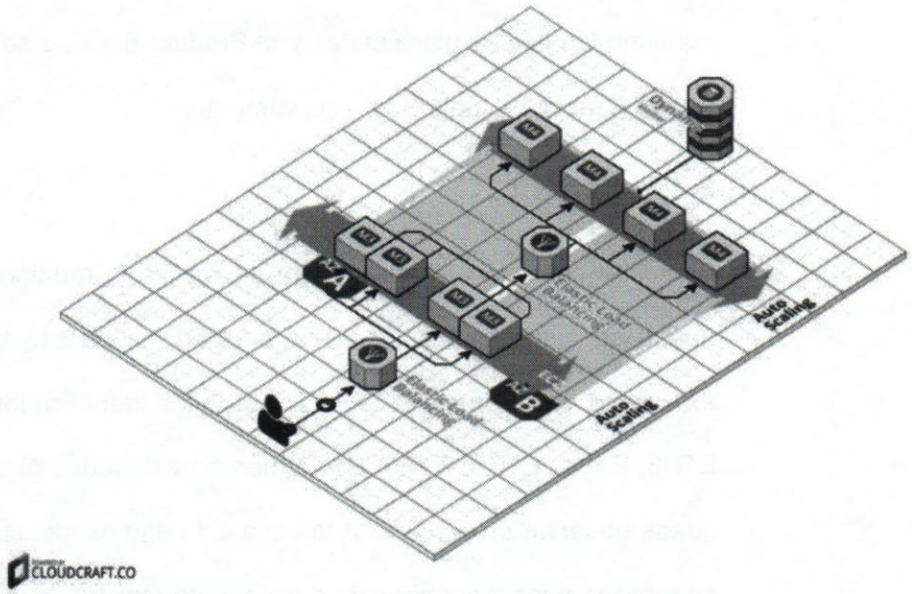

Figura 3 Arquitectura de la solución

Para el desarrollo de la aplicación de web administrativa y aplicación móvil se especificó el uso del framework Angular 2 y Ionic 2. Una de las grandes ventajas de estos frameworks es la posibilidad de desarrollar aplicaciones híbridas para sistemas operativos como iOS y Android con apariencia nativa.

Posterior a la definición de la arquitectura y herramientas a utilizar se realizó la planificación inicial de acción agrupando los procesos por su naturaleza. Una vez agrupados estos procesos se les otorga el nombre de épicas, los cuales representan un momento del proyecto y son alcanzados luego de cumplir una serie de hitos.

Al definir cada hito y a que épica pertenece se procedió a la definición de las actividades requeridas para cumplir cada hito. Finalmente, se organiza el Product Backlog teniendo en cuenta que cada sprint debe abarcar un hito o meta

funcional de forma completa, es decir, culminarla en dicho sprint. La tabla que contiene las épicas planificadas y el Product Backlog se puede visualizar en el anexo A tabla 1 y tabla 2 respectivamente.

2.2 Sprint 1

Al comienzo de este sprint se realizó la reunión con el master Scrum donde se seleccionó del product backlog los hitos a trabajar en este sprint expuestos en el anexo A tabla 3. Los hitos identificados como E1H1, E1H2, E1H3, E1H4, E1H5, E1H5, E1H6 fueron trasladados al sprint backlog como se puede observar en el anexo A la tabla 2. Luego de identificar los hitos a trabajar se realizaron los diagramas de procesos de registro de usuario, envío de código de validación a través de correo electrónico y reenvío de código de validación a través de correo electrónico. Estos diagramas se pueden observar en el anexo B figura 3, figura 4 y figura 5 respectivamente.

Se seleccionó el hito identificado dentro del sprint backlog como E1H1. La primera tarea de este hito fue identificada como E1H1A1 y consistió en el diseño de los atributos utilizados en AWS Cognito para el manejo de usuarios, esta definición puede ser visualizada en el anexo D tabla 19. A partir de esta definición se procedió a ejecutar la actividad E1H1A2 que consistió en la creación y configuración del pool de usuarios con sus respectivos atributos para el registro de usuario. Posteriormente se realizaron las actividades E1H2A1 y E1H2A2 en las cuales se configuró el servicio para el envío del código de validación de cuentas y se integró el diseño del correo electrónico que recibirá el usuario. En cuanto al servicio de reenvío de código de validación de cuentas se ejecutaron las actividades identificadas como E1H3A1 y E1H3A2 culminando así la configuración del registro de usuario y cumpliendo con los hitos E1H1, E1H2 y

E1H3 respectivamente. Una vez culminada esta configuración, se procedió a documentar las credenciales que posteriormente serían necesarias para acceder a este pool de usuarios de AWS cognito.

Se inició el desarrollo del hito E1H4, compuesto por las actividades E1H4A1 y E1H4A2. Se realizó el diseño de la ventana de registro de usuario, luego se procedió a la estructura del HTML y CSS necesario para estilizar dicha ventana teniendo en cuenta los atributos definidos previamente tal como se observa en el anexo E figura 20. Una vez finalizado se realizó la funcionalidad de la ventana y la integración con la librería de AWS cognito para realizar la comunicación con el servicio de registro de usuarios.

Para la validación de cuentas de usuario se procedió a realizar las actividades identificadas como E1H5A1 y E1H5A2. Una vez desarrollada dicha ventana, como se aprecia en el anexo E figura 21, se procedió a la integración con la librería de AWS Cognito para establecer la comunicación con el servicio correspondiente. De esta forma el usuario puede ingresar el código recibido vía correo electrónico y concluir el registro de la cuenta. En el caso que el usuario no recibiera el correo electrónico con el código de validación, se desarrollaron las actividades E1H6A1 y E1H6A2 con las cuales el usuario puede solicitar el reenvío del código de validación de cuenta como se observa en el anexo E figura 22. Una vez concluidas estas actividades se cumplen con los hitos E1H5 y E1H6.

2.3 Sprint 2

En la apertura de este sprint se seleccionaron los hitos del product backlog expuestos en el anexo A tabla 4 identificados como E1H7, E1H8, E1H9 con sus respectivas actividades como se observa en el anexo A tabla 3 del sprint

backlog. Luego de identificar los hitos y sus actividades se realizaron los diagramas de procesos de inicio de sesión del usuario y reinicio de claves de acceso para usuarios no autenticados. Estos diagramas se pueden observar en el anexo B figura 6 y figura 7 respectivamente.

Para la autenticación de usuarios a través de la aplicación móvil se seleccionaron las actividades identificadas como E1H7A1 y E1H7A2. Se tomó en cuenta las políticas de acceso de CityWallet, las cuales dicen que un usuario podrá realizar la autenticación a través de un nombre de usuario o correo electrónico. Una vez culminada esta ventana con sus respectivos atributos, validaciones y estilos, como se aprecia en el anexo E figura 23, se procedió a la integración con la librería de AWS Cognito para establecer la comunicación con el servicio correspondiente. Con esta integración el usuario puede realizar el ingreso a la plataforma de CityWallet.

El siguiente paso fue la realización del hito E1H8, comenzando con la actividad E1H8A1 en la cual se realizó el modelo de email que recibirá el usuario al solicitar el reinicio de contraseña. Este correo posee los logos de la compañía, un mensaje instructivo y un código validador que permite el cambio de contraseña. Luego se procedió a ejecutar la actividad E1H8A2 que consiste en la configuración en AWS Cognito para enviar los correos con el código validador una vez el usuario lo solicite y así permitir a los usuarios registrados reiniciar su contraseña de coincidir este código.

Para finalizar el sprint, se efectuó el hito E1H8. Este hito está compuesto de las actividades identificadas como E1H8A1 y E1H8A2. Se procedió a realizar el HTML y el estilo de dicha ventana tomando en cuenta que la librería de AWS Cognito requiere el nombre de usuario o correo electrónico y código de validación

tal y como se observa en el anexo E figura 24. Finalmente se le agregó funcionalidad a la ventana y se integró con las librerías de AWS Cognito para realizar la comunicación entre el pool de usuarios de Citywallet y la aplicación móvil.

2.4 Sprint 3

En la planificación de este sprint se comenzó con el desarrollo del núcleo del sistema y la definición de la base de datos no relacional, los hitos a realizar fueron identificados como E2H1, E2H2, E2H3 con sus respectivas actividades como se observa en anexo A tabla 5. Luego de identificar los hitos y sus actividades se realizó la definición de las tablas users y notificationChannels con sus respectivos atributos, estas definiciones se pueden observar en el apéndice C tabla 14 y tabla 15 respectivamente.

Luego de la definición se realizó la actividad E2H1A1, que consiste en la creación de la tabla users. El siguiente paso fue realizar el manager de esta tabla, el cual maneja la lógica de acceso a la tabla, las validaciones necesarias para mantener la consistencia de la misma y el manejo de excepciones en caso de ocurrir algún error o inconsistencia en el proceso. Este manager tiene como nombre user Manager, esta actividad fue identificada como E2H1A2.

Para concluir con la épica E2H1 se diseñó el API llamado users, este servicio es el encargado de recibir, verificar y enviar al manager correspondiente los atributos enviados por AWS cognito una vez las cuentas creadas sean verificadas, la definición de este endpoint se puede observar en el anexo G figura 37.

La plataforma de CityWallet requiere la comunicación con los dispositivos

móviles a través de notificaciones push. Debido a esto se realizó la actividad E2H2A1 que consiste en la creación de la tabla notificacionChannels, en esta tabla se almacenó los datos necesarios de cada dispositivo para realizar la comunicación a través de las notificaciones push. De forma inmediata inició la actividad E2H2A1, la cual consiste en la creación del manager llamado notificacionChannels. Este manager se encarga de manejar la información recibida a través del respectivo endpoint, validaciones y manejo de excepciones. De igual forma se desarrolló la actividad E2H2A3 que por medio del método post recibirá del dispositivo el número único de identificación y que tipo de sistema operativo posee que será necesario para el envío de notificaciones push. La definición del API notificationChannels puede observarse en el anexo G figura 38 y el diagrama de proceso en el anexo B figura 8.

Para el cumplimento del hito E2H3 se diseñó el API que lleva por nombre users por el cual a través del método get la aplicación móvil solicitará los datos de la cuenta del usuario, este método puede observarse en el anexo G figura 39 y su diagrama de proceso en el anexo B figura 9.

2.5 Sprint 4

En la reunión de planificación de este sprint se trasladaron al sprint backlog los hitos identificados como E2H4, E3H1, E3H2, E3H3 como se observa en el anexo A tabla 7.

Se tomó el hito E2H4 el cual consiste en mostrar al usuario a través de la aplicación móvil los datos de su cuenta como lo son nombre, apellido y nombre de usuario. Para el cumplimiento de la misma se realizaron las actividades

E2H4A1, E2H4A y E1H4A. Primero se diseñó el HTML y el estilo de ventana que lleva por nombre dashboard, esta ventana es la primera interacción del usuario luego de tener un inicio de sesión exitoso. Posteriormente se realizó toda la funcionalidad de la ventana para mostrar los datos solicitados por el usuario y finalmente se procedió a realizar la comunicación con el endpoint previamente definido en el hito E2H3. Esta ventana puede observarse en el anexo E figura 25

En este sprint se dio inicio a realizar el aporte funcional de este Trabajo Instrumental de Grado, identificado como la épica E3 y compuesto por los hitos E3H1, E3H2 y E3H3.

Los usuarios de la plataforma de CityWallet deben poder realizar transacciones de dinero entre cuentas. Estas transferencias se harían a través de un número hexadecimal de 5 dígitos que identificaría a cada usuario y sería llamado como pin único de usuario. En un estudio de mercadeo realizado por CityWallet se llegó a la conclusión que identificar un usuario por medio de un número hexadecimal sería poco amigable y difícil de recordar, por ello se realizó el cambio de un número hexadecimal de 5 dígitos a un nombre de usuario único en la plataforma. Este nombre de usuario estaría compuesto por al menos 3 caracteres y un máximo de 60 caracteres.

Para la realización de esta épica se identificaron los entes participantes en la transferencia de fondos monetarios. Estos participantes son dos usuarios de la plataforma de CityWallet. De la misma forma se identificó las acciones que los usuarios pueden realizar en la plataforma, obteniendo como resultado:

- Enviar dinero a un usuario por medio de la plataforma de CityWallet.
- Solicitar dinero a un usuario por medio de la plataforma de CityWallet.

2.5.1 Enviar dinero a un usuario por medio de la plataforma de CityWallet.

Se inició la actividad E3HA1 que consiste en el diagrama de flujo para el envío de dinero de un usuario a otro usuario, dando como resultado el diagrama ubicado en el anexo B figura 11.

Como se puede observar en el diagrama de flujo una vez el usuario emisor se encuentre autenticado en la aplicación se dirige a la pestaña de envío de dinero, una vez allí se solicita nombre del usuario receptor, monto y concepto de la transacción. Una vez enviada la transacción a la plataforma de CityWallet esta valida, procesa el envío y notifica el estatus de la transacción. De ser exitosa la transacción se les notifica a ambos usuarios, de no serlo se le notifica al usuario emisor el motivo por el cual no se pudo realizar.

Para el envío de dinero de un usuario a otro usuario la plataforma de CityWallet requerirá al emisor una clave de transacciones, dicha clave debe ser numérica y de cuatro dígitos. Esta clave se podrá crear en la aplicación, de igual forma se podrá restaurar en cualquier momento respondiendo preguntas de seguridad.

Para brindar seguridad al servicio de envío de dinero la clave viajará bajo el algoritmo de encriptación conocido como MD5, de igual forma la aplicación solicitará al núcleo de CityWallet una pre autorización de transacción. Esta pre autorización tendrá el nombre de ticket de transacción y contendrá un número identificador generado aleatoriamente, un string firmado por el servidor y una fecha de creación

que permitirá vencer este ticket si no se realiza la transacción en un tiempo determinado.

2.5.2 Solicitar dinero a un usuario por medio de la plataforma de CityWallet.

En este proceso el usuario de igual forma debe estar autenticado en la aplicación, una vez ubicado en la pestaña solicitud de dinero completa los datos obligatorios y envía la solicitud. La plataforma de CityWallet notifica al usuario receptor que se le está solicitando un monto determinado. Esta notificación llega al usuario al buzón de la aplicación, una vez leído el mensaje el usuario receptor autoriza o no la transacción monetaria completando así el ciclo de la transferencia. Para finalizar el hito E3H1 se realizó la actividad E3H1A2, dando como resultado el diagrama de flujo ubicado en el anexo B figura 11. En el mismo se puede ver en detalle el proceso de solicitud de dinero.

Una vez diseñado el proceso de pago, se procedió realizar el hito identificado como E3H2 que se descompone en la actividad E3H1A1 y la actividad E3H1A2.

Para culminar este sprint se identificaron los módulos necesarios a desarrollar el proceso de transferencias de fondos monetarios a través de un nombre de usuario. Para culminar este diseño se dividieron en tres módulos que serán listados a continuación e incluidos en el product backlog

- Gestión de claves de transacción.
- Gestión de transacciones.

- Gestión de transacciones por medios alternativos

2.6 Sprint 5

En la apertura de este sprint se planificó la épica identificada como E4, específicamente los hitos E4H1, E4H2, E4H3, E4H4 sus respectivas actividades como se observa en el anexo A tabla 7.

Se inició con la actividad E4H1A1, que consiste en la creación de la tabla que aloja la clave de transacción de cada usuario. Esta tabla tiene por nombre transactionPins y está compuesta por el identificador del usuario, una pregunta, una respuesta secreta y la clave de 4 dígitos. Cabe destacar que la clave y la respuesta secreta están encriptadas con MD5, esta tabla se puede observar en el anexo C tabla 16.

Luego se procedió a realizar los métodos necesarios en el núcleo del sistema para poder soportar la funcionalidad de creación de clave, necesaria para el envío de fondos monetarios por parte de un usuario cumpliendo así con la actividad E4H2A3. Finalmente se dio inicio a la actividad E4H2A3 que por medio del método post el recurso denominado transactionPins que recibirá la información necesaria para la creación de la clave de transacción. También se creó el método get para obtener la pregunta secreta del usuario, en el anexo G figura 40 y figura 41 respectivamente.

Para la integración con la aplicación móvil se realizaron las actividades E4H2A1, E4H2A y E4H2A que consistieron en la creación de la ventana con sus respectivos componentes y estilos. De forma inmediata se creó la funcionalidad de la ventana y se estableció la comunicación con el recurso transactionPins a través del método post.

Para el hito E4H3 se comenzó con la actividad E4H2A1 que consistió en el desarrollo de los métodos pertinentes en el núcleo del sistema que permiten al usuario restablecer la clave de transacción. Para que sea exitoso el restablecimiento de clave, el usuario debe enviar la respuesta secreta que facilitó al crear su clave. La siguiente actividad fue identificada como E4H2A2, la cual agregó al web services el método put de transactionPins. Este método se puede observar en el anexo G figura 40.

Posteriormente se desarrolló en la aplicación móvil la ventana de restablecimiento de clave con sus respectivos atributos y diseños, esta ventana puede observarse en el anexo E figura 26 y 27. También se procedió a darle funcionalidad a esta ventana y a realizar la comunicación a través del método put con el recurso transactionPins. De esta forma se cumplió con las tareas E4H2A1, E4H2A2 yE4H2A3.

2.7 Sprint 6

El sexto sprint se inició con el desarrollo de la épica identificada como E5 que tiene como objetivo la implementación del diseño de pago realizado en el sprint 4. Los hitos seleccionados fueron E5H1, E5H2, E5H3 y E5H4 y sus respectivas actividades. El sprint backlog puede visualizarse en el anexo A tabla 8.

La actividad E5H1A1 partió con la creación de la tabla transacciones la cual aloja las operaciones realizadas por los usuarios con los atributos que se pueden observar en el anexo C tabla 17. Una vez creada esta tabla se procedió con la actividad E5H1A2, en esta actividad se crearon los métodos necesarios

en el núcleo de la plataforma de CityWallet. Estos métodos permiten generar un número de ticket aleatorio que recibirá el usuario en su aplicación móvil y que permitirá la transferencia de fondos monetarios. Si el usuario no posee este ticket, se encuentra alterado o vencido la transferencia no se realizará. Una vez culminada esta actividad se realizó el diseño del API llamado transactionTickets que por medio del método get envía el ticket solicitado a la aplicación móvil, cumpliendo así con la actividad E5H1A3. El diagrama de proceso se puede observar en el anexo B figura 12.

Con el fin de hacer la aplicación móvil más amigable al momento de realizar transferencias monetarias, se desarrolló la actividad E5H2 compuesta por las actividades E5H2A1 y E5H2A2. La primera actividad consiste en realizar los métodos necesarios para obtener el nombre de los usuarios, descripción e imagen de perfil de los primeros 5 usuarios que coincidan con los datos enviados por medio de la aplicación móvil. La segunda actividad consiste en realizar el API llamado contacts que por medio del método get enviará la información requerida, el diagrama de proceso se puede visualizar en el anexo B figura 13.

