PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

SISTEMA DE INSCRIPCIÓN DE PROPIEDADES USANDO ETHEREUM BLOCKCHAIN

CRISTÓBAL EDUARDO ALLENDES CASTRO

INFORME FINAL DEL PROYECTO
PARA OPTAR AL TÍTULO PROFESIONAL DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor guía: Ricardo Soto De Giorgis

DICIEMBRE, 2018

Dedico el presente proyecto a mi familia y amigos que me apoyaron en esta etapa de mi vida. También a las personas que fui conociendo en el camino, que fueron aportando a mi crecimiento. Además, agradezco a mis profesores guía y co-referente por creer en mí y tener paciencia.

Resumen

Actualmente, la gran mayoría de los servicios ofrecidos en Internet administran sus datos de manera centralizada. Un ejemplo de este tipo de prestaciones es el que brindan los conservadores de bienes raíces, que hasta hace muy poco realizaban su trabajo en libros. Esto genera que la información almacenada esté constantemente comprometida a distintos ataques. Cabe señalar que los notarios participan en el proceso de compraventa de una propiedad generando altos costos para las personas, al tener que desembolsar una cantidad significativa de dinero a los dos entes nombrados.

Esta investigación propone una aplicación web de inscripción de propiedades que utiliza la Blockchain de Ethereum como remedio a los problemas nombrados anteriormente. Esta tecnología tiene la capacidad de almacenar datos de manera descentralizada y brinda de inmutabilidad a los mismos. Por un lado, dará solución a la centralización de los datos, ya que se utilizará esta red P2P para guardar la información. Por otro, la plataforma viene a reemplazar las funciones realizadas por el conservador de bienes raíces y el notario.

Como resultado se obtiene una plataforma web familiar al usuario en donde puede mantener la inscripción de sus propiedades sin la intervención de un tercero. En la aplicación, al almacenar los datos de manera descentralizada en la Blockchain se otorgará seguridad al usuario, al ser este el real dueño de su información. Por último, se genera un cambio en la manera de realizar los procesos correspondientes a un bien raíz y un impacto a nivel económico en la sociedad.

Palabras-clave: *Internet, Blockchain, Ethereum, P2P.*

Abstract

Currently, the vast majority of services offered on the Internet manage their data centrally. An example of this type of provision is provided by real estate conservatives, who until recently did their work in books. This generates that the information stored is constantly committed to different attacks. It should be noted that notaries participate in the process of buying and selling a property generating high costs for people, having to disburse a significant amount of money to the two entities named.

This research proposes a web application of property registration that uses the Blockchain of Ethereum as a remedy to the problems mentioned above. This technology has the ability to store data in a decentralized manner and provides immutability to them. On the one hand, it will provide a solution to the centralization of the data, since this P2P network will be used to store the information. On the other, the platform comes to replace the functions performed by the real estate conservator and the notary.

As a result, a user-friendly web platform is obtained where you can maintain the registration of your properties without the intervention of a third party. In the application, by storing the data in a decentralized manner in the Blockchain, the user will be granted security, since this is the real owner of his information. Finally, a change is generated in the way of carrying out the processes corresponding to a real estate and an economic impact on society.

Keywords: *Internet, Blockchain, Ethereum, P2P.*

Índice

Resumen ..... **i**

Abstract ..... **ii**

Lista de Figuras ..... **vi**

Lista de Tablas ..... **x**

1 Introducción **1**

2 Definición de Objetivos **5**

2.1 Objetivo General **5**
2.2 Objetivos Específicos **5**

3 Estado del Arte **6**

4 Marco Teórico **8**

4.1 SHA-256 ..... **8**
4.2 Árbol de Merkle **8**
4.3 Transacciones **9**
4.4 Bloques **11**
4.5 Mineros **11**

5 Documentación de la Aplicación **12**

5.1 Definición y Especificación de Requerimientos **12**
5.1.1 Definición General del Proyecto 12
5.1.2 Especificación de Requerimientos 12
5.1.2.1 Requisitos Funcionales 12
5.1.2.2 Alcance 17
5.3.1 Especificación de Procedimientos 18
5.2 Arquitectura del Sistema **19**
5.2.1 Capa Front-end 19
5.2.2 Capa Back-end API 20
5.2.3 Capa Back-end P2P 20
5.2.4 Puente de Comunicación 20
5.2.5 APIs Simuladas 20
5.2.6 APIs Externas 21

5.3 Diseño del Modelo de Datos **21**
5.4 Procesos y Servicios Ofrecidos **22**
5.4.1 Paso 1: Ingresar la Propiedad al Sistema 22
5.4.2 Paso 2: Poner la Propiedad a la Venta 22
5.4.3 Paso 3: Interesado se Contacta con el Propietario 23
5.4.4 Paso 4: Agregar Contrato de Compraventa 24
5.4.5 Paso 5: Dar Permiso para Firmar al Comprador 24
5.4.6 Paso 6: Firmar el Contrato de Compraventa 25
5.4.7 Paso 7: Realizar Custodia Bancaria 25
5.4.8 Paso 8: Ingresar Custodia Bancaria 26
5.4.9 Paso 9: Finalizar Venta de Propiedad 26
5.4.10 Paso 10: Solicitar Devolución de Custodia Bancaria 27
5.5 Documentación Técnica **28**

6 Conclusiones **32**

Referencias **33**

Anexos .....