El hito E5H3 compuesto por las actividades E5H31 y E5H3A2 consiste en implementar el diseño de transacciones realizado en el sprint número cuatro. Para ello se realizan los métodos indispensables en el núcleo del sistema y el recurso llamado transactions que por el método post recibe la solicitud de envío de fondos monetarios. El API puede observar en el anexo G figura 43 y su diagrama de proceso en el anexo B figura 14.

Para finalizar el sprint se ejecutó el hito E5H4 que consistió en la creación de la ventana para enviar los fondos monetarios, su funcionalidad y la comunicación entre la aplicación móvil y el recurso transactions. Esta ventana se

puede observar en el anexo E figura 28.

2.8 Sprint 7

Para el cumplimiento de este sprint se tomaron los hitos faltantes de la épica 5 identificados como E5H5, E5H6, E5H7, E5H8, E5H9 y sus respectivas actividades tal y como se aprecia en el sprint backlog en el anexo A tabla 9.

Comenzando con el identificador E5H5 cuyas actividades eran E5H5A1 y E5H5A2. Se procedió al desarrollo en el núcleo de la plataforma los métodos necesarios para obtener las transacciones realizadas por un usuario ordenadas cronológicamente y paginadas, tal y como se definió en el sprint cuatro. El siguiente paso fue realizar el método get para el recurso transactions. Este API puede observarse en el anexo G figura 44 y su diagrama de proceso en el anexo B figura 15.

El siguiente paso fue cumplir con el hito E5H6 compuesto por las actividades E5H6A1, E5H6A2, E5H6A3, E5H6A4 y E5H6A5. Se agregó a la ventana dashboard creada previamente en el sprint cuatro una lista de las últimas 3 transacciones realizadas por el usuario y la funcionalidad necesaria para que se pudiera visualizar. Del mismo modo se realizó la pantalla llamada balance donde se puede consultar todas las transacciones ejecutadas por el usuario. Finalmente se le dio funcionalidad y se realizó la comunicación entre la pantalla dashboard y balance el API respectivo. La ventana balance puede observarse en el anexo E figura 29.

A continuación se trabajó el hito E5H7 desarrollando los métodos necesarios para calcular el saldo disponible del usuario y realizando el recurso que consulta a través del método get por la aplicación móvil, el diagrama de

proceso se puede observar en el anexo B figura 16. Una vez finalizado se realizó la comunicación con el API y la integración en las distintas ventanas para que el usuario siempre visualice el saldo disponible.

Para concluir la épica E5 se creó la ventana correspondiente a recarga de saldo, se desarrolló su funcionalidad y se integró con el API de recarga de saldo que ya posee CityWallet en su plataforma actual. De esta forma se cumple con el hito E5H9 y sus actividades, esta ventana puede apreciarse en el anexo E figura 30.

2.9 Sprint 8

Se realizó la organización del sprint backlog tomando los hitos E6H1, E6H2, E6H3, E7H1 y E7H2 con sus actividades tal y como se observa en el anexo A tabla 10.

Para iniciar el aporte tecnológico planteado en este trabajo instrumental de grado se diseñó una capa que permitiera realizar la traducción de mensajes SMS a datos móviles para garantizar la disponibilidad de las transacciones monetarias. Esta capa realiza la conversión de datos recibe el nombre de Gateway SMS.

Si la aplicación móvil de CityWallet realiza el envío de dinero a otro usuario pero la señal de datos móviles es 2G, enviará los datos vía SMS siendo recibida por el Gateway SMS. Este a su vez realiza la conversión por medio de un script. La conversión es recibida por el núcleo de CityWallet que a su vez gestiona la respectiva transacción y da respuesta al Gateway SMS y luego a la aplicación.

Para exponer los servicios de la capa de traducción contemplado en el

hito E6H2 se configuró un dispositivo de SMS que a través de un número telefónico asignado por una empresa de telefonía venezolana recibirá los SMS. Una vez configurado este dispositivo se realiza el script con el cual se procesa el mensaje recibido y se traduce a datos entendibles por el núcleo de CityWallet para que este pueda procesar la transferencia. También se agregaron métodos en el núcleo de envío de transferencias de Citywallet para que pudiera distinguir y procesar si una transacción se está realizado vía SMS. Una vez completada la transacción se le devuelve un mensaje de éxito o error al Gateway SMS que a su vez forma y envía por medio del dispositivo de SMS el mensaje al usuario. Este mensaje es interpretado por la aplicación y mostrado al usuario, vale la pena destacar que este proceso tiende a tardar aproximadamente unos 20 segundos más que vía datos móviles. El diagrama de proceso puede consultarse en el anexo B figura 17.

Para finalizar este aporte tecnológico, se modificó la funcionalidad de la ventana envío de dinero para que sea capaz de detectar si dicha transacción será enviada al Gateway SMS o al API correspondiente.

Tal y como se diseñó en el cuarto sprint, la petición de dinero de un usuario a otro usuario se realiza a través de mensajes. En el hito E7H1 se procede a crear la tabla messages con la estructura que se observa en el anexo C tabla 18 y se crean los métodos necesarios en el núcleo de CityWallet para gestionar los mensajes de solicitudes de dinero. También se crea el API que por medio del método post recibirá la información enviada por la aplicación móvil. Este API puede observarse en el anexo G figura 45 y el diagrama de proceso en el anexo B figura 16.

Acto seguido se creó la ventana que lleva por nombre solicitud de dinero,

esta ventana posee los mismos inputs que se requieren para el envío de dinero tal y como se observa en el anexo E figura 31.

2.10 Sprint 9

Este sprint backlog estuvo compuesto de los últimos hitos referentes a la aplicación móvil los cuales son, E7H3, E7H4, E8H1, E8H1 y E8H2. Estos hitos y sus actividades se pueden observar en el anexo A tabla 11.

Se inicia con el hito E7H3 que consiste en agregar al núcleo de CityWallet los métodos requeridos para listar todos los mensajes dirigidos al usuario que los consulta. También se diseña el API que sirve la lista generada a la aplicación móvil, el API generado puede observarse en la en el anexo G figura 44 y su diagrama de proceso en el anexo B figura 19.

El hito siguiente a realizar fue el E7H4 que consistió en el desarrollo de la pantalla que lleva por nombre Inbox. La funcionalidad de esta pantalla es listar a los usuarios todos los mensajes enviados por el sistema y las solicitudes de dinero realizadas por otros usuarios. Una vez culminada la pantalla y su funcionalidad se procedió a realizar la comunicación con el recurso messages a través del método get. Esta pantalla se puede observar en el anexo E figura 32.

Luego se desarrollaron los hitos E8H1 y E8H2 que consiste en agregar al núcleo del sistema los métodos requeridos para el retiro de dinero de los usuarios a sus cuentas bancarias. Debido a que las políticas de CityWallet no permiten almacenar datos bancarios de los usuarios en la base de datos, una vez recibido la solicitud de retiro de dinero se procede a descontar el dinero de la cuenta de usuario y a enviar un correo al departamento administrativo para la realización de la transferencia bancaria de forma clásica. Se publicó el API necesario para

recibir la petición por medio del método post, se puede visualizaren en el anexo G figura 47

Para concluir este sprint se creó la ventana donde se solicitan los datos necesarios para iniciar el retiro de dinero, se desarrolló su funcionalidad y se procedió a integrar con el API correspondiente. Esta ventana se puede visualizar en el anexo E figura 33.

2.11 Sprint 10

La planificación de este sprint se basó en épica E9, específicamente los hitos E9H1, E9H2, E9H3 y E9H4, se visualizan en el anexo A tabla 12. Estos consisten en el desarrollo de una aplicación de web donde el departamento administrativo pudiese obtener algunos reportes de interés. Se agendo y se sostuvo una reunión con la persona encargada de dicho departamento solicitando tres reportes necesarios:

- Transacciones por usuario.
- Transacciones probadas o rechazadas de un usuario.
- Recargas realizadas por un usuario en la plataforma.

Se solicitó que el acceso a esta aplicación seria exclusivamente por los usuarios del departamento administrativo. Debido a que existe otra plataforma de CityWallet referente a estacionamientos y esta nueva plataforma referente a transferencias monetarias, los reportes se actualizan cada media noche para sincronizar ambas bases de datos. De esta forma se garantiza que aparezcan todas las operaciones realizadas por un mismo usuario en ambas plataformas.

La aplicación se inició con el desarrollo del hito E9H1 que consiste en la

creación de la ventana de inicio de sesión con su correspondiente estilo y funcionalidad. Luego se procedió a realizar la integración con AWS cognito para autenticar al usuario administrativo en el user pool creado en este trabajo instrumental de grado.

El proceso de descarga de información se realiza por medio de un servicio realizado por CityWallet, este proceso no se encuentra en el alcance de este trabajo instrumental de grado. Para que este proceso pueda descargar la información de la plataforma de estacionamiento y de la plataforma de transferencias monetarias se requiere la creación de una tabla donde se alojará dicha información que luego será consultada por la aplicación web.

Para el cumplimiento del hito E9H2 se procede a realizar la tabla contemplada en la actividad E9H2A1. Luego se procede a la creación de un método de consulta que seleccione el listado de las transacciones realizadas por un usuario, otro método que retorna una lista con las transacciones fallidas o exitosas de un usuario y por último se crea una tabla de recargas y el método que consulta la información de esta tabla. Acto seguido se desarrolló el webservice y cada método necesario cumpliendo así con los hitos E9H3 y E9H4.

2.12 Sprint 11

Para finalizar este trabajo instrumental de grado se organizaron en el sprint backlog los hitos E9H5, E9H6, E9H7, E10H1 y E10H2 con sus actividades tal y como se observa en el anexo A tabla 13.

Para visualizar el contenido de las listas en la aplicación web se procedió a diseñar una ventana que contuviera 3 pestañas, en cada pestaña se despliega un reporte. En la realización de los hitos E9H3, E9H3 y E9H4 se desarrollaron

los HTML cada uno con su estilo. Luego se implementó una librería que lleva por nombre Angular Smart Table Component la cual facilita el ordenamiento y la búsqueda de la información. Estas ventanas se pueden observar en el anexo F figura 35, figura 36 y figura 37 respectivamente.

En cumplimiento de la épica E10, que consiste en la automatización de las recargas por transferencias o depósitos bancarios, se creó un repositorio en AWS S3 que será el encargado de almacenar los archivos bancarios para su procesamiento tal y como lo especifica la actividad E10H1A1. Una vez creado se realiza la permisología de y la configuración del tipo de archivo a recibir cumpliendo así con la actividad E10H1A2 Para el procesamiento del archivo se creó un recurso en AWS que toma el archivo alojado en el repositorio y consulta la base de datos para verificar si existen transferencias o depósitos bancarios notificados por el usuario. Si el número de referencia reportado por el usuario coincide con el número de referencia contenido en el archivo facilitado por el banco automáticamente se acredita el monto la cuenta CityWallet. Si no existen coincidencias se registra en un log los números de referencia que presentan irregularidades, terminando así el hito E10H2. En la aplicación de CityWallet cuando un usuario va a reportar una transferencia se le notifica que será acreditada en un máximo de 24h, esto es debido a que el proceso correrá y hará la asignación de saldo todos los días a media noche.

CAPÍTULO V – RESULTADOS

1. RESULTADOS

Una vez concluida la etapa de desarrollo la cual tuvo una duración de 12 sprints se presentan los resultados obtenidos basados en cada objetivo específico planteado.

1.1 DISEÑAR E IMPLEMENTAR UNA BASE DE DATOS QUE SE ADAPTE A LOS REQUERIMIENTOS DEL SISTEMA.

El diseño final obtenido para la base de datos no relacional implementada se obtuvo durante la iteración del tercer, quinto, sexto y octavo sprint. Obteniendo las tablas users, notificationChannels, transactionPins, transiciones y messages necesarias para el funcionamiento del sistema.

1.2 Desarrollar un webservice basado en REST el manejo de peticiones

Se cumplió con la publicación del webservice necesario para la comunicación entre la aplicación móvil y el núcleo del sistema de transferencias monetarias de CityWallet. Se diseñaron siete APIS y se trabajaron en el tercer, quinto sexto y séptimo sprint.

1.3 Desarrollar el núcleo del sistema.

El desarrollo del núcleo del sistema permite la gestión y manipulación de toda la información del usuario, así como el manejo de todas las excepciones para que el sistema se mantenga funcional constantemente. La ejecución de este núcleo dio como resultado siete clases cada una con los métodos requeridos para su funcionamiento, estos realizándose progresivamente durante el tercer, quinto, sexto y séptimo sprint.

1.4 Desarrollar aplicación web para gestión administrativa.

El desarrollo de este producto se obtuvo en el sprint 9 y 10 obteniendo la aplicación donde el departamento administrativo consultara los usuarios, el

estado de sus transacciones y las recargas realizadas. Para este producto se realizaron métodos de consulta, el webservice y el front-end donde se visualizan los reportes.

1.5 Desarrollar proceso batch para recargas de depósitos en cuentas bancarias.

El proceso de recarga de CityWallet se realizaba de forma manual, al desarrollar este proceso se obtuvo como resultado la automatización de las recargas en el sistema.

1.6 Desarrollar aplicación móvil para usuarios de Citywallet.

La unión de los objetivos anteriores dio como resultado la nueva aplicación móvil de CityWallet, con la cual usuarios podrán tener la facilidad de transferir dinero entre sí, así como recargar y retirar dinero a sus cuentas bancarias de una manera rápida y sin complicaciones.

1.7 Diseñar e implementar un proceso de transferencias de fondos monetarios a través de un nombre único.

Al diseñar el proceso de pago se obtuvo la documentación necesaria para el entendimiento del proceso de transferencias. Entre esta documentación se tiene el diagrama de flujo y, el diagrama de proceso. Al implementar este diseño desde el back-end hasta el front-end los usuarios de CityWallet podrán realizar y solicitar las transferencias monetarias según sus necesidades.

2. CONCLUSIONES

El cumplimiento de los objetivos planteados en este trabajo instrumental de grado trajo como consecuencia un nuevo sistema de micro pagos que no requieren de la banca privada o pública para operar.

Este nuevo sistema permite a través de una aplicación móvil registrarse como usuario de la plataforma, una vez completado el registro y verificada la cuenta el usuario, tendrá acceso a realizar pagos. El sistema permite al usuario realizar recargas de dinero a su cuenta CityWallet con el objetivo de pagar o cobrar productos y/o servicios con tan solo el nombre de usuario. Para confirmar el envío de dinero el usuario solo debe colocar su clave de transferencia y esperar que le llegue un mensaje de confirmación.

Con este sistema desarrollado para CityWallet se provee mitigar el problema de efectivo que afecta a Venezuela, así como la dificultad al momento de realizar pagos con instrumentos bancarios clásicos. El sistema estará disponible así el usuario no posea internet, esto permite la posibilidad de que CityWallet expanda operaciones a sitios remotos con señal telefónica e incluiría a usuarios no frecuentes de la banca venezolana.

También cuenta con un sistema donde se podrá visualizar las transacciones de cada usuario para brindarle apoyo en caso de algún inconveniente con las transacciones o recargas realizadas.

Para concluir, este sistema agilizará las operaciones de pagos y cobros del territorio nacional, así como la disminución del costo de operación de los comercios al recibir el pago de sus clientes. De igual forma disminuirá el manejo de efectivo lo que brinda seguridad y comodidad a los venezolanos.

3. RECOMENDACIONES

En el desarrollo de esta plataforma de pagos, se observaron una serie de mejoras que podrían realizarse en un mediano plazo.

En primer lugar se recomienda diseñar y planificar un sistema de monitoreo de transacciones que sea capaz de detectar cuando un usuario realice un pago de un monto no habitual en su cuenta de CityWallet para la seguridad del cliente.

En cuanto a la usabilidad de la aplicación se recomienda realizar un algoritmo de búsqueda que permita al usuario obtener a los usuarios frecuentes o cercanos para realizar o solicitar un pago. También se recomienda permitir al cliente cambiar su nombre de usuario a través de su perfil por lo menos una vez.

Finalmente se recomienda solicitar una serie de documentos si el usuario es jurídico. Una vez cumpla con estos documentos solicitados se acredita como usuario verificado. De esta forma crearía confianza dentro de la red, también se aconseja crear un sistema de puntaje donde el cliente jurídico sea calificado en caso de prestar servicios.

4. REFERENCIAS BILIOGRÁFICAS

[1] A. Banal-Estañol, «¿Qué son las Fintech?,» La Vanguardia. [En línea]. Disponible en: http://www.lavanguardia.com/economia/management/20160811/403831350794/que-son-fintech.html. [Último acceso: 3 de abril de 2017].

[2] Igual D., (2016), Fintech: lo que la tecnología hace por las finanzas. Barcelona, España: Producción editorial.

[3] Spanish Fintech, «Definición de Fintech,» Spanish Fintech. [En línea]. Disponible en: http://spanishfintech.net/definicion-de-fintech/. [Último acceso: 3 de abril de 2017].

[4] R. Botsman, «Economías colaborativas: el P2P,» Rachel Botsman, 12 de diciembre de 2012. [En línea]. Disponible en: http://www.rachelbotsman.com/. [Último acceso: 10 de abril de 2016].

[5] A. Murakami-Fester, «Venmo, PayPal, Square Cash ando more: What are Peer-to-Peer Payments?,» NerdWallet, 21 de octubre de 2016. [En línea]. Disponible en: https://www.nerdwallet.com/blog/banking/p2p-payment-systems/. [Último acceso: 10 de noviembre de 2016].

[6] A. Ruben, «Qué es el NFC Móvil, cómo funciona y que puedes hacer con él,» Computer Hoy, 8 de febrero de 2017. [En línea]. Disponible en: http://computerhoy.com/noticias/life/que-es-nfc-movil-para-que-sirve-como-funciona-24207 [Último acceso: 27 de enero de 2017].

[7] J. Penalva, «NFC: qué es y para qué sirve,» 25 de enero de 2011. [En línea]. Disponible en: https://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve. [Último acceso: 27 de enero de 2017].

[8] Amazon Web Services «¿Qué es la informática en la nube?,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/what-is-cloud-computing/. [Último acceso: 28 de enero de 2017]

[9] Amazon Web Services, «Tipos de cloud computing,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/types-of-cloud-computing/. [Último acceso: 27 de febrero de 2017]

[10] Amazon Web Services, «Productos de la nube,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/products/. [Último acceso: 27 de febrero de 2017]

[11] Amazon Web Services, «Amazon DynamoDB,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/dynamodb/ [Último acceso: 27 de febrero de 2017]

[12] Amazon Web Services, «¿Qué Amazon Cognito?,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/cognito/faqs/. [Último acceso: 27 de febrero de 2017]

[13] Amazon Web Services, «Amazon Cognito,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/cognito/. [Último acceso: 27 de febrero de 2017]

[14] Amazon Web Services, «Amazons3,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/api-gateway/. [Último acceso: 27 de febrero de 2017]

[15] Amazon Web Services, «AWS Device Farm,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/device-farm/. [Último acceso: 12 de marzo de 2017].

[16] Amazon Web Services, «Preguntas frecuentes sobre AWS Device Farm,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/device-farm/faq/. [Último acceso: 12 de marzo de 2017].

[17] Amazon Web Services, «Amazon Mobile Analytics,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/mobileanalytics/. [Último acceso: 12 de

marzo de 2017].

[18] Amazon Web Services, «Preguntas frecuentes sobre Amazon Mobile Analytics,» Amazon, 2017. [En línea]. Disponible en: https://aws.amazon.com/es/mobileanalytics/faqs/. [Último acceso: 12 de marzo de 2017].

[19] Software Engineering Institute, «Community Software Architecture Definitions,» Software Engineering Institute, 2017. [En línea]. Disponible en: http://www.sei.cmu.edu/architecture/start/glossary/community.cfm. [Último acceso: 27 de febrero de 2017].

[20] Java, «¿Qué es la tecnología Java y para qué sirve?,» Oracle. [En línea]. Disponible en: https://www.java.com/es/download/faq/whatis_java.xml. [Último acceso: 3 de mayo de 2017].

[21] M. Álvarez, «Qué es Java,» Desarrollo Web, 18 de julio de 2001. [En línea]. Disponible en: https://desarrolloweb.com/articulos/497.php. [Último acceso: 3 de mayo de 2017].

[22] J. Gutierrez, «¿Qué es un framework?,» Escuela técnica de Ingeniería Informática, Universidad de Sevilla. [En línea]. Disponible en: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf. [Último acceso: 16 de abril de 2017].

[23] Global Mentoring, «¿Qué es un framework?,» Global Developers Mentoring, 16 de enero de 2012. [En línea]. Disponible en: http://globalmentoring.com.mx/cursos-java/java-frameworks/que-es-un-framework/. [Último acceso: 16 de abril de 2017].

[24] Oja.la, «¿Ya sabes cómo usar Angular 2 para tus proyectos web?,» Oja.la. [En línea]. Disponible en: https://oja.la/blog/ya-sabes-como-usar-angular-2-para-tus-proyectos-web/. [Último acceso: 15 de abril de 2017].

[25] M. Álvarez «Qué es Ionic 2,» Desarrollo web, 2 de marzo de 2017. [En línea].

Disponible en: https://www.desarrolloweb.com/articulos/que-es-ionic2.html. [Último acceso: 15 de abril de 2017].

[26] Slash Mobility «Ionic 2: el framework del futuro,» Slash Mobility, 17 de agosto de 2016. [En línea]. Disponible en: http://slashmobility.com/blog/2016/08/ionic-2-el-framework-del-futuro/. [Último acceso: 15 de abril de 2017].

[27] C. Mellon «*Client/Server software architectures-an overview,*» *Software Engineering* Institute. [En línea]. Disponible en: http://www.sei.cmu.edu/str/descriptions/clientserver_body.html. [Último acceso: 10 de noviembre de 2016].

[28] B. Márquez y J. Zulaica «Implementación de un reconocedor de voz gratuito a el sistema de ayuda a invidentes Dos-Vox en español,» Universidad de las Américas, 12 de enero de 2004. [En línea]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/marquez_a_bm/. [Último acceso: 10 de noviembre de 2016].

[29] IBM Developer Works, «Introducción al SOA y Servicios Web,» IBM, 2017. [En línea]. Disponible en: https://www.ibm.com/developerworks/ssa/webservices/newto/service.html. [Último acceso: 13 de marzo de 2017].

[30] Culturación «¿Qué es y para qué sirve un Web Service?,» Culturación. [En línea]. Disponible en: http://culturacion.com/que-es-y-para-que-sirve-un-web-service/. [Último acceso: 13 de marzo de 2017].

[31] TechTerms «Definicion de API,» TechTerms. [En línea]. Disponible en: https://techterms.com/definition/api [Último acceso: 13 de marzo de 2017].

[32] ECMA International, The Json Data Interchange Format, Ginebra: ECMA International, 2013.

[33] National Institute of Standard and Technology. General Principles for Information

Systems Security Policies. New York: Mc Graw Hill, 1996.

[34] J. Angel, «Criptografía para principiantes,» HTML Web. [En línea]. Disponible en: http://www.htmlweb.net/seguridad/cripto_p/cripto_princ_1.html. [Último acceso: 19 de abril del 2017].

[35] Y. Marrero, «La criptografía como elemento de la seguridad informática,» Scientific Electronic Library Online. [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352003000600012. [Último acceso: 19 de abril de 2017].

[36] American National Standards Institute. ANSI X3.106 American National Standard - Data Encryption Algorithm Modes of Operation. Washington DC:American National Standards Institute,1983.

[37] I. Martinez «Qué es MD5, cómo funciona y para que se usa,» Rootear. [En línea]. Dispomible en: https://rootear.com/seguridad/md5-como-funciona-usos. [Último acceso: 20 de marzo de 2017].

[38] Herramientas Web para la enseñanza de protocolos de comunicación, «MD5,» [En línea]. Disponible en: http://neo.lcc.uma.es/evirtual/cdd/tutorial/presentacion/md5.html. [Último acceso: 20 de marzo de 2017].

[39] Informática Hoy, «Android, el sistema operativo para móviles de Google,» Informática Hoy, 2010. [En línea]. Disponible en: http://www.informatica-hoy.com.ar/soluciones-moviles/Android-el-sistema-operativo-para-moviles-de-Google.php. [Último acceso: 3 de mayo de 2017].

[40] SoftDoit «¿Qué es el sistema operativo iOS?,» SoftDoit. [En línea]. Disponible en: https://www.softwaredoit.es/definicion/definicion-sistema-operativo-ios.html. [Último acceso: 3 de mayo de 2017].

[41] Fast Company, «¿How Zappos Uses One-Week Work Sprints to Laungh Big

Projects Fast?», Fast Company & Inc, 2.017 [En línea]. Disponible en: https://www.fastcompany.com/3032947/how-zappos-uses-one-week-work-sprints-to-launch-big-projects-fast . [Último acceso: 10 de Abril de 2017].

[42] Tech Crunch, «Here´s How Spotify Scales Up And Stays Agile: It Runs "Squads" Like Lean Startups», AOL Inc. 2.013-2.017, [En línea]. Disponible en: https://techcrunch.com/2012/11/17/heres-how-spotify-scales-up-and-stays-agile-it-runs-squads-like-lean-startups/[Último acceso: 15 de Abril de 2017].

[43] Scrum Guides, «Scrum Guide», Scrum.Org & ScrumInc 2.016 [En línea]. Disponible en: http://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100. [Último acceso: 15 de Abril de 2.017]

[44] Dr. Alistair Cockburn, «Using Both Incremental and Iterative Development», Software Ingeniering Technolgy [En línea]. Disponible en http://static1.1.sqspcdn.com/static/f/702523/9242211/1288741989673/200805-Cockburn.pdf?token=b4O8EtA2%2F7Dyr8oBCXwjr54d%2BTE%3D. [Último acceso: 15 de Abril de 2.017]

[45] Scrum Guides, «Scrum Guide», Scrum.Org & ScrumInc 2.016 [En línea]. Disponible en: http://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100. [Último acceso: 15 de Abril de 2.017]

[46] Scrum Guides, «Scrum Guide», Scrum.Org & ScrumInc 2.016 [En línea]. Disponible en: http://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100. [Último acceso: 15 de Abril de 2.017]

[47] Scrum Guides, «Scrum Guide», Scrum.Org & ScrumInc 2.016 [En línea]. Disponible en: http://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100. [Último acceso: 16 de Abril de 2.017]

[48] Scrum Guides, «Scrum Guide», Scrum.Org & ScrumInc 2.016 [En línea]. Disponible en: http://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf#zoom=100. [Último acceso: 16 de Abril de 2.017]

[49] Scrum Manager, «Epics», Scrum Manager 2.014 [En línea]. Disponible en: https://www.scrummanager.net/bok/index.php?title=Epic. [Último acceso: 16 de Abril de 2.017]

[50] Scaled Digital Framework, «Milestones», 17 de junio de 2017 [En línea]. Disponible en: http://www.scaledagileframework.com/milestones. [Último acceso: 16 de Abril de 2.017]

ANEXOS

A. Planificación

ID	Épica
E1	Registro y autenticación de usuario
E2	Creación y acceso de cuentas en plataforma
E3	Proceso de transferencia de fondos
E4	Gestión de claves de transacción
E5	Gestión de transacciones
E6	Gestión de transacciones alternativas
E7	Gestión de mensajes
E8	Gestión de retiro de fondos
E9	Aplicación del Administrador
E10	Automatización de integración Bancaria

Tabla 1. Épicas del proyecto.

Fuente: [Elaboración propia]

ID	Épica	Hito
E1H1	Registro y autenticación de usuario	Exponer servicio de registro de usuario en la plataforma CityWallet
E1H2		Exponer servicio de envío de código confirmación del usuario en la plataforma CityWallet
E1H3		Exponer servicio de reenvío de código confirmación del usuario en la plataforma CityWallet
E1H4		Registro de usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet
		Envío de código validación del usuario a través de

E1H5	Registro y autenticación de usuario	aplicación móvil consumiendo servicios de plataforma CityWallet.
E1H6		Reenvío de código validación del usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E1H7		Autenticar usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E1H8		Exponer Servicio para reiniciar contraseña de acceso para usuario no autenticado en la plataforma CityWallet
E1H9		Reinicio de claves de acceso para usuario no autenticado a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E2H1	Creación y acceso de cuentas en plataforma.	Creación automática de cuentas en plataforma a través de detección de evento de nuevo usuario
E2H2		Creación automática de canales de notificación de usuario en plataforma a través de detección de evento de nueva cuenta creada.
E2H3		Exponer servicio para obtener datos de usuario en la plataforma CityWallet
E2H4		Desplegar información de usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet

E3H1	Proceso de transferencia de fondos	Diseñar proceso de transferencia de fondos monetarios
E3H2		Diagramar procesos de transferencia de fondos monetarios
E3H3		Identificar módulos a desarrollar para los procesos de transferencias de fondos monetarios
E4H1	Gestión de claves de transacción	Exponer servicio crear clave de transacciones en plataforma CityWallet
E4H2		Crear clave de transacciones a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E4H3		Exponer servicio para reiniciar clave de transacciones en plataforma CityWallet
E4H4		Restablecer clave de transacciones a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E5H1	Gestión de transacciones	Exponer servicio obtener ticket de transferencias en plataforma CityWallet
E5H2		Exponer servicio obtener los usuarios de la plataforma CityWallet
E5H3		Exponer servicio el transferencias monetarias en plataforma CityWallet
E5H4		Crear envío de transferencias a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E5H5		Exponer servicio listar transferencias en plataforma CityWallet

E5H6	Gestión de transacciones	Listar transferencias a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E5H7		Exponer Servicio para obtener el saldo del usuario en la plataforma CityWallet
E5H8		Desplegar información del saldo a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E5H9		Integrar recargas monetarias a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E6H1	Gestión de transacciones alternas	Diseñar capa de traducción SMS a datos móviles
E6H2		Exponer servicio para crear transferencias vía recepción de SMS en la plataforma CityWallet
E6H3		Realizar envío de transferencias consumiendo un servicio SMS a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E7H1	Gestión de mensajes	Exponer servicio el solicitud de transferencia monetarias en plataforma CityWallet
E7H2		Crear mensaje de solicitud de transferencia monetaria a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E7H3		Exponer servicio de listar mensajes en plataforma CityWallet
E7H4		Listar mensajes a través de aplicación móvil consumiendo servicios de plataforma CityWallet

E8H1	Gestión de retiro de fondos	Exponer servicio listar mensajes en plataforma CityWallet
E8H2		Crear solicitud de retiro de dinero a través de aplicación móvil consumiendo servicios de plataforma CityWallet
E9H1	Aplicación del Administrador	Autenticar usuario a través de aplicación Web consumiendo servicios de plataforma CityWallet
E9H2		Exponer servicio obtener reporte de transacciones por usuario en la plataforma CityWallet
E9H3		Exponer servicio de recargas por usuario de la plataforma CityWallet
E9H4		Exponer servicio de transacciones por su estatus en la plataforma CityWallet
E9H5		Desplegar reportes transacciones por su estatus través de aplicación Web consumiendo servicios de plataforma CityWallet
E9H6		Desplegar reporte de transacciones por usuario a través de aplicación Web consumiendo servicios de plataforma CityWallet
E9H7		Desplegar reportes de recargas por usuario través de aplicación Web consumiendo servicios de plataforma CityWallet del usuario en la plataforma CityWallet
E10H1	Automatización de integración Bancaria.	Exponer repositorio en Amazon S3
E10H2		Desarrollar proceso de recargas manuales

Tabla 2 Product Backlog

Fuente: [Elaboración propia]

ID	Hito	Actividad
E1H1A1	Exponer Servicio de registro de usuario en la plataforma CityWallet	Diseño de los atributos de AWS Cognito User pool.
E1H1A2		Configuración de los servicios del User pool de AWS Cognito
E1H2A1	Exponer servicio de envío de código confirmación del usuario en la plataforma CityWallet	Integración del diseño de correo electrónico para la confirmación de cuenta
E1H2A2		Configurar envío de código validador de Cognito
E1H3A1	Exponer servicio de reenvío de código confirmación del usuario en la plataforma CityWallet	Integración del diseño de correo electrónico para el reenvío de confirmación de cuenta
E1H3A2		Configurar reenvío de código validador de Cognito
E1H4A1	Exponer Servicio de registro de usuario en la plataforma CityWallet	Diseño de los atributos de AWS Cognito User pool.
E1H4A2		Configuración de los servicios del User pool de AWS Cognito
E1H5A1	Envío de código validación del usuario a través de aplicación móvil	Desarrollo de la ventana de código de validación
		Funcionalidad de pantalla de validación

E1H5A2	consumiendo servicios de plataforma CityWallet	
E1H6A1	Reenvío de código validación del usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de la ventana de reenvío de código de validación de cuenta
E1H6A2		Funcionalidad de pantalla de reenvío de código de validación de cuenta

Tabla 3 Sprint Backlog # 1

Fuente: [Elaboración propia]

ID	Hito	Actividad
E1H7A1	Autenticar usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de pantalla login
E1H7A2		Funcionalidad de pantalla de login
E1H8A1	Exponer servicio para reiniciar contraseña de acceso para usuario no autenticado en la plataforma CityWallet	Integración del template Email con código de validación
E1H8A2		Configurar envío de validador Cognito
E1H9A1	Reinicio de claves de acceso para usuario no autenticado a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de pantalla reset
E1H9A2		Funcionalidad de pantalla de reset

Tabla 4 Sprint Backlog # 2

Fuente: [Elaboración propia]

ID	Hito	Actividad
E2H1A1	Creación automática de cuentas en plataforma a través de detección de evento de nuevo usuario	Crear la tabla users
E2H1A2		Desarrollo de los métodos de userManager
E2H1A3		diseño del API Users método post
E2H2A1	Creación automática de canales de notificación de usuario en plataforma a través de detección de evento de nueva cuenta creada.	Creación de la tabla notificationChannels
E2H2A1		Desarrollo de los métodos de notificationChannels
E2H2A1		Diseño del API notificationChannels método post
E1H3A1	Exponer servicio para obtener datos de usuario en la plataforma CityWallet	Desarrollo de los métodos de users
		Diseño del API users método get

Tabla 5 Sprint Backlog # 3

Fuente: [Elaboración propia]

ID	Hito	Actividad
E2H4A1	Desplegar información de usuario a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de pantalla Dashboard
E2H4A2		Funcionalidad de pantalla de Dashboard
E2H4A3		Desarrollo de servicios de comunicación para el obtención de datos de usuario

E3H1A1	Diseñar proceso de transferencia de fondos monetarios	Diseñar proceso de envío de dinero
E3H1A2		Diseñar proceso de solicitud de dinero
E3H2A1	Diagramar procesos de transferencia de fondos monetarios	Diagrama de flujo envío de dinero
E3H2A2		Diagrama de proceso envío de dinero
E3H2A3		Diagrama de flujo solicitud de dinero
E3H2A4		Diagrama de proceso solicitud de dinero
E3H3A1	Identificar módulos a desarrollar para los procesos de transferencias de fondos monetarios	Identificar módulos a necesarios para el proceso de transferencia de fondos monetarios
E3H3A2		Agregar los módulos en la planificación del product backlog

Tabla 6 Sprint Backlog # 4

Fuente: [Elaboración propia]

ID	Hito	Actividad
E4H1A1	Exponer servicio crear clave de transacciones en plataforma CityWallet	Creación de la tabla transactionPins
E4H1A2		Desarrollo de los métodos de transactionPins
E4H1A3		Diseño del API transactionPins método post
E4H2A3	Crear clave de transacciones a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de pantalla crear clave de transacción
E4H2A3		Funcionalidad de pantalla de crear clave de transacción
E4H2A3		Desarrollo de servicios de comunicación para el creación clave de transacción

E4H3A1	Exponer servicio para reiniciar clave de transacciones en plataforma CityWallet	Desarrollo de los métodos de reiniciar clave
E4H3A2		Diseño del API transactionPins método put
E4H4A1	Restablecer clave de transacciones a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de pantalla crear clave de transacción
E4H4A2		Funcionalidad de pantalla de crear clave de transacción
E4H4A3		Desarrollo de servicios de comunicación para el creación clave de transacción

Tabla 7 Sprint Backlog # 5

Fuente: [Elaboración propia]

ID	Hito	Actividad
E5H1A1	Exponer servicio obtener ticket de transferencias en plataforma CityWallet	Creación de la tabla transaction
E5H1A2		Desarrollo de los métodos de transactionTickets
E5H1A3		Diseño del API transactionTickets método get
E5H2A1	Exponer servicio obtener los usuarios de la plataforma CityWallet	Desarrollo de los métodos en el núcleo para obtener los usuarios existentes
E5H2A2		Diseño del API contacts para el método get
E5H3A1	Exponer servicio el transferencias monetarias	Desarrollo de los métodos en el núcleo transaction

E5H3A2	en plataforma CityWallet	Diseño del API transaction para el método post
E4H4A1	Crear envío de transferencias a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo de pantalla enviar fondos monetarios
E4H4A2		Funcionalidad de pantalla de enviar fondos monetarios
E4H4A3		Desarrollo de servicios de comunicación para transaction por el método post

Tabla 8 Sprint Backlog # 6

Fuente: [Elaboración propia]