A: Capturas de Pantalla del Sistema .....
A.1: Registro de Usuario .....
A.2: Menú de Inicio de la Plataforma .....
A.3: Funciones Creador .....
A.4: Funciones Tasador .....
A.5: Funciones Usuario General .....
A.6: Funciones Propietario .....
A.7: Funciones Comprador .....
B: Flujo de Datos .....
B.1: Flujo de Datos Registro de Usuario .....
B.2: Flujo de Datos Creador .....
B.3: Flujo de Datos Tasador .....
B.4: Flujo de Datos Usuario General .....
B.5: Flujo de Datos Propietario .....
B.6: Flujo de Datos Comprador .....
B.7: Flujo de Datos Propietario y Comprador .....
C: Código de las Funciones del Sistema .....

C.1: Registro de Usuario.....
C.2: Funciones Creador .....
C.3: Funciones Tasador .....
C.4: Funciones Usuario General.....
C.5: Funciones Propietario .....
C.6: Funciones Comprador.....
C.7: Funciones Propietario y Comprador .....
C.8: Funciones de Consulta .....
C.9: Funciones Internas .....

Lista de Figuras

Figura 1.1: Estructura de la Cadena de Bloques 2
Figura 4.1: Ejemplo de Árbol de Merkle Binario 9
Figura 4.2: Esquema de Transacción en Bitcoin 10
Figura 4.3: Estructura de los Bloques 11
Figura 5.1: Caso de Uso Registro Usuario 13
Figura 5.2: Caso de Uso Creador Contrato 13
Figura 5.3: Caso de Uso Usuario General 14
Figura 5.4: Caso de Uso Propietario 15
Figura 5.5: Caso de Uso Comprador 16
Figura 5.6 Caso de Uso Tasador 17
Figura 5.7: EDT de la Plataforma 17
Figura 5.8: Metodología de Desarrollo de la Plataforma 18
Figura 5.9: Arquitectura del Sistema 19
Figura 5.10: Modelo de Datos Back-end P2P 21
Figura 5.11: Modelo de Datos Back-end API 21
Figura 5.12: Diagrama de Flujo Agregar Propiedad 22
Figura 5.13: Diagrama de Flujo Poner Propiedad a la Venta 23
Figura 5.14: Diagrama de Flujo Ver Propiedad 23
Figura 5.15: Diagrama de Flujo Agregar Contrato de Compraventa 24
Figura 5.16: Diagrama de Flujo Dar Permiso Firmar Contrato 25
Figura 5.17: Diagrama de Flujo Firmar Contrato de Compraventa 25
Figura 5.18: Diagrama de Flujo Ingresar Custodia Bancaria 26
Figura 5.19: Diagrama de Flujo Finalizar Venta de Propiedad 27
Figura 5.20: Diagrama de Flujo Solicitar Devolución de Custodia Bancaria 28
Figura 5.21: Función Agregar Usuario 28
Figura A.1: Pantalla Registro de Usuario Validación .....
Figura A.2: Pantalla de Registro de Usuario Validado .....
Figura A.3: Pantalla Menú de Inicio de la Plataforma .....
Figura A.4: Pantalla Agregar Tasador .....

Figura A.5: Pantalla Cambiar Clave del Sistema Paso 1 .....
Figura A.6: Pantalla Cambiar Clave del Sistema Paso 2 .....
Figura A.7: Pantalla Ver Balance.....
Figura A.8: Pantalla Retirar Balance .....
Figura A.9: Pantalla Cambiar Valor Propiedad .....
Figura A.10: Pantalla Agregar Propiedad Paso 1.....
Figura A.11: Pantalla Agregar Propiedad Paso 2.....
Figura A.12: Pantalla Ver Propiedad .....
Figura A.13: Pantalla Ver Información de Venta .....
Figura A.14: Pantalla Ver Precio Trámite.....
Figura A.15: Pantalla Actualizar Precio Trámite.....
Figura A.16: Pantalla Poner Propiedad a la Venta.....
Figura A.17: Pantalla Cancelar Venta de Propiedad.....
Figura A.18: Pantalla Agregar Contrato de Compraventa .....
Figura A.19: Pantalla Dar Permiso Firmar Venta.....
Figura A.20: Pantalla Finalizar Venta.....
Figura A.21: Pantalla Dar Permiso Ver Venta.....
Figura A.22: Pantalla Ver Estado Venta .....
Figura A.23: Pantalla Transferir Propiedad .....
Figura A.24: Pantalla Firmar Contrato de Compraventa .....
Figura A.25: Pantalla Agregar Custodia Bancaria.....
Figura A.26: Pantalla Solicitar Devolución Custodia Bancaria.....
Figura A.27: Pantalla Dar Permiso Ver Venta.....
Figura A.28: Pantalla Ver Estado Venta .....
Figura B.1: Flujo de Datos Registro de Usuario .....
Figura B.2: Flujo de Datos Agregar Tasador .....
Figura B.3: Flujo de Datos Cambiar Clave del Sistema .....
Figura B.4: Flujo de Datos Ver Balance .....
Figura B.5: Flujo de Datos Retirar Balance .....
Figura B.6: Flujo de Datos Cambiar Valor Propiedad.....
Figura B.7: Flujo de Datos Agregar Propiedad.....