ID	Hito	Actividad
E5H5A1	Exponer servicio listar transferencias en plataforma CityWallet	Desarrollo de los métodos de transactions get
E5H5A2		Diseño del API transaction método get
E5H6A1	Listar transferencias a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla dashboard los últimos 3 movimientos
E5H6A2		Funcionalidad de la pantalla dashboard
E5H6A3		Desarrollo en la pantalla balance
E5H6A4		Funcionalidad de la pantalla balance
E5H6A5		Desarrollo de servicios de comunicación para transaction por el método get
vE5H7A1	Exponer Servicio para	Desarrollo de los métodos en el núcleo

	obtener el saldo del usuario en la plataforma CityWallet	balance
E5H7A2		Diseño del API balance para el método get
E5H8A1	Desplegar información del saldo a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Adicionar visualización de saldo en la aplicación
E5H8A2		Desarrollo de servicios de comunicación para el obtención de datos de usuario
E5H9A1	Integrar recargas monetarias a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla recarga
E5H9A2		Funcionalidad de la pantalla recarga
E5H9A3		Desarrollo de servicios de comunicación para recarga por el método post

Tabla 9 Sprint Backlog # 7

Fuente: [Elaboración propia]

ID	Hito	Actividad
E6H1A1	Diseñar capa de traducción SMS a datos móviles	Diseñar capa de traducción
E6H1A2		Diagramar proceso de envío de dinero por medio de la capa de traducción
E6H2A1	Exponer servicio para crear transferencias vía recepción de SMS en la	Configuración del receptor SMS
E6H2A2		Desarrollo del script de comunicación entre el

	plataforma CityWallet	receptor y el núcleo de Citywallet
E6H2A3	monetarios	Desarrollo de los métodos en el núcleo para obtener los transferencias existentes
E6H3A1	Realizar envío de transferencias consumiendo un servicio SMS a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Agregar funcionalidad para envío de transacciones vía SMS en la ventana envío de dinero de la aplicación móvil
E7H1A1	Exponer servicio el solicitud de transferencia monetarias en plataforma CityWallet	Creación de la tabla messages
E7H1A1		Desarrollo de los métodos de messages
E7H1A1		Diseño del API messages método post
E7H2A1	Crear mensaje de solicitud de transferencia monetaria a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla solicitar dinero
E7H2A2		Funcionalidad de la pantalla solicitar dinero
E7H2A3		Desarrollo de servicios de comunicación para solicitar dinero por el método post

Tabla 10 Sprint Backlog # 8

Fuente: [Elaboración propia]

ID	Hito	Actividad
E7H3A1	Exponer servicio listar mensajes en plataforma CityWallet	Desarrollo de los métodos de mensajes get
E7H3A2		Diseño del API mensajes método get
E7H4A1	Listar mensajes a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla Inbox
E7H4A2		Funcionalidad de la pantalla Inbox
E7H4A3		Desarrollo de servicios de comunicación para Inbox por el método get
E8H1A1	Exponer Servicio para retirar dinero del usuario en la plataforma CityWallet	Desarrollo de los métodos en el núcleo retirar dinero
E8H1A2		Diseño del API retirar dinero para el método post
E8H2A1	Crear solicitud de retiro de dinero a través de aplicación móvil consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla retirar dinero
E8H2A2		Funcionalidad de la pantalla retirar dinero
E8H2A3		Desarrollo de servicios de comunicación para retirar dinero por el método post

Tabla 11 Sprint Backlog # 9

Fuente: [Elaboración propia]

ID	Hito	Actividad
E9H1A1	Autenticar usuario a través de aplicación Web consumiendo servicios de	Desarrollo de pantalla login
		Funcionalidad de pantalla de login

E9H1A2	plataforma CityWallet	
E9H2A1	Exponer servicio obtener reporte de transacciones por usuario en la plataforma CityWallet	Creación de la tabla transactions
E9H2A2		Desarrollo de los métodos de transaction
E9H2A3		Diseño del API transaction método post
E9H3A1	Exponer servicio de recargas por usuario de la plataforma CityWallet	Creación de la tabla cashIn
E9H3A2		Desarrollo de los métodos de transaction
E9H3A3		Diseño del API transaction método get

Tabla 12 Sprint Backlog # 10

Fuente: [Elaboración propia]

ID	Hito	Actividad
E9H5A1	Desplegar reportes transacciones por su estatus través de aplicación Web consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla retirar dinero
E9H5A2		Funcionalidad de la pantalla retirar dinero
E9H5A3		Desarrollo de servicios de comunicación para retirar dinero por el método post
E9H6A1	Desplegar reporte de transacciones por usuario a través de aplicación Web consumiendo servicios de plataforma CityWallet	Desarrollo en la pantalla reporte de transacciones por usuario
E9H6A2		Funcionalidad de la pantalla reporte de transacciones por usuario
E9H6A3		Integración de servicios de comunicación de transacciones para método get
E9H7A3	Desplegar reportes de	Desarrollo en la pantalla reporte de recargas

	recargas por usuario	por usuario
E9H7A3	través de aplicación Web consumiendo servicios de plataforma CityWallet	Funcionalidad de la pantalla reporte de recargas por usuario
E9H7A3		Desarrollo de servicios de comunicación para recargas por método get
E10H1A1	Exponer repositorio en Amazon S3	Crear repositorio en AWS S3
E10H1A2		Configurar permisologia en repositorio AWS S3
E10H1A1	Desarrollar proceso de recargas manuales	Analizar archivos bancarios
E10H1A2		Desarrollar métodos para el procesamiento de archivos en el repositorio de AWS s3

Tabla 13 Sprint Backlog # 11

Fuente: [Elaboración propia]

B. Diagramas

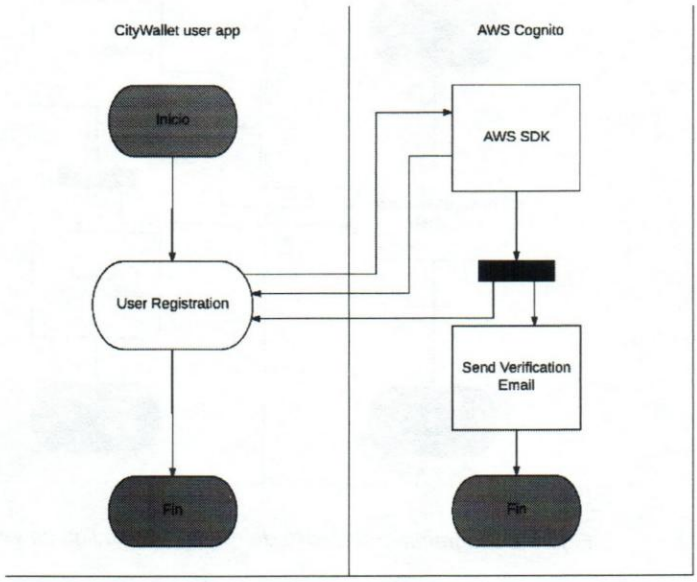

Figura 4 Diagrama de proceso de registro de usuario

Fuente: [Elaboración propia]

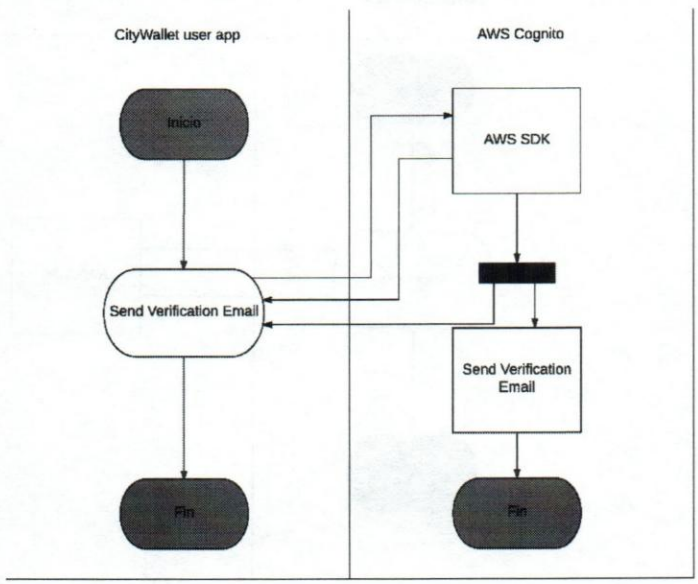

Figura 5 Diagrama de proceso de envío de código de validación

Fuente: [Elaboración propia]

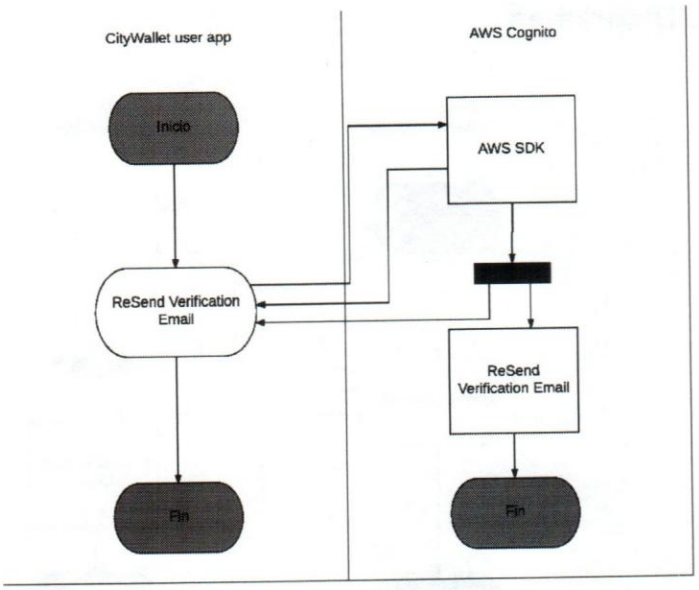

Figura 6 Diagrama de proceso de reenvío de código de validación

Fuente: [Elaboración propia]

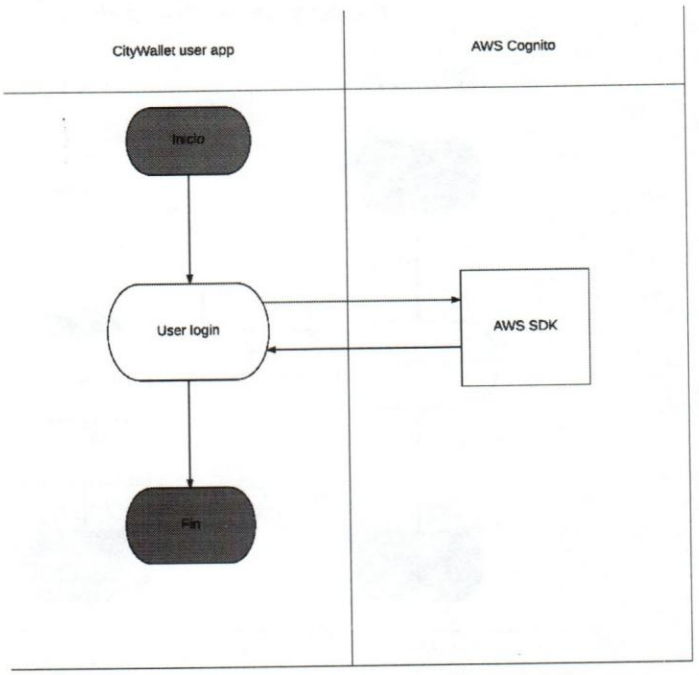

Figura 7 Diagrama de proceso de inicio de sesión

Fuente: [Elaboración propia]

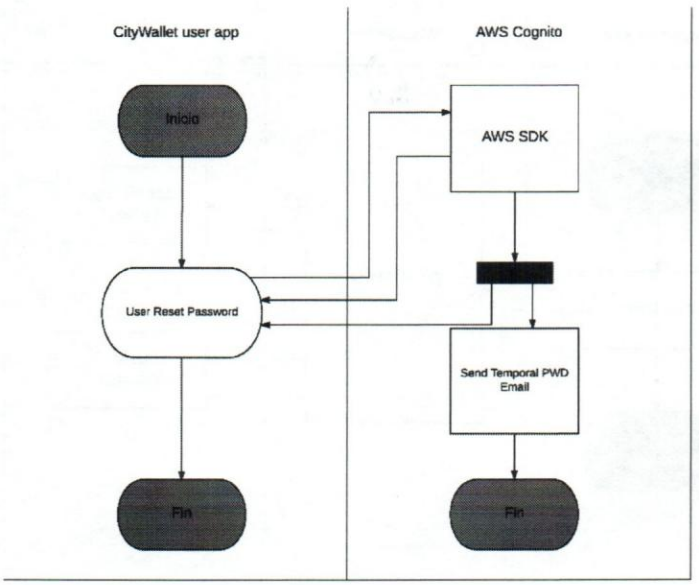

Figura 8 Diagrama de proceso de restablecimiento de contraseña.

Fuente: [Elaboración propia]

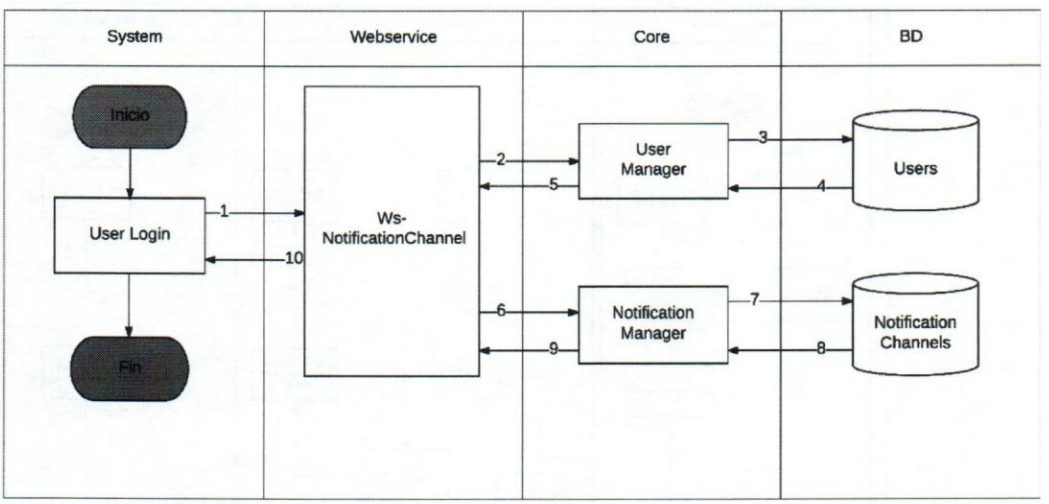

Figura 9 Diagrama de proceso de notificationChannels

Fuente: [Elaboración propia]

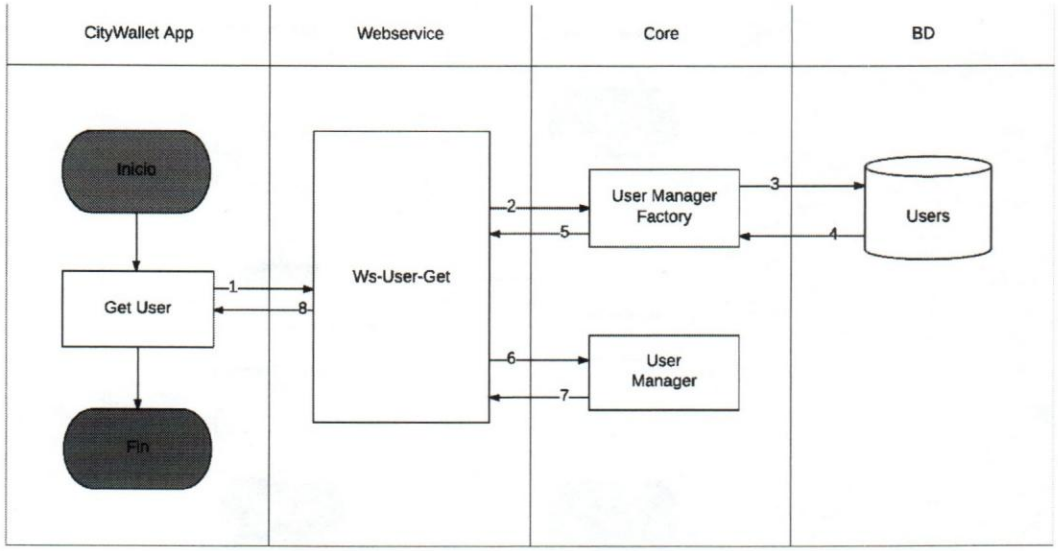

Figura 10 Diagrama de proceso de obtener un usuario

Fuente: [Elaboración propia]

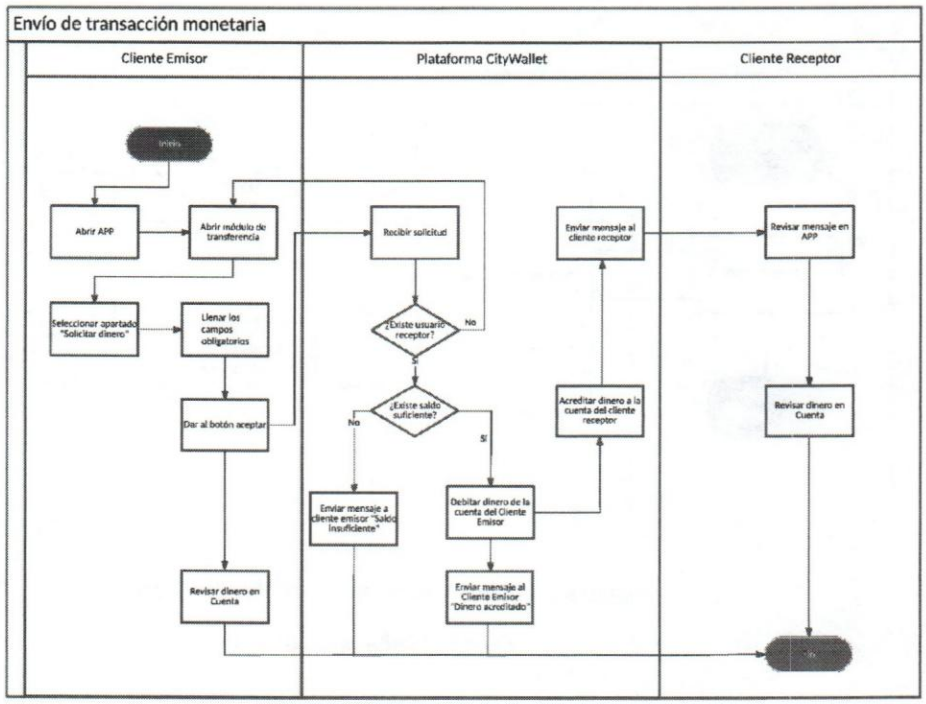

Figura 11 Diagrama de envío de transacción monetaria

Fuente: [Elaboración propia]