Figura B.8: Flujo de Datos Ver Propiedad.....
Figura B.9: Flujo de Datos Ver Información Venta.....
Figura B.10: Flujo de Datos Ver Precio Trámite .....
Figura B.11: Flujo de Datos Actualizar Precio Trámite .....
Figura B.12: Flujo de Datos Poner Propiedad a la Venta .....
Figura B.13: Flujo de Datos Cancelar Venta Propiedad.....
Figura B.14: Flujo de Datos Agregar Descripción Venta.....
Figura B.15: Flujo de Datos Dar Permiso Firmar Venta .....
Figura B.16: Flujo de Datos Finalizar Venta .....
Figura B.17: Flujo de Datos Transferir Propiedad.....
Figura B.18: Flujo de Datos Firmar Venta Propiedad .....
Figura B.19: Flujo de Datos Agregar Custodia Bancaria .....
Figura B.20: Flujo de Datos Solicitar Devolución Custodia .....
Figura B.21: Flujo de Datos Dar Permiso Ver Venta .....
Figura B.22: Flujo de Datos Ver Estado Venta .....
Figura C.1: Registro de Usuario .....
Figura C.2: Agregar Tasador.....
Figura C.3: Cambiar Clave Sistema.....
Figura C.4: Ver Balance.....
Figura C.5: Retirar Balance .....
Figura C.6: Cambiar Valor Propiedad .....
Figura C.7: Agregar Propiedad .....
Figura C.8: Ver Propiedad .....
Figura C.9: Ver Propietario.....
Figura C.10: Ver Descripción Venta.....
Figura C.11: Ver Precio Trámite.....
Figura C.12: Actualizar Precio Trámite.....
Figura C.13: Poner en Venta Propiedad.....
Figura C.14: Cancelar Venta Propiedad.....
Figura C.15: Agregar Descripción Venta.....
Figura C.16: Agregar Permiso Para Firmar Venta.....

Figura C.17: Finalizar Venta de Propiedad.....
Figura C.18: Transferencia Propiedad .....
Figura C.19: Firmar Compra Propiedad.....
Figura C.20: Agregar Documento Custodia.....
Figura C.21: Devolver Pago Custodia .....
Figura C.22: Dar Permiso Temporal Venta .....
Figura C.23: Ver Estado Venta .....
Figura C.24: Soy en Custodia .....
Figura C.25: Soy Creador .....
Figura C.26: Soy Tasador .....
Figura C.27: Soy Propietario.....
Figura C.28: Soy Usuario.....
Figura C.29: Get Nombre Usuario .....
Figura C.30: Get Apellido Paterno Usuario .....
Figura C.31: Get Apellido Materno Usuario.....
Figura C.32: Get Rut Usuario .....
Figura C.33: Get Digito Usuario .....
Figura C.34: Soy en Venta .....
Figura C.35: Rut esta Registrado .....
Figura C.36: Agregar Venta.....
Figura C.37: Agregar Custodia .....
Figura C.38: Get Número Propietario .....

Lista de Tablas

Tabla 4.1: Ejemplo de Resultados de SHA-256 8
Tabla 5.1: Herramientas Utilizadas en el Proceso de Desarrollo 18
Tabla 5.2: Documentación Funciones Creador 29
Tabla 5.3: Documentación Funciones Tasador 29
Tabla 5.4: Documentación Funciones Usuario General Parte 1 29
Tabla 5.5: Documentación Funciones Usuario General Parte 2 30
Tabla 5.6: Documentación Funciones Propietario 30
Tabla 5.7: Documentación Funciones Comprador 30
Tabla 5.8: Documentación Funciones Vendedor y Comprador 31
Tabla 5.9: Documentación Funciones Consulta del Sistema 31
Tabla 5.10: Documentación Funciones Internas 31
Tabla B.1: Explicación Flujo de Datos Registro de Usuario .....
Tabla B.2: Explicación Flujo de Datos Agregar Tasador .....
Tabla B.3: Explicación Flujo de Datos Cambiar Clave del Sistema .....
Tabla B.4: Explicación Flujo de Datos Ver Balance .....
Tabla B.5: Explicación Flujo de Datos Retirar Balance .....
Tabla B.6: Explicación Flujo de Datos Cambiar Valor Propiedad .....
Tabla B.7: Explicación Flujo de Datos Agregar Propiedad .....
Tabla B.8: Explicación Flujo de Datos Ver Propiedad .....
Tabla B.9: Explicación Flujo de Datos Ver Información Venta .....
Tabla B.10: Explicación Flujo de Datos Ver Precio Trámite .....
Tabla B.11: Explicación Flujo de Datos Actualizar Precio Trámite .....
Tabla B.12: Explicación Flujo de Datos Poner Propiedad a la Venta .....
Tabla B.13: Explicación Flujo de Datos Cancelar Venta Propiedad .....
Tabla B.14: Explicación Flujo de Datos Agregar Descripción Venta .....
Tabla B.15: Explicación Flujo de Datos Dar Permiso Firmar Venta .....
Tabla B.16: Explicación Flujo de Datos Finalizar Venta .....
Tabla B.17: Explicación Flujo de Datos Transferir Propiedad .....
Tabla B.18: Explicación Flujo de Datos Firmar Venta Propiedad .....
Tabla B.19: Explicación Flujo de Datos Agregar Custodia Bancaria .....