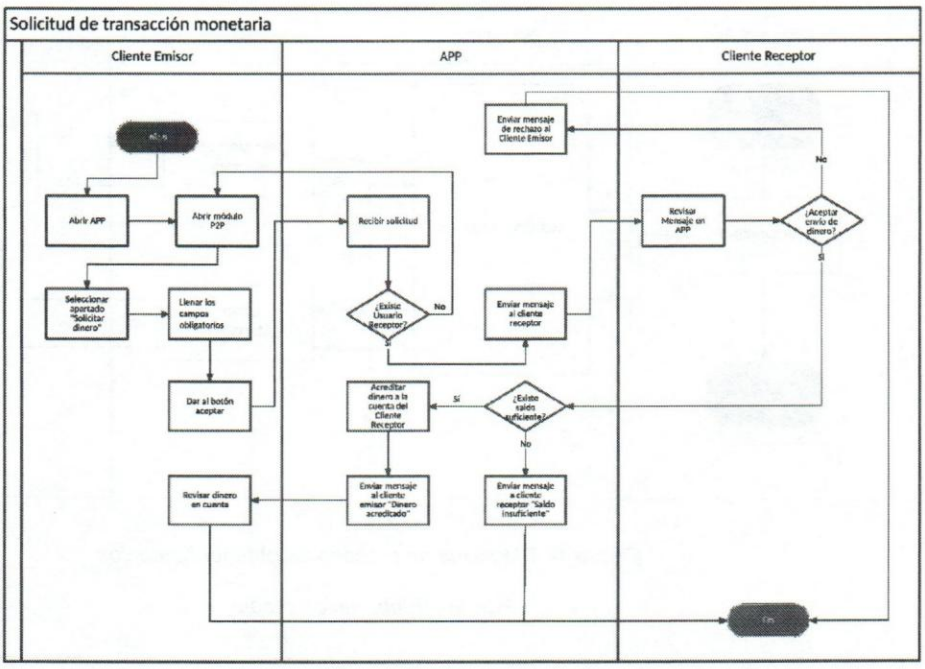

Figura 12 Diagrama de flujo de solicitud de transacción monetaria

Fuente: [Elaboración propia]

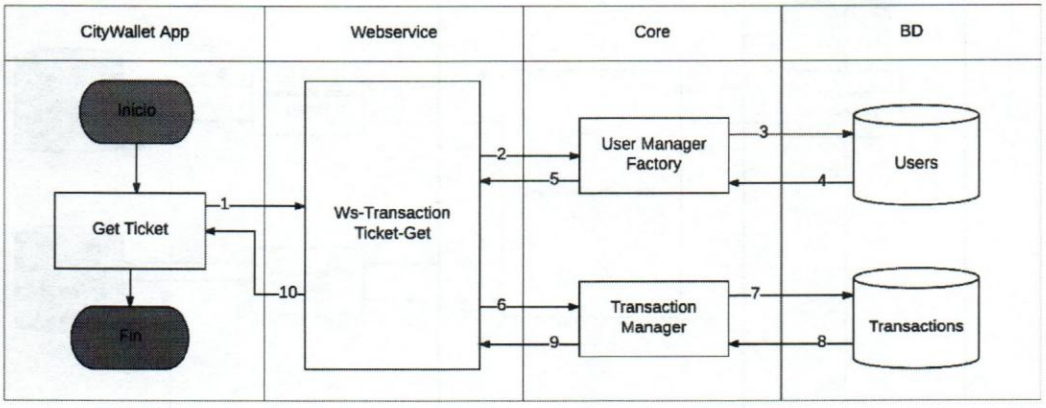

Figura 13 Diagrama de proceso de obtener tickets de transacciones

Fuente: [Elaboración propia]

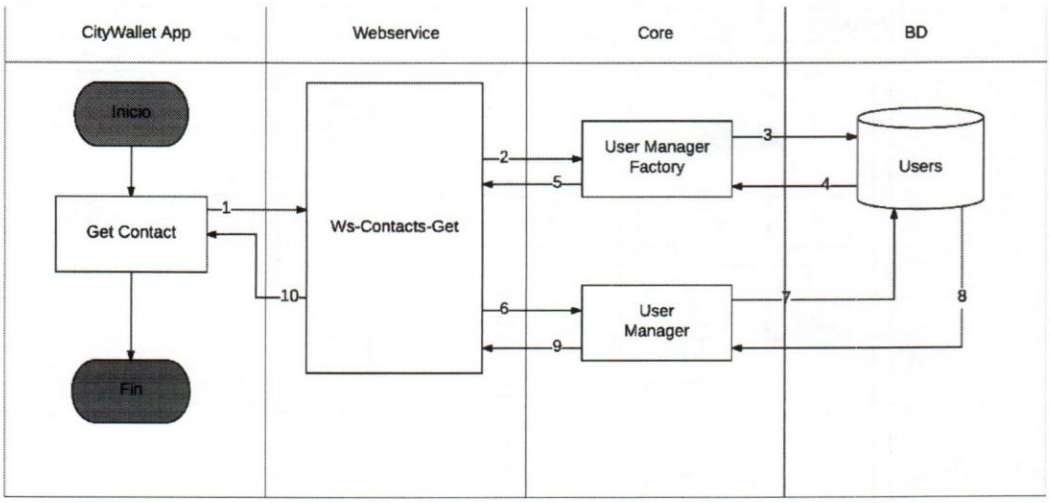

Figura 14 Diagrama de proceso de obtener contactos

Fuente: [Elaboración propia]

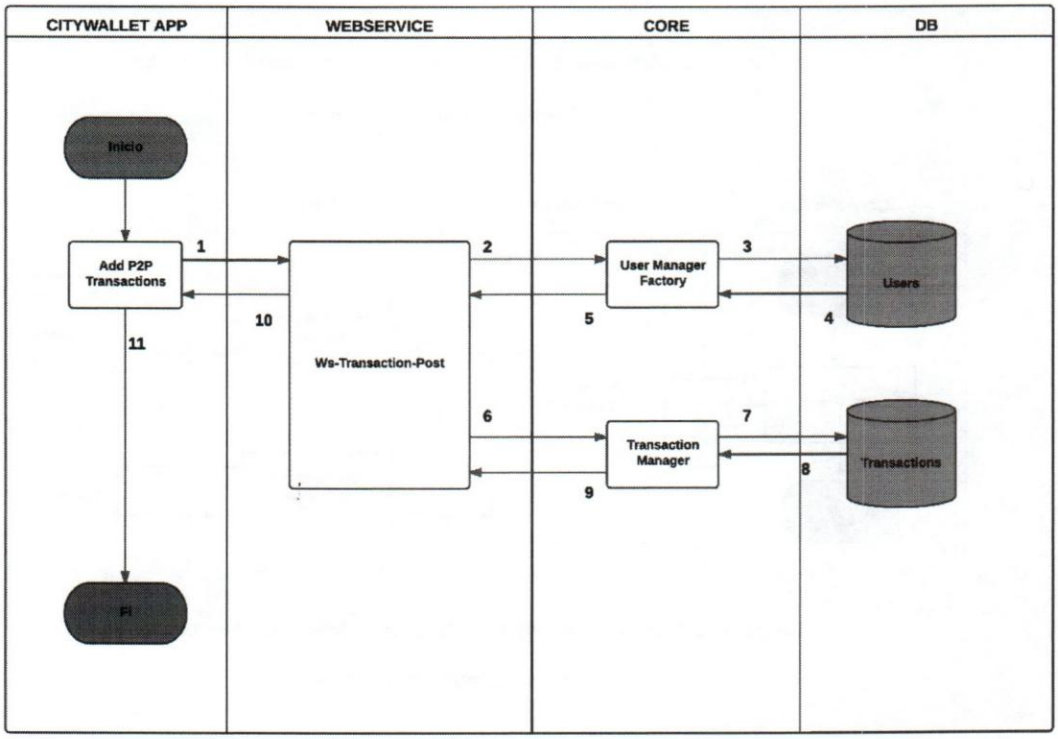

Figura 15 Diagrama de proceso de envío de dinero

Fuente: [Elaboración propia]

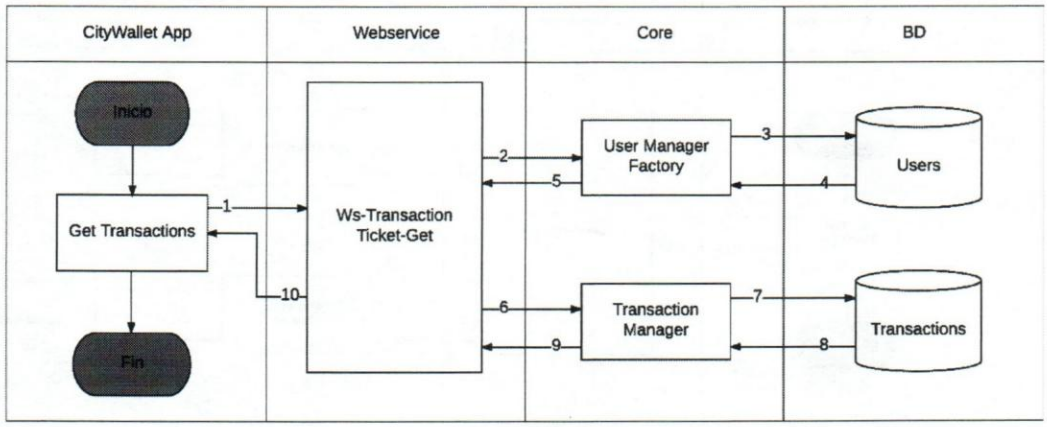

Figura 16 Diagrama de proceso de obtener transacciones

Fuente: [Elaboración propia]

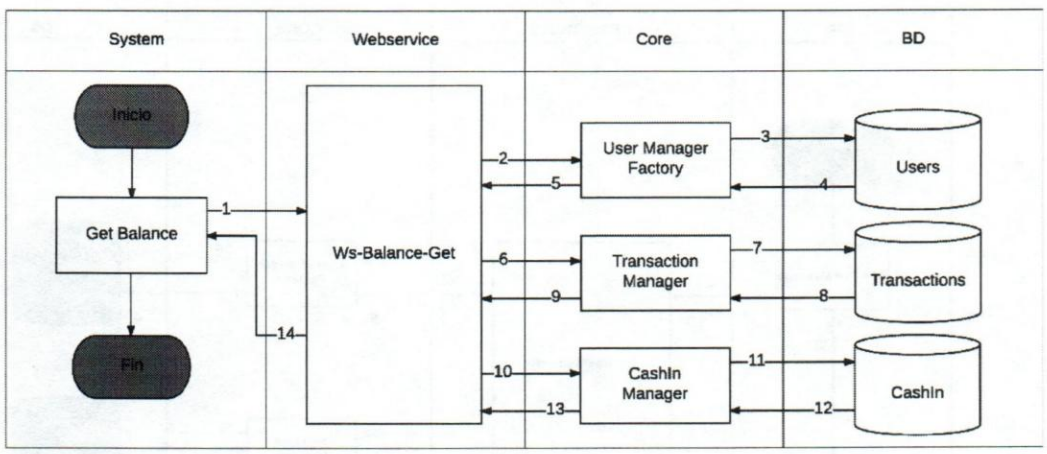

Figura 17 Diagrama de proceso de obtener saldo

Fuente: [Elaboración propia]

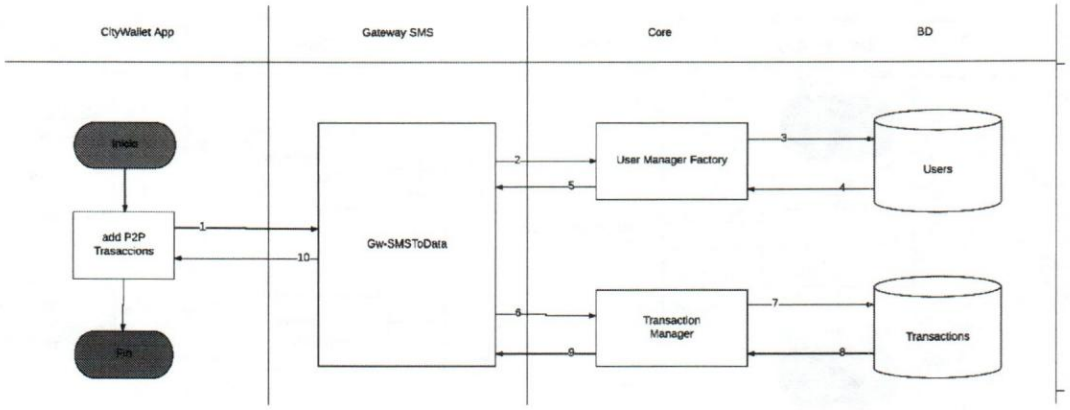

Figura 18 Diagrama de proceso de envío de dinero vía SMS

Fuente: [Elaboración propia]

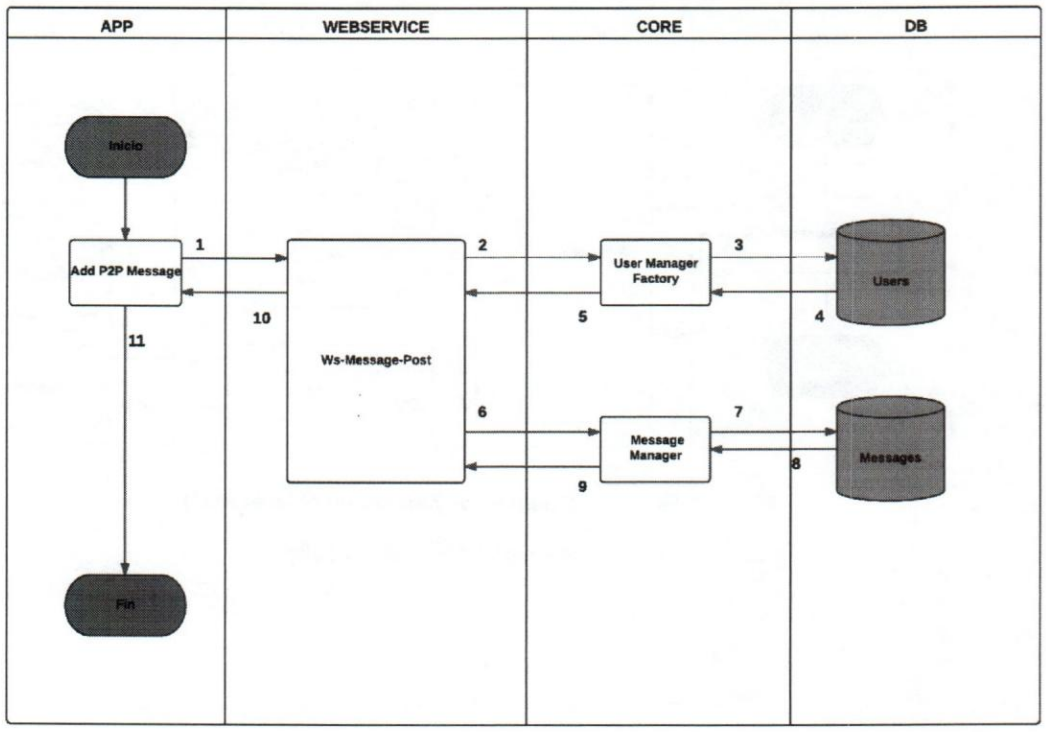

Figura 19 Diagrama de proceso de solicitud de dinero

Fuente: [Elaboración propia]

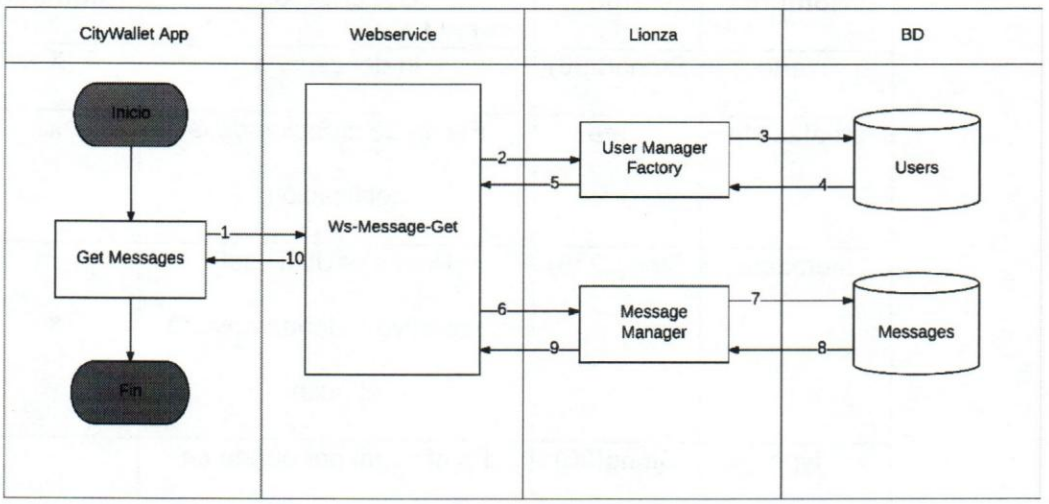

Figura 20 Diagrama de proceso de obtener mensajes

Fuente: [Elaboración propia]

C. Tablas de Base de Datos

Nombre	Tipo	Descripción	Requerido
userId	String(256)	Id autogenerado	x
alias	String(60)	Nombre de usuario	x
createDate	Date	Fecha de creación de la cuenta	x
status	String(60)	Estatus de la cuenta, puede ser activa o inactiva	x
name	String(256)	Nombre del usuario	X
lastName	String(256)	Apellido del usuario	
cognitoID	String(256)	Identificador en el pool de usuarios de cognito	x

Tabla 14 Estructura de tabla users

Fuente: [Elaboración propia]

Nombre	Tipo	Descripción	Requerido
userId	String(256)	Id del usuario	X
createDate	Date	Fecha de creación de la notificación	x
metadata	String(256)	Posee el UUID del dispositivo a donde enviará el push	x
type	String(60)	Es el canal por donde se enviara la notificación, puede ser push o email	x

Tabla 15 Estructura de tabla notificationChannels

Fuente: [Elaboración propia]

Nombre	Tipo	Descripción	Requerido
userId	String(256)	Id del usuario	x
answer	String(256)	Respuesta del usuario encriptada en MD5	x
challenge	String(256)	Pregunta secreta del usuario	x
pin	String(256)	Clave de transacción usuario encriptada en MD5	x

Tabla 16 Estructura de tabla transactionPins

Fuente: [Elaboración propia]

Nombre	Tipo	Descripción	Requerido
transactionId	String(256)	Id de la transacción	x
userIdFrom	String(256)	Id del usuario que envía el dinero	x
userIdTo	String(256)	Id del usuario que recibe el dinero	x
createDate	Date	Fecha de creación de la transferencia	x
status	String(256)	Estatus que se encuentra la transferencia de dinero. Puede ser error o completada	x
type	String(256)	Tipo de transacción, puede ser: recarga, p2p o comisión CW	x
metadata	String(256)	Posee el ticket generado por el servidor y los alias de los usuarios	x

Tabla 17 Estructura de tabla transactions

Fuente: [Elaboración propia]

Nombre	Tipo	Descripción	Requerido
messageID	String(256)	Id del mensaje	x
userIdFrom	String(256)	Id del usuario que envía el mensaje	x
userIdTo	String(256)	Id del usuario que recibe el mensaje	x
body	String(256)	Cuerpo del mensaje	x
status	String(256)	Estatus del mensaje, puede ser: READ o UNREAD	x
createDate	Date	Fecha de creación del mensaje	x
type	String(256)	Tipo de mensaje, puede ser: system, request, transfer	x
metadata	String(256)	Información adicional del mensaje	x

Tabla 18 Estructura de tabla messages

Fuente: [Elaboración propia]

D. AWS Cognito

Nombre	Tipo	Descripción	Requerido
Name	String(256)	Nombre del usuario	x
Family_Name	String(256)	Apellido del usuario	x

Profile	String(256)	Tipo de usuario: Administrador o Cliente	X
City	String(256)	Ciudad del usuario	
State	String(256)	Estado del usuario	
Address	String(256)	Dirección del usuario	
Email	String(256)	Correo electrónico del usuario	X
Phone	String(256)	Número celular del usuario	X
Alias	String(60)	Nombre de usuario	X
Verified	Boolean	El campo toma true si la cuenta fue verificada	X

Tabla 19 Atributos del user pool en AWS Cognito

Fuente: [Elaboración propia]

E. Aplicación móvil

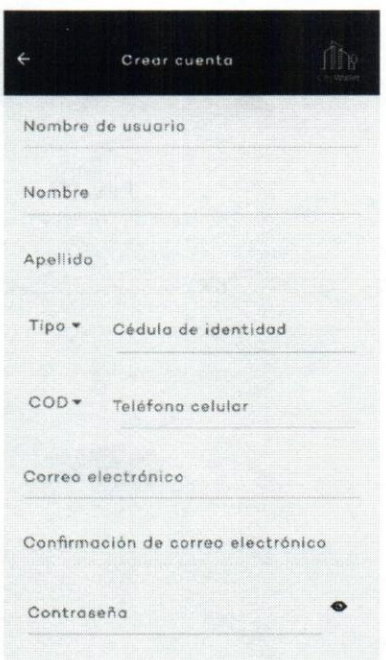

Figura 21 Crear cuenta

Fuente: [Elaboración propia]

Figura 22 Verificar cuenta

Fuente: [Elaboración propia]

Figura 23 Reenvío de código de validación

Fuente: [Elaboración propia]

Figura 24 Inicio de sesión

Fuente: [Elaboración propia].