Tabla B.20: Explicación Flujo de Datos Solicitar Devolución Custodia.....
Tabla B.21: Explicación Flujo de Datos Dar Permiso Ver Venta.....
Tabla B.22: Explicación Flujo de Datos Ver Estado Venta.....

1 Introducción

En el mundo la gran mayoría de los servicios que encontramos en Internet, como por ejemplo Facebook, tienen toda su información guardada en bases de datos centralizadas [1]. Al no estar distribuida y manejarse de manera central lo hace susceptible a fallos, dado que si algo le ocurre al servidor central afectará considerablemente al sistema. Es por esta razón que el sistema es susceptible a ser atacado y tenga pérdidas tanto de información como de confiabilidad. Además, la probabilidad del daño físico que pueda sufrir el hardware es muy alta.

Un claro ejemplo de este tipo de servicios es el administrado por los conservadores de bienes raíces los que hasta hace poco tiempo realizaban su trabajo en libros y actualmente utilizan sistemas web centralizados [2][3]. Además, estas entidades realizan distintos cobros por sus labores, siendo la inscripción de una propiedad la más relevante. Esto genera conflictos en la integridad de los datos, por las razones explicadas en el párrafo anterior, y problemas de características económicas, ya que el trámite nombrado no es un monto fijo, sino un porcentaje del valor del bien raíz [4].

Por otro lado, los notarios también participan en el proceso cuando hacen de intermediario en el contrato de compraventa de una propiedad. Este tercero de confianza que valida el documento realiza cobros excesivos por su trabajo, según indica la Fiscalía Nacional Económica [5]. Generando así elevados gastos por realizar las acciones correspondientes a las descritas en este párrafo y el anterior.

En el mercado de los notarios se generan distintas irregularidades que producen cobros excesivos por distintas razones, las que se revelan en un estudio realizado por la Fiscalía Nacional Económica [5], dentro de estas se mencionan la poca competencia y una normativa muy restrictiva en cuanto al funcionamiento. En el Código Orgánico de Tribunales se indica que los notarios son los únicos que pueden ejercer como conservadores de bienes raíces [6] generando los mismos vicios mencionados. Con esto el problema económico se acrecienta aún más para el común de las personas y beneficiando a unos pocos.

Para solucionar el primero de los problemas que se describen anteriormente se requiere usar un sistema distribuido en vez de uno centralizado para evitar diferentes tipos de ataques a los datos de la plataforma. Al usar redes peer-to-peer se generan ciertas ventajas como el bajo costo en cuanto a hardware, la elevada disponibilidad de la información y alta confiabilidad del sistema. Las características que presenta este tipo de aplicaciones la hacen más robusta y la probabilidad de dejarla fuera de funcionamiento es mínima [7].

En cuanto a el problema económico se deberá reemplazar tanto al conservador de bienes raíces como al notario por un sistema que realice las tareas de estos terceros validadores, pero con un costo ínfimo para las personas, esto en comparación con lo que se paga actualmente. Aquello se puede lograr dado que las tareas que realizan estos entes son simples de implementar, sólo sería necesario otorgar validez a los contratos de compraventa que se almacenarán en la plataforma. Esto se puede conseguir ya que existe una tecnología

descentralizada que lo logra y un framework JavaScript que se conecta a la misma ofreciendo una interfaz web familiar.

En concreto la aplicación mantendrá un registro de propiedades utilizando la Blockchain de Ethereum lo que permite la compraventa de las mismas en el software. Además, se utiliza al Registro Civil, el Servicio de Impuestos Internos y los Bancos como certificadores de los datos que se desean ingresar. De esta forma se mantiene la información guardada de manera descentralizada y reemplazamos dentro del proceso a los terceros involucrados mencionados anteriormente.

Al almacenar todos los datos en la cadena de bloques, incluidos los documentos, estos serán públicos y válidos instantáneamente por lo que no se necesitará de un notario. Los enormes gastos realizados normalmente se reemplazarán por pequeños cobros que el sistema utiliza para poder procesar la información que se ingresa a la aplicación y un pago en el trámite de transferencia de dominio de una propiedad. En definitiva, las características que presenta la plataforma permiten dar solución a los problemas que se plantean en los párrafos anteriores generando un impacto social que beneficia económicamente a los usuarios.