Figura 25 Reestablecer contraseña

Fuente: [Elaboración propia].

Figura 26 Dashboard

Fuente: [Elaboración propia].

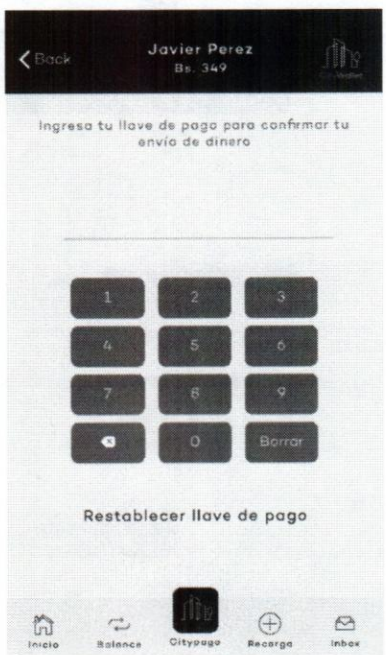

Figura 27 Reestablecer clave de transacción

Fuente: [Elaboración propia].

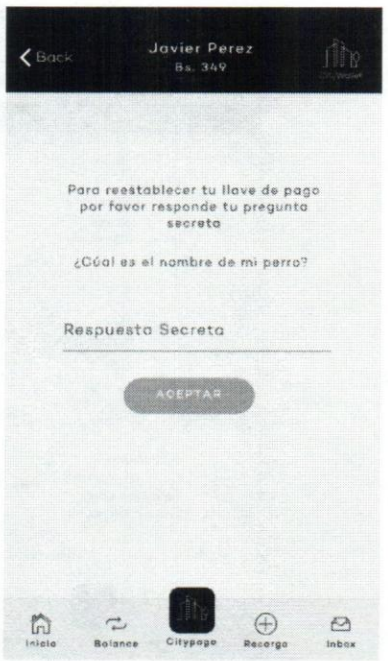

Figura 28 Reestablecer clave de transacción

Fuente: [Elaboración propia].

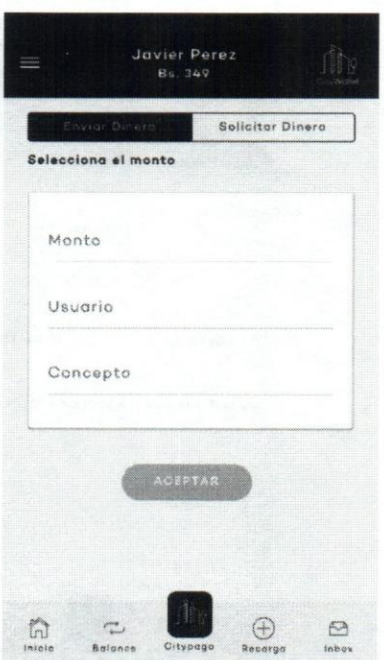

Figura 29 Enviar transacción monetaria

Fuente: [Elaboración propia].

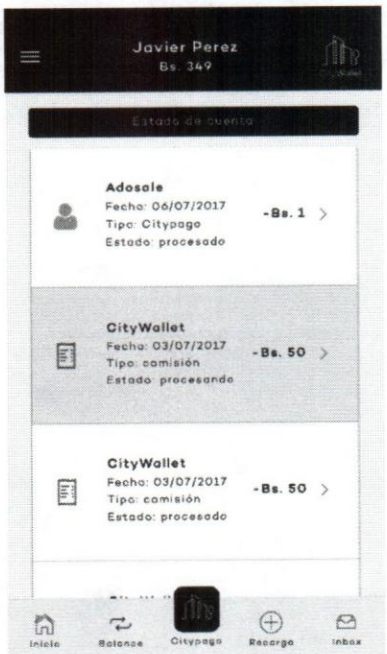

Figura 30 Balance

Fuente: [Elaboración propia].

Figura 31 Recarga de saldo

Fuente: [Elaboración propia].

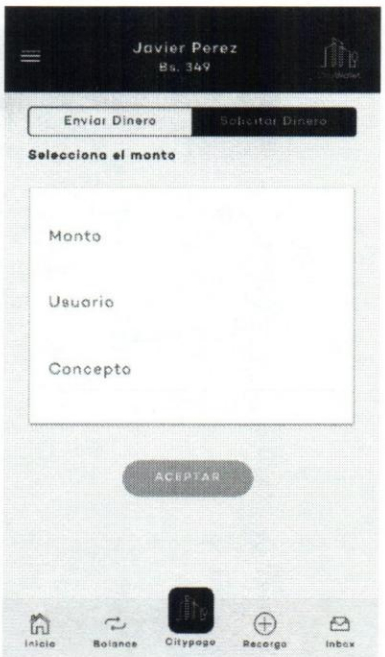

Figura 32 Solicitud de transacción monetaria.

Figura 33 Inbox.

Fuente: [Elaboración propia].

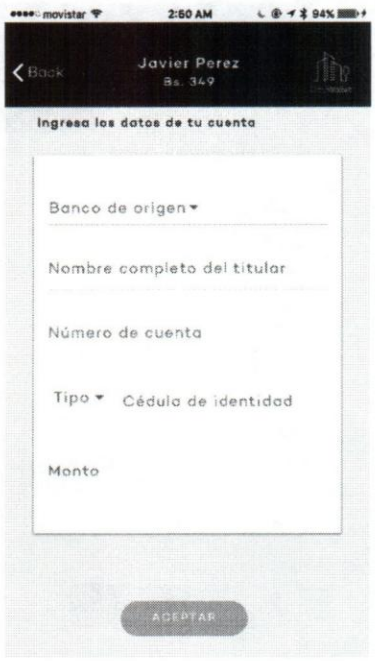

Figura 34 Retiro de dinero

Fuente: [Elaboración propia].

F. Aplicación web

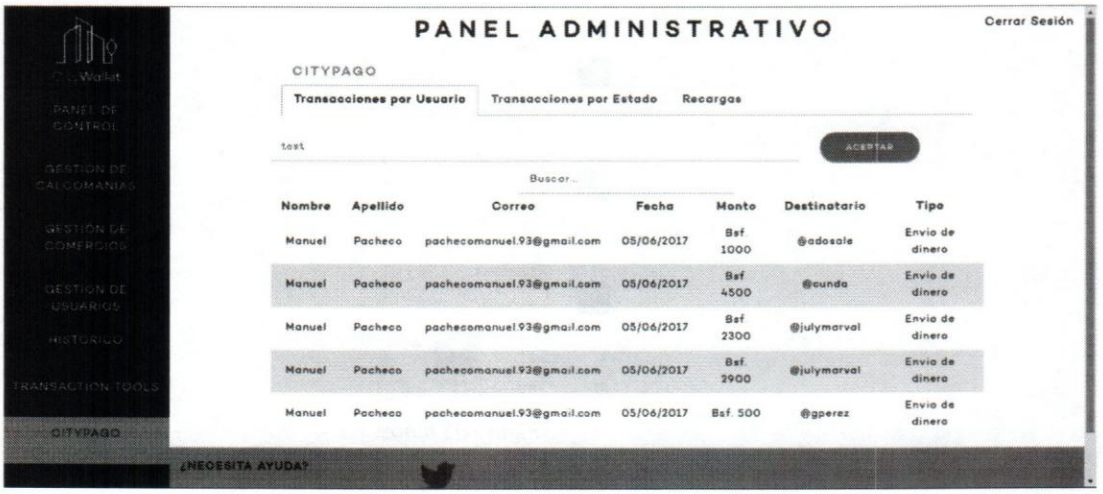

Figura 35 Reporte de transacciones por usuario.

Fuente: [Elaboración propia].

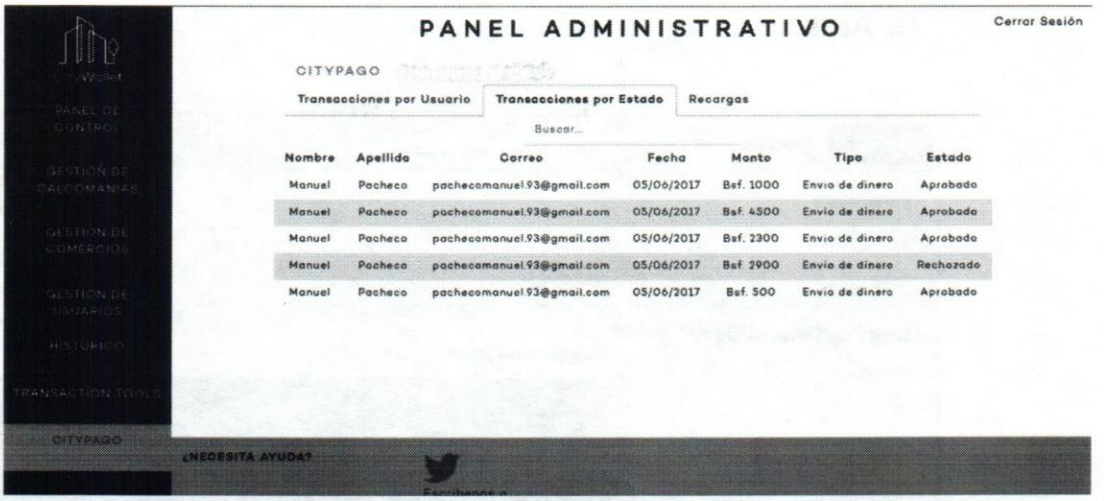

Figura 36 Reporte de transacciones por estado

Fuente: [Elaboración propia].

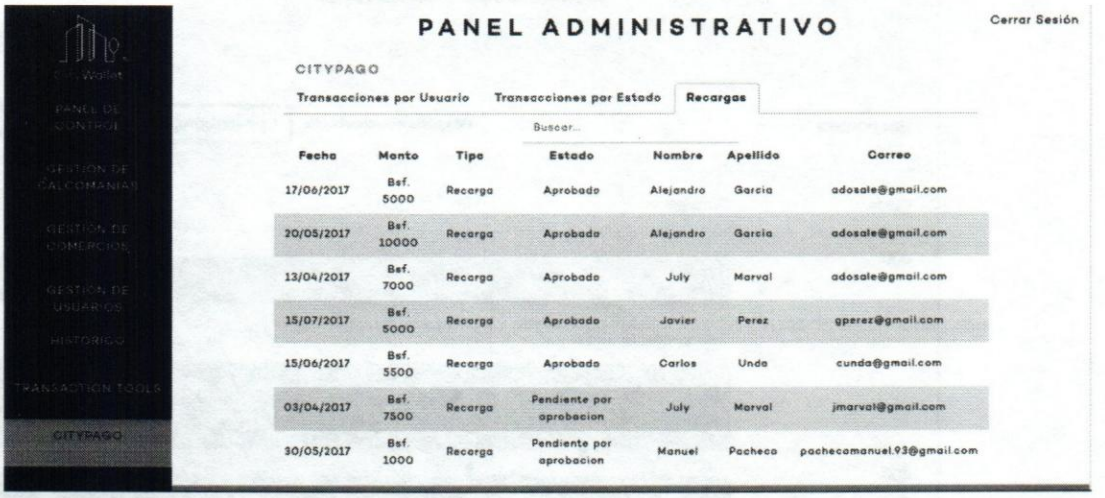

Figura 37 reporte de recargas de saldo

Fuente: [Elaboración propia].

G. APIs.

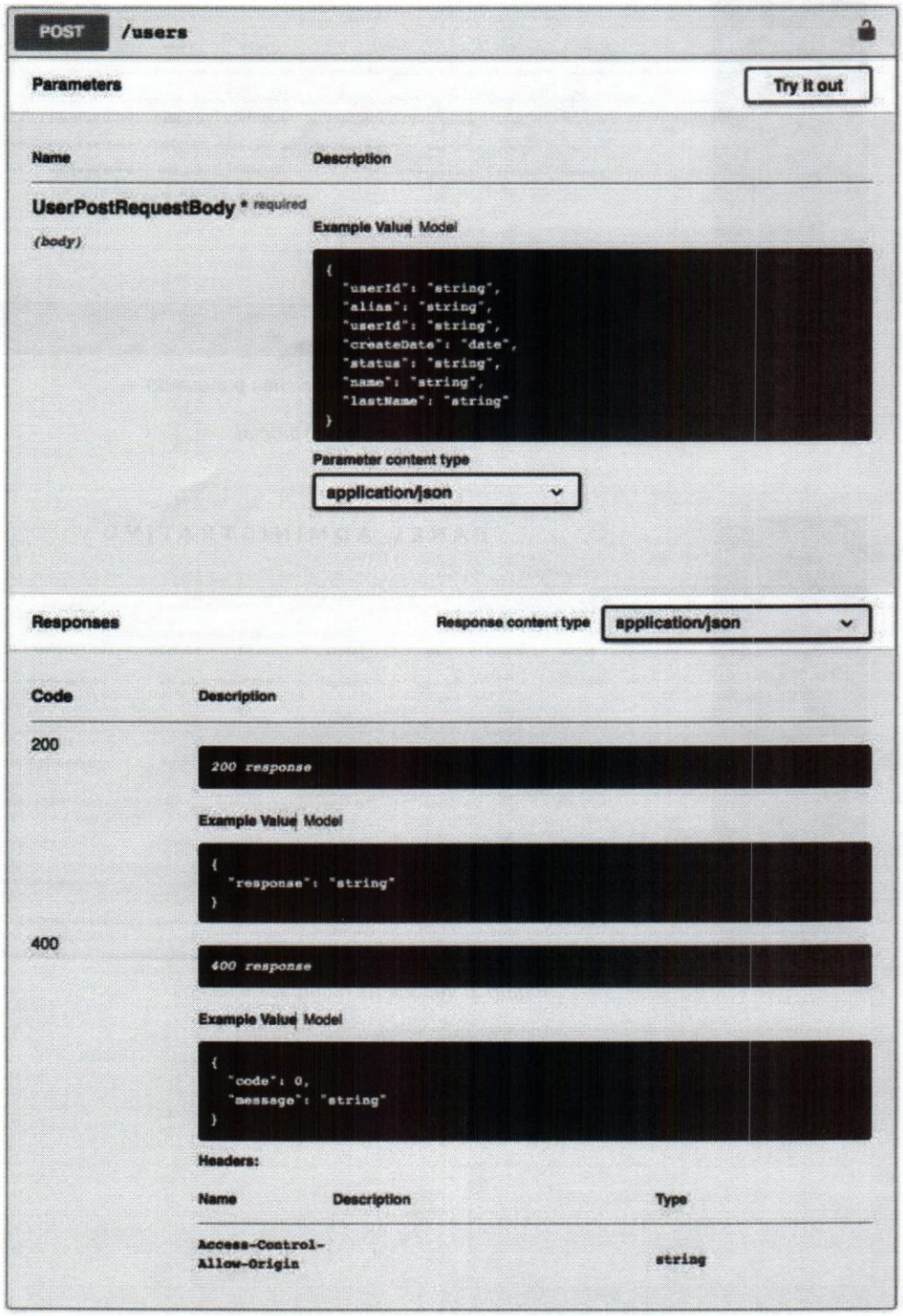

Figura 38 Método post, recurso users

Fuente: [Elaboración propia].

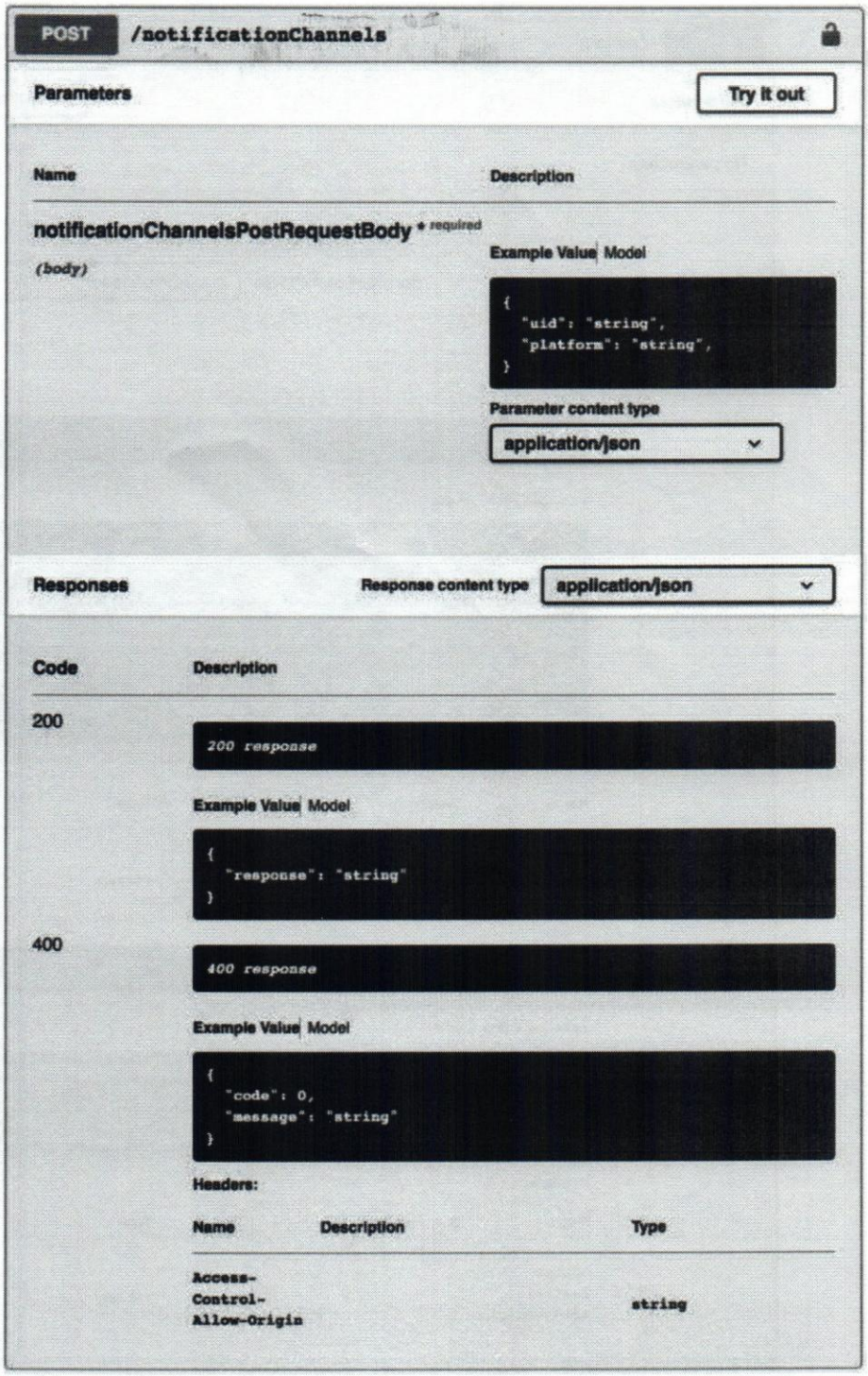

Figura 39 Método post, recurso notificationChannels

Fuente: [Elaboración propia].

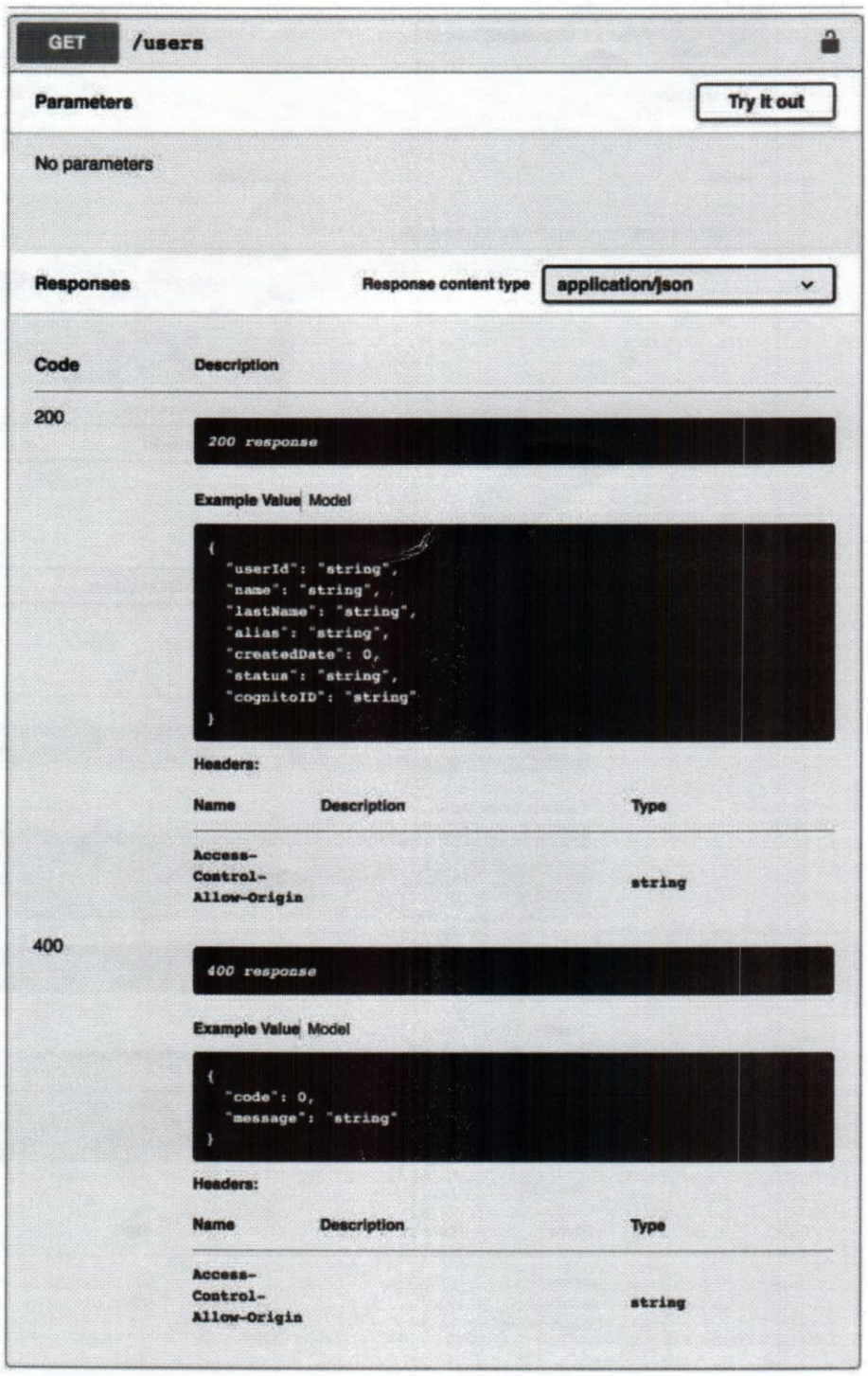

Figura 40 Método get, recurso users

Fuente: [Elaboración propia].

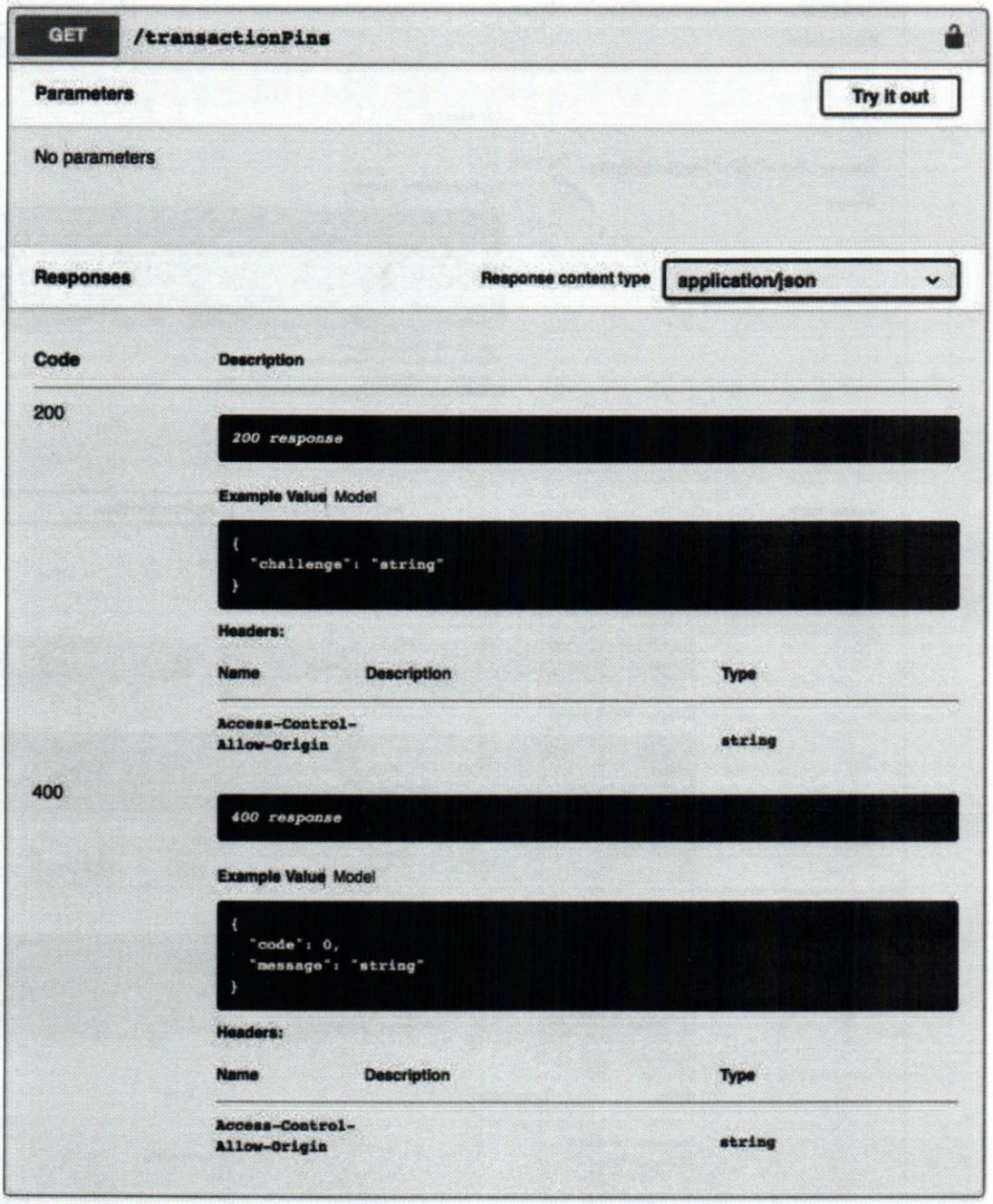

Figura 41 Método post, recurso transactionPins

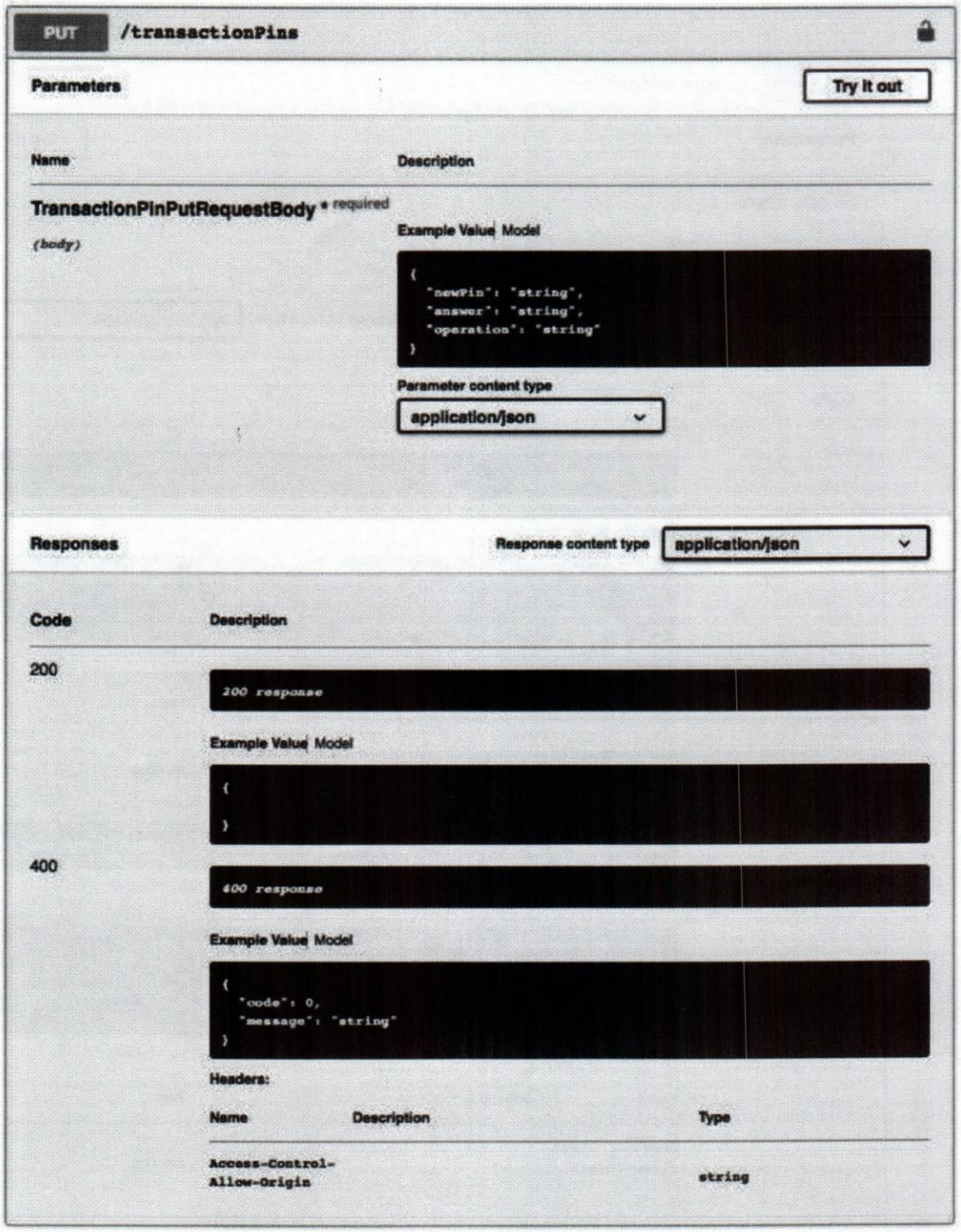

Figura 42 Método put, recurso transactionPins.

Fuente: [Elaboración propia].

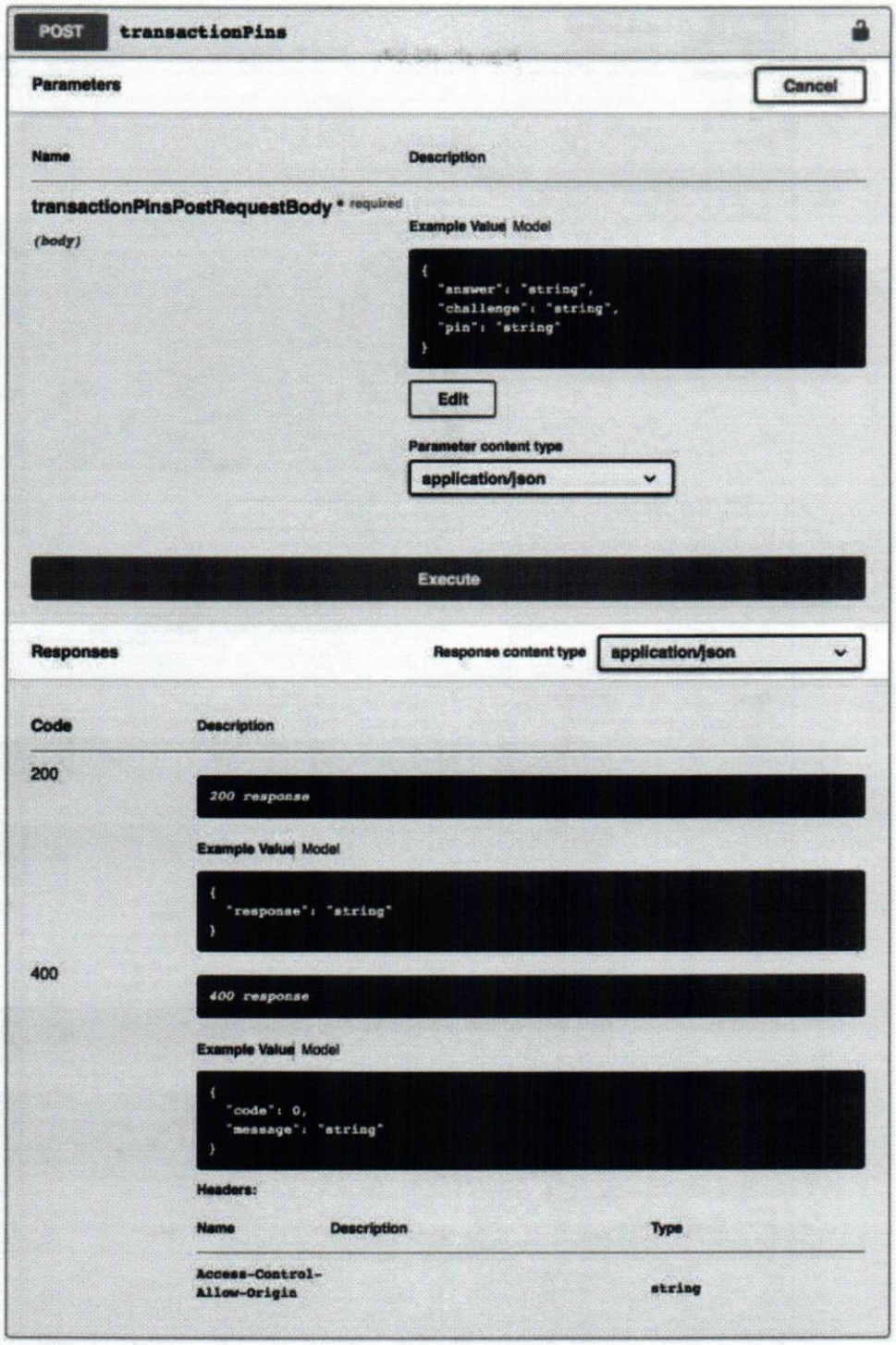

Figura 43 Método post, recurso transactionPins.

Fuente: [Elaboración propia].