Como menciona el autor la base de la aplicación propuesta es la tecnología Blockchain que ha tomado mucha fuerza en los últimos años y ha sido tildada de disruptiva. El precursor y quien propuso este concepto es Satoshi Nakamoto con Bitcoin en el año 2009 [8]. La cadena de bloques es, a grandes rasgos, una base de datos distribuida que almacena transacciones en estructuras llamadas bloques, diseñados para evitar la modificación de la información que se ingresa al sistema.

En Blockchain los bloques están encadenados formando una estructura con forma de árbol y no de hilera como se podría pensar al escuchar su nombre. Esto es porque dos de estos pueden estar en el mismo nivel al tener el mismo padre. Además, tienen una cadena principal, usada por los mineros, que es la más larga como indica la Figura 1.1 o en caso de igualdad de altura es la que mayor dificultad total posea.

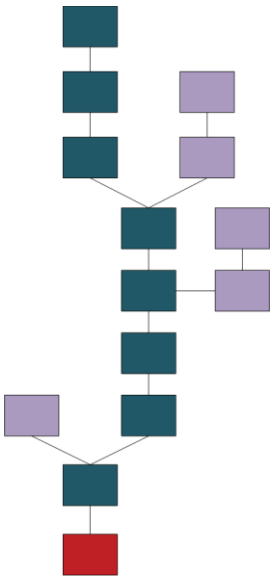

Figura 1.1: Estructura de la Cadena de Bloques

Dentro de Blockchain las transacciones son la manera de transferir valor entre los usuarios. El emisor A indica la dirección (hash de la clave pública) del destinatario B y el monto que se quiere enviar firmando la misma con su clave privada. El sistema validará si la operación realizada por A es correcta, de ser así B recibirá la cantidad de criptomoneda que corresponde, acción que se ve reflejada en el saldo de este último.

Anteriormente el autor menciona el concepto de bloques, cuya función principal es guardar una cierta cantidad de transacciones ordenadas cronológicamente. Además de lo señalado una de estas estructuras almacena una marca de tiempo, una dificultad, un campo de 32 bits llamado NONCE, el cual es el único no constante, y el hash del bloque anterior. Este último campo es el hash de la combinación de todos los campos del bloque anterior y es vital para el funcionamiento de Blockchain junto con el NONCE y los mineros.

Ethereum [9] utiliza Blockchain, mencionado en párrafos anteriores, en su funcionamiento con un sistema de Turing completo integrado. Esto permite ofrecer una plataforma de programación de aplicaciones descentralizadas mediante contratos inteligentes. Esta aplicación ocupa una divisa interna, el Ether, usada para el procesamiento de transacciones dentro de la red, pero además se puede utilizar a modo de criptomoneda como Bitcoin o cualquier otra.

En Ethereum se utilizan dos tipos de cuentas, las de propiedad externa y las de contratos inteligentes. Las primeras son las administradas por los usuarios que pueden realizar transferencias y ejecutar código dentro de la red. Las segundas son códigos de programas descentralizados las que reciben transacciones para ejecutar sus funciones y almacenar información.

En el contexto de los contratos inteligentes para poder realizar alguna función se debe pagar a los mineros por la ejecución de código y por almacenar datos. Para ello Ethereum define a los conceptos de GAS, GASPRICE y STARTGAS importantes para evitar los ataques de denegación de servicio. El primero, es el criptocombustible interno y representa el valor que tiene la ejecución de un paso computacional, además por cada byte de datos en una transacción se debe pagar cinco de estas unidades. El segundo, es el precio que va a pagar el emisor por cada paso de cálculo. El último, es la cantidad máxima de pasos computacionales que se le va permitir ejecutar.

Ethereum al ser un sistema aislado del Internet que conocemos necesita de un sistema que lo comunique con el mundo exterior. Para ello existe Truffle, un framework de JavaScript puro que mediante librerías incluidas junto con la extensión MetaMask permite la conexión de un servicio web con la red de Ethereum [10]. Con esto se logra tener un acceso a los contratos inteligentes mediante una interfaz amigable y con las ventajas de ambos mundos.

MetaMask es una extensión de navegadores de Internet que proporciona un nodo Ethereum funcionando permanentemente en el mismo [11]. Así, con este puente de comunicación, el servicio web se podrá comunicar con la red Ethereum utilizando las librerías de JavaScript. Con esto la plataforma podrá almacenar la información necesaria de forma segura además de poder mostrarla al usuario de manera simple.

Como resultado de la combinación de la tecnología descentralizada de Ethereum y la interfaz familiar que ofrece Truffle se obtiene un lugar en donde los usuarios podrán realizar la

inscripción de propiedades a un bajo costo sin depender de un conservador de bienes raíces. Además, se podrá realizar la compraventa de las propiedades ingresadas en el sistema sin la necesidad de utilizar los servicios de un notario. Por último, la información que se ingresará a la plataforma será previamente validada por el Registro Civil, el Servicio de Impuestos Internos y el Banco.

En el presente trabajo el autor comienza exhibiendo los objetivos que se definieron para poder cumplir a cabalidad con la realización de la plataforma. Prosigue informando sobre la literatura que hay disponible de aplicaciones que utilizan Ethereum en su funcionamiento principal. Después se plantean las bases de Blockchain definiendo distintos conceptos importantes que sostienen esta tecnología y las características que se generan. Posteriormente se establece la documentación de la plataforma abarcando las generalidades de esta. Para finalizar se dan a conocer las conclusiones del presente documento.

2 Definición de Objetivos

2.1 Objetivo General

El objetivo principal de este proyecto es el desarrollo de un sistema de inscripción de propiedades usando la Blockchain de Ethereum que genere una nueva manera de realizar las acciones derivadas de un bien raíz. Además, esta aplicación debe producir un impacto económico y social para los usuarios que usen el servicio. Por último, la plataforma debe llegar a ser sencilla de utilizar para las personas que deseen registrar su propiedad.

2.2 Objetivos Específicos

Para poder cumplir con el objetivo general el autor plantea unos más específicos, los que en su conjunto permitirán completar el primero. Estos últimos se detallan a continuación:

1. Investigar y comprender la tecnología Blockchain.
2. Estudiar y entender el sistema Ethereum y su lenguaje de programación Solidity.
3. Investigar alguna tecnología que conecte una aplicación web con la red de Ethereum.
4. Realizar análisis y especificación de requerimientos de la plataforma.
5. Realizar diseño de la aplicación.
6. Realizar el desarrollo de las distintas capas del sistema.
7. Realizar las distintas pruebas para comprobar el funcionamiento correcto del sistema.

3 Estado del Arte

En el mundo de Ethereum varios autores han abordado distintas problemáticas a solucionar con contratos inteligentes. Una de ellas fue la realizada por Christoph Jentzsch en su plataforma creada en el año 2016 [12], una organización autónoma digital descentralizada. Esta aplicación llamada The DAO fue desarrollada en Solidity con el objetivo de automatizar el gobierno organizacional y la toma de decisiones.

Para su funcionamiento The DAO necesita capital en Ether, el cual se reúne a través de aportes a cambio de tokens o acciones que otorgan a su titular derechos de voto y propiedad. En esta etapa de creación de la organización si no se alcanza la meta mínima de recaudación en el tiempo establecido se devolverá todo el Ether a sus dueños. Además, las propiedades de los tokens son de libre transferencia en el sistema cuando la fase descrita finaliza con éxito.

Las personas dueñas de tokens de DAO pueden presentar propuestas en forma de contratos inteligentes para poder usar el Ether de la organización, éstas se deben debatir y votar en un tiempo determinado. Si hay quorum y la mayoría de los votos están a favor de la proposición, la aplicación automáticamente transferirá dinero al contrato propuesto para que realice sus actividades; en caso contrario se rechazará la iniciativa y se cerrará. Para que no se genere spam de propuestas se exige un depósito mínimo al momento de la creación de la misma, el cual se devuelve si se logra el número mínimo de votantes.

Uno de los problemas que aborda la aplicación mencionada es que, si una persona tiene más del 50% de los tokens, esta presente una propuesta para enviar todo el Ether a sí mismo; y como siempre tendrá la mayoría de votos todas sus proposiciones serán aprobadas. Para ello la DAO se puede dividir en dos cuando una o más personas no estén de acuerdo con alguna propuesta y quieran recuperar el Ether invertido en la plataforma. Además, se elige a una persona (Conservador) encargada de controlar una lista de usuarios los cuales pueden recibir criptomoneda con el objetivo de evitar el abuso de poder de los mayores inversores; cuando se divide la organización se escoge un nuevo Conservador mediante una votación.

DAO fue atacada en junio del 2016 a menos de dos meses de su lanzamiento en la Blockchain de Ethereum por una persona que explotó una vulnerabilidad del sistema. La mencionada está relacionada con un error de código que aprovecho el atacante para obtener el equivalente a 50 millones de dólares para luego publicar en internet que todo lo realizado seguía el código de la aplicación y que si le retiraban sus Ether los llevaría a tribunales. En consecuencia, de lo mencionado DAO tuvo que cesar su funcionamiento, ya que no se pudo identificar el responsable del ataque por la poca información que es posible recaudar en este tipo de tecnología.

Otra plataforma es Gnosis realizada por el equipo de la empresa de mismo nombre en el año 2017 [13] con el propósito de generar un lugar para crear aplicaciones de mercado de predicción descentralizadas que permita el libre flujo de información útil. Ésta estimulará el cambio en distintos mercados como lo son las finanzas, los juegos de azar, los seguros y la información. Esta aplicación requiere de muchas tecnologías trabajando para su funcionamiento siendo los contratos inteligentes una de ellas formando una capa importante como se presenta a continuación.

Gnosis está formada por tres capas siendo la primera, el núcleo, la fundamental y más importante en el uso de la aplicación proporcionando los contratos inteligentes que son la base del funcionamiento del negocio. La segunda es la de servicios en donde se implementarán toda clase de servicios de mercado predictivo mediante plantillas y herramientas de personalización. La tercera capa es la de aplicaciones, que contiene principalmente interfaces orientadas a algún mercado de predicción en específico o segmento de clientes. Además, esta última pretende brindar un ambiente en donde convivan aplicaciones creadas tanto por Gnosis como por terceros.