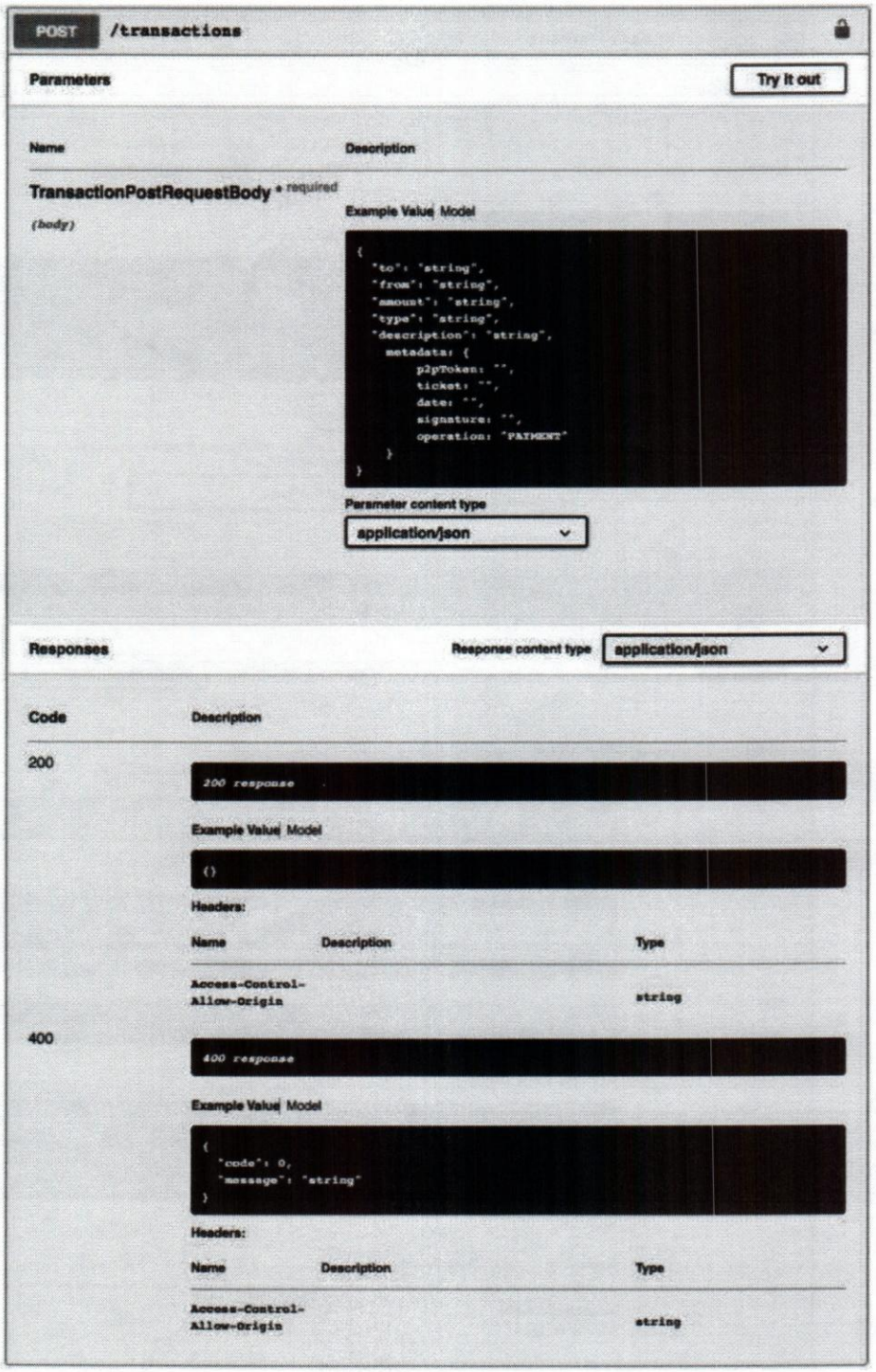

Figura 44 Método post, recurso transactions.

Fuente: [Elaboración propia].

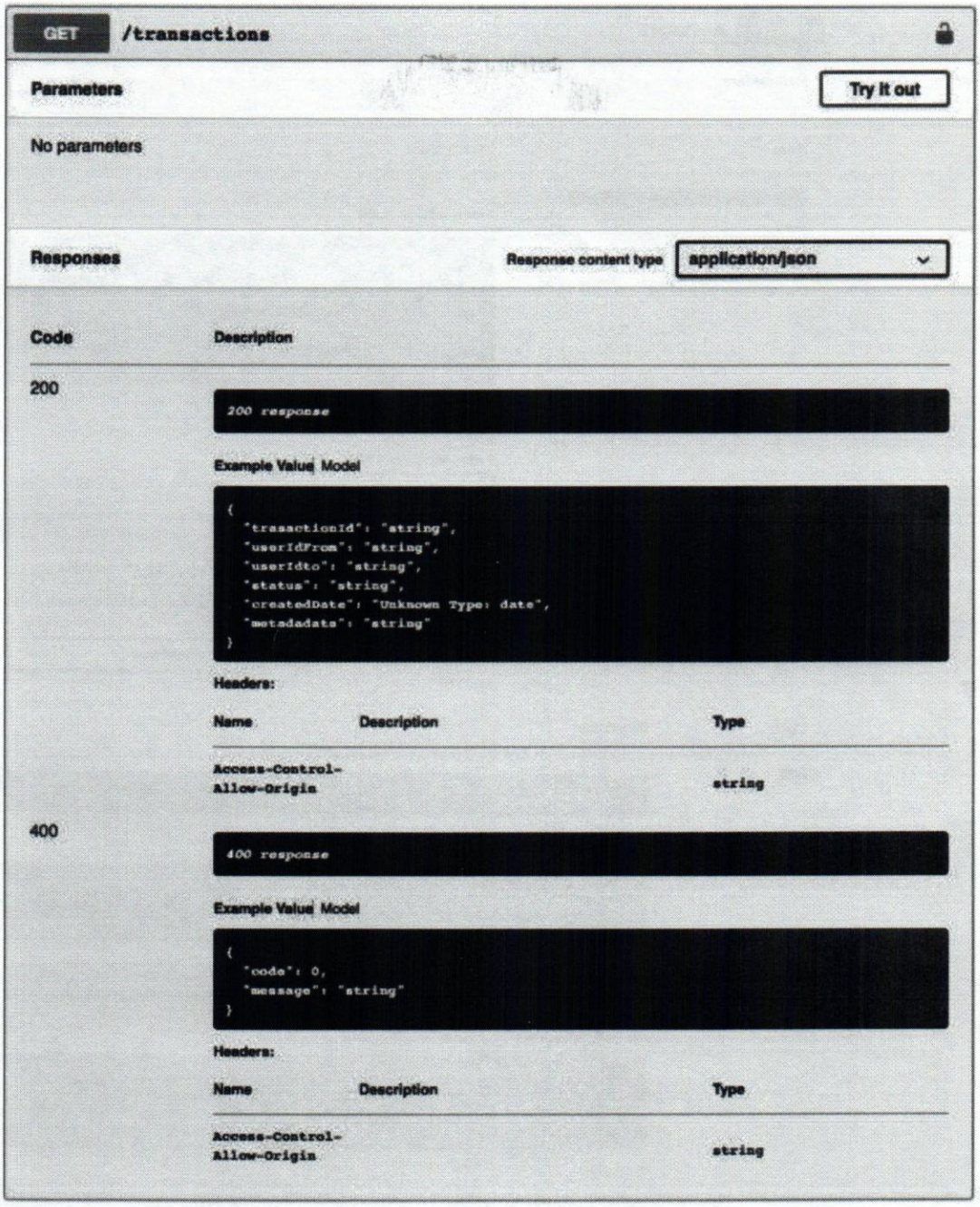

Figura 45 Método get, recurso transactions.

Fuente: [Elaboración propia].

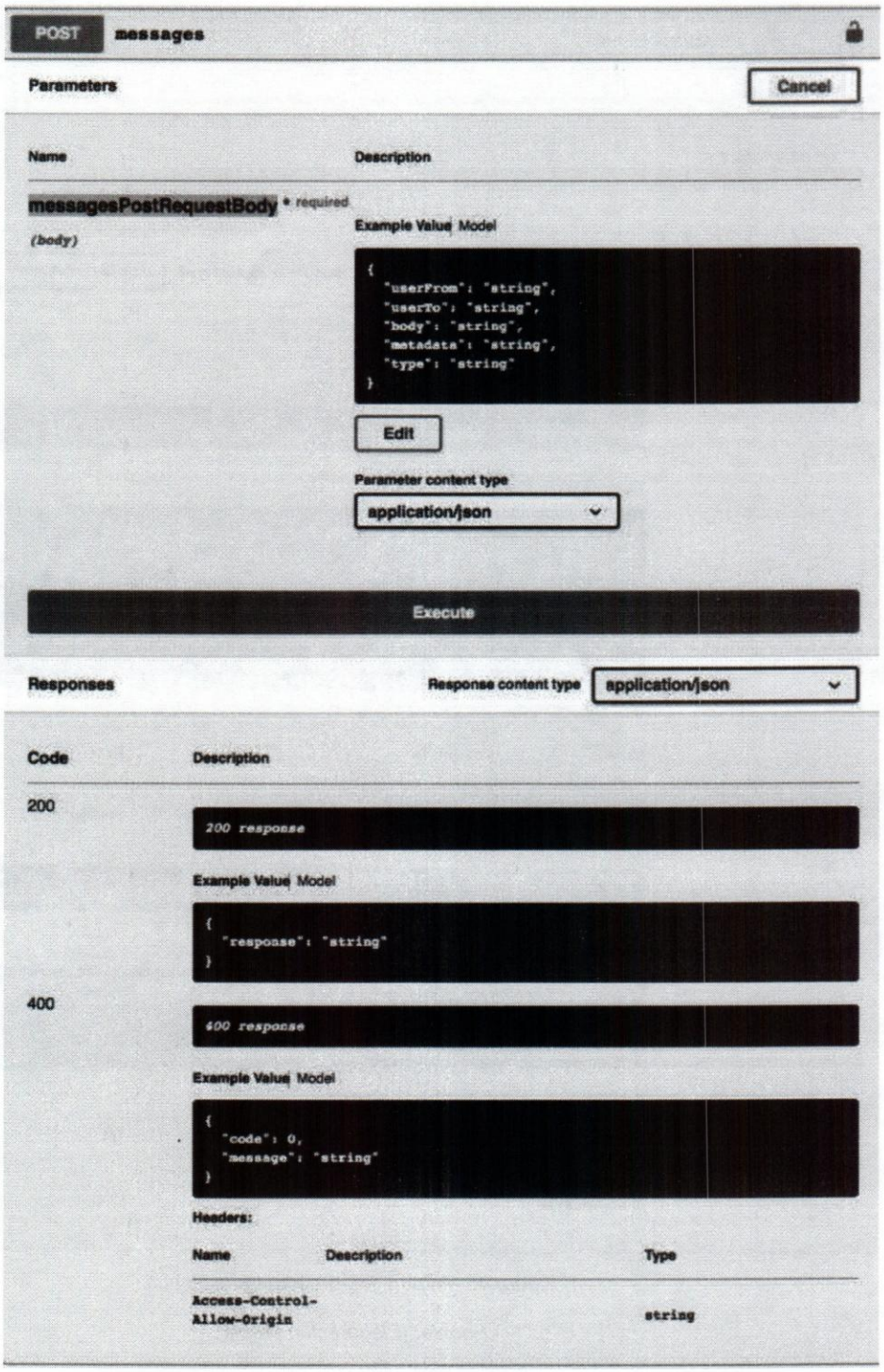

Figura 46 Método post, recurso messages.

Fuente: [Elaboración propia].

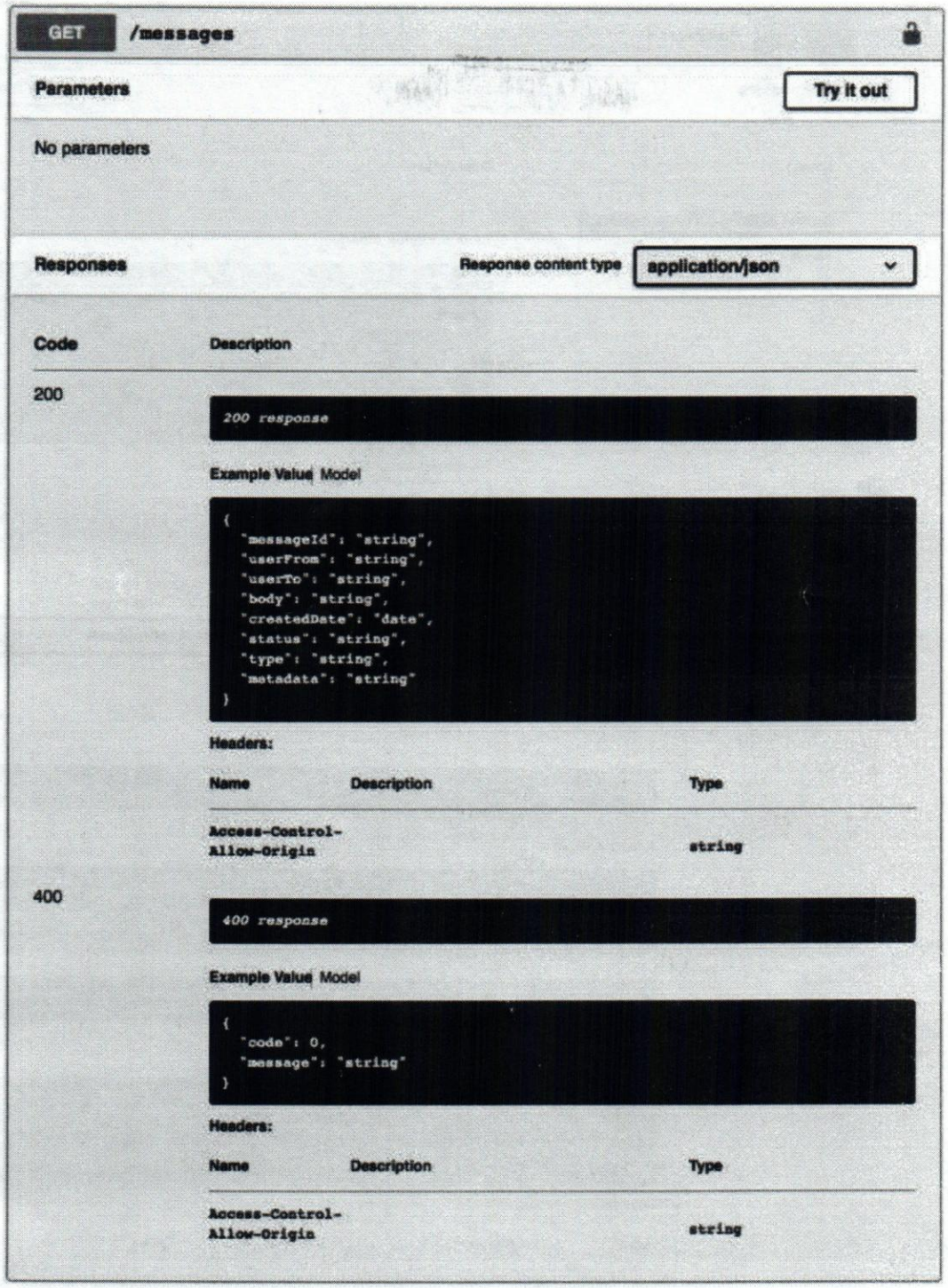

Figura 47 Método get, recurso messages.

Fuente: [Elaboración propia].

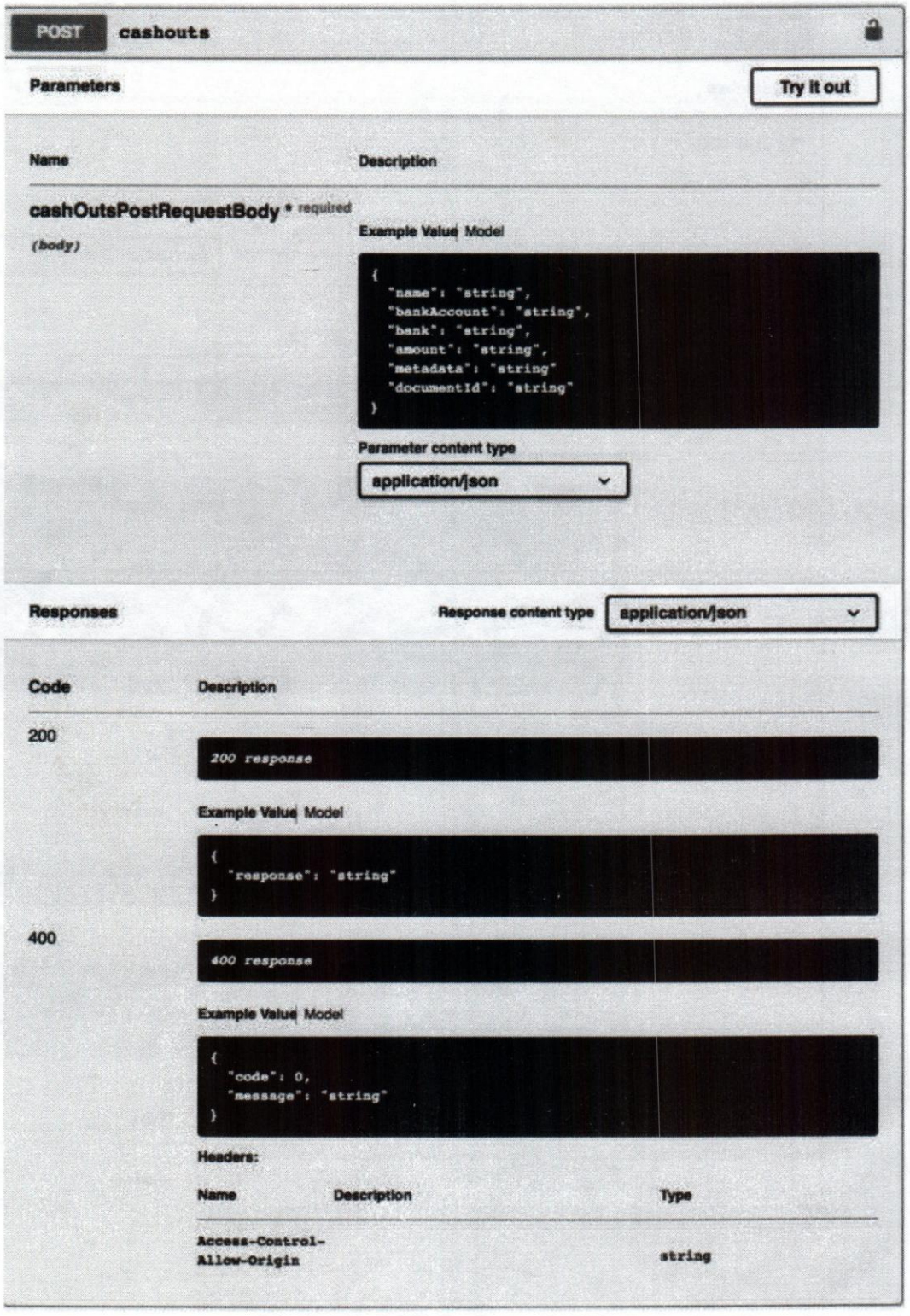

Figura 48 Método post, recurso cashouts

Fuente: [Elaboración propia].