Dentro de los principales objetivos que tiene Gnosis están el construir la herramienta de pronóstico más eficiente del mundo, crear el Google de la búsqueda personalizada de información y convertirse en el estándar para los activos predictivos [13]. Por otro lado, esta plataforma se puede utilizar para aplicaciones en el ámbito de instrumentos financieros, instrumentos de seguros y cobertura, información, gobernanza, incentivación y apuestas deportivas.

Como se ha apreciado en esta sección mediante los ejemplos descritos Ethereum permite la creación de soluciones a distintos problemas utilizando aplicaciones descentralizadas. Si bien DAO no se pudo mantener en el tiempo debido a errores en el código no quiere decir que sea un mal ejemplo, sino que se debe plantear como aprendizaje de que el código debe estar correcto y que si se pueden hacer aplicaciones que perduren como es el caso de Gnosis. Estas dos nombradas anteriormente son ejemplos documentados de aplicaciones en la Blockchain de Ethereum, pero hay muchas otras que no están documentadas públicamente mediante papers y en su conjunto ninguna aborda el tema de administrar propiedades de las tierras. Finalmente, el trabajo presentado en el presente apunta a plantear este tema usando la tecnología nombrada para crear una DApp con este objetivo.

4 Marco Teórico

El sistema propuesto en este documento se realizará mediante contratos inteligentes usados en Ethereum para realizar DApps implementadas en lenguaje de programación Solidity. Por debajo de esta plataforma se encuentra la tecnología que le da vida, Blockchain, que proporciona a la presente de inmutabilidad e incorruptibilidad, además de su naturaleza distribuida que otorga alta disponibilidad al sistema. Es por esto que esta sección se dedicará a explicar los conceptos de cadena de bloques de Ethereum que son la base de la aplicación.

4.1 SHA-256

SHA-256 es una función hash criptográfica del conjunto SHA-2 que permite generar una firma única para uno o un conjunto de elementos y no se puede decodificar, o sea, no se pueden obtener los datos que se han codificado a partir del hash. Esta función produce un hash de 64 dígitos hexadecimales aparentemente aleatorio y casi singular de un tamaño fijo de 256 bits independiente de la cantidad y largo de las entradas. Para que una función hash esté bien implementada se debe verificar que dada dos entradas muy parecidas el resultado de esta debe ser muy distinta, como se muestra en la Tabla 4.1.

Tabla 4.1: Ejemplo de Resultados de SHA-256

Entrada	Salida
hola mundo	0b894166d3336435c800bea36ff21b29eaa801a52f584c006c49289a0dcf6e2f
hola mundo!	9b8b048f951b60542c00a3f2754233fb339b60d0cddb5a73025897cf1af4e55f

Dentro de los usos que se le dan a SHA-256 es para ver si algún archivo recibido de otra persona se ha modificado durante el camino comparando al hash del fichero original con el del archivo recibido, además es una buena forma de resumir datos al tener un tamaño fijo como se mencionó en el párrafo anterior. Una de las utilidades en el mundo de Blockchain es en la creación de direcciones o claves públicas, visibles al utilizar alguna billetera de Bitcoin o Ethereum. Otra aplicación en la cadena de bloques se observa en el proceso de minería, más específicamente en la prueba de trabajo que realizan los mineros.

4.2 Árbol de Merkle

El árbol de Merkle es una estructura de datos en árbol, ya sea binario o no, en el que todos los nodos que no son hoja son el resultado de aplicar una función hash (por ejemplo, SHA-256) al valor de sus nodos hijos hasta llegar al nodo raíz del árbol llamado “Merkle root”. Este tipo de estructuras permite asignar un valor hash único a cada elemento de un gran conjunto de datos y de esta manera proveer un método de verificación segura y eficiente de los contenidos almacenados en estas. En Blockchain se utiliza el árbol de Merkle para almacenar en las hojas de esta organización de datos las transacciones de los bloques. La Figura 4.1 presenta lo señalado en este párrafo.

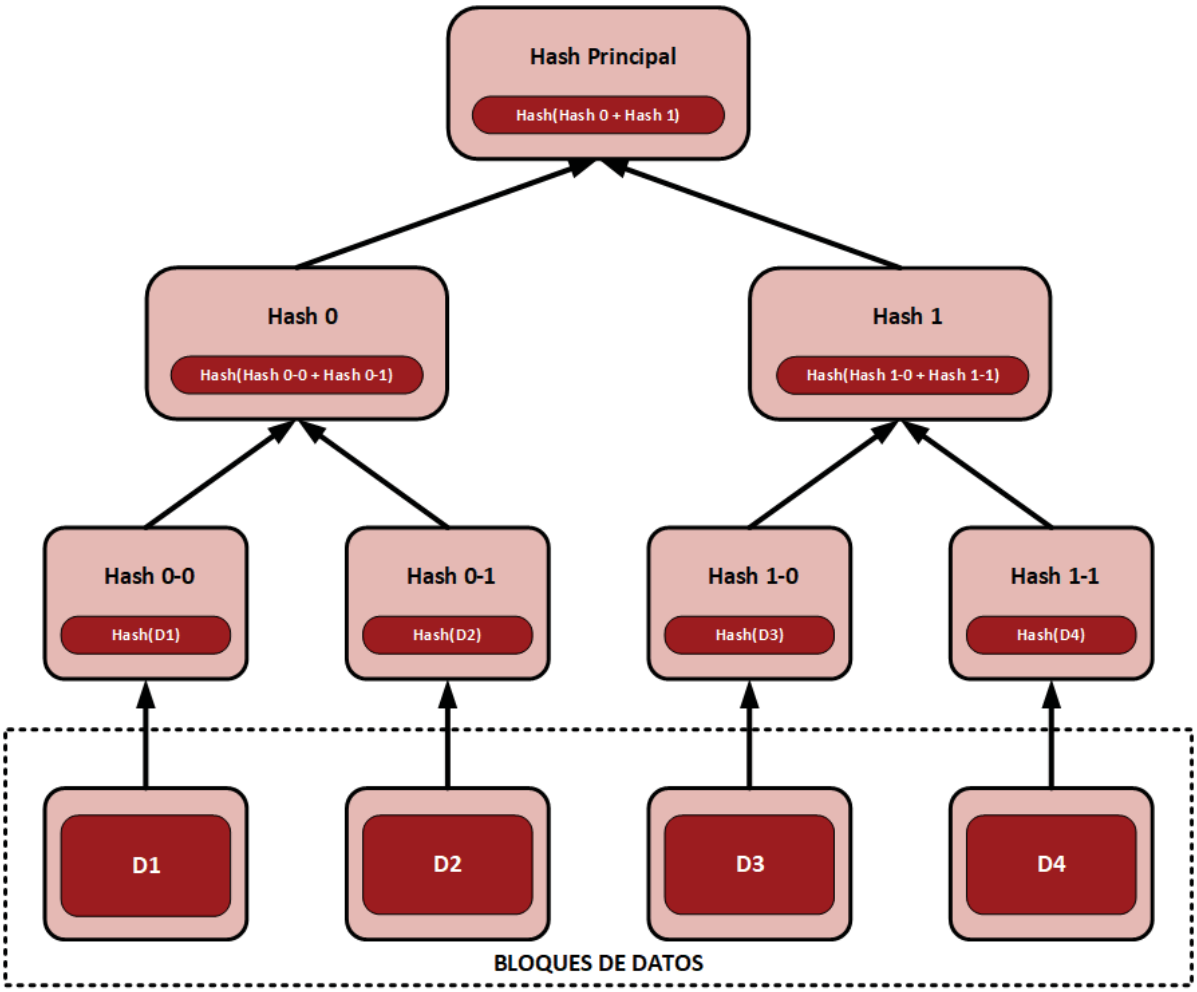

Figura 4.1: Ejemplo de Árbol de Merkle Binario

4.3 Transacciones

Las transacciones son la estructura principal y la manera de enviar valor entre usuarios en la Blockchain para pagar algún producto o servicio. Se componen en términos simples de dos listas una de entradas y otra de salidas. En cada salida se define el monto a transferir y la el hash de la clave pública del destinatario de los fondos, por lo general en las transacciones sólo hay una salida, o sea, un receptor. El funcionamiento se realiza a través de criptografía asimétrica, el cual detallaremos a continuación:

1. El emisor firma la transacción con el hash de la clave pública del destinatario, así este último es el único que puede usar los fondos de la transacción con la clave privada.
2. El emisor firma con el hash de su clave privada para validar que los fondos de la transacción los puede usar el destinatario y que realmente es él quien envía la transacción.
3. El destinatario recibe clave pública del emisor para recibir la transacción.
4. Se actualiza el saldo disponible del destinatario.
5. Para usar los fondos el destinatario utiliza su clave privada al momento de crear otra transacción.

Para ver como se constituye una transacción se utilizará un ejemplo de una de éstas en Bitcoin con el objetivo de entender de mejor manera lo explicado en el párrafo anterior. Como se puede observar en la Figura 4.2 un usuario al realizar una transacción utiliza otras que ha recibido con el fin de acumular el monto a transferir. Además, se detallan los campos que se involucran en la creación de una operación:

A. Hash que identifica de forma única una transacción.
B. Una lista de entradas de la transacción, en el que cada elemento hace referencia a una salida de alguna transacción recibida por el usuario.
C. Una lista de salidas de la transacción.
D. Hash de la transacción a la que esta entrada hace referencia.
E. El índice de la salida del listado de salidas de la transacción referenciada en D.
F. La clave pública que se debe corresponder con el address de la salida referenciada.
G. Firma digital con la clave privada del usuario emisor de la transacción, para demostrar que se puede utilizar la salida referenciada.
H. El monto a transferir a la salida.
I. Dirección de destino, éste es el hash de la clave pública del destinatario.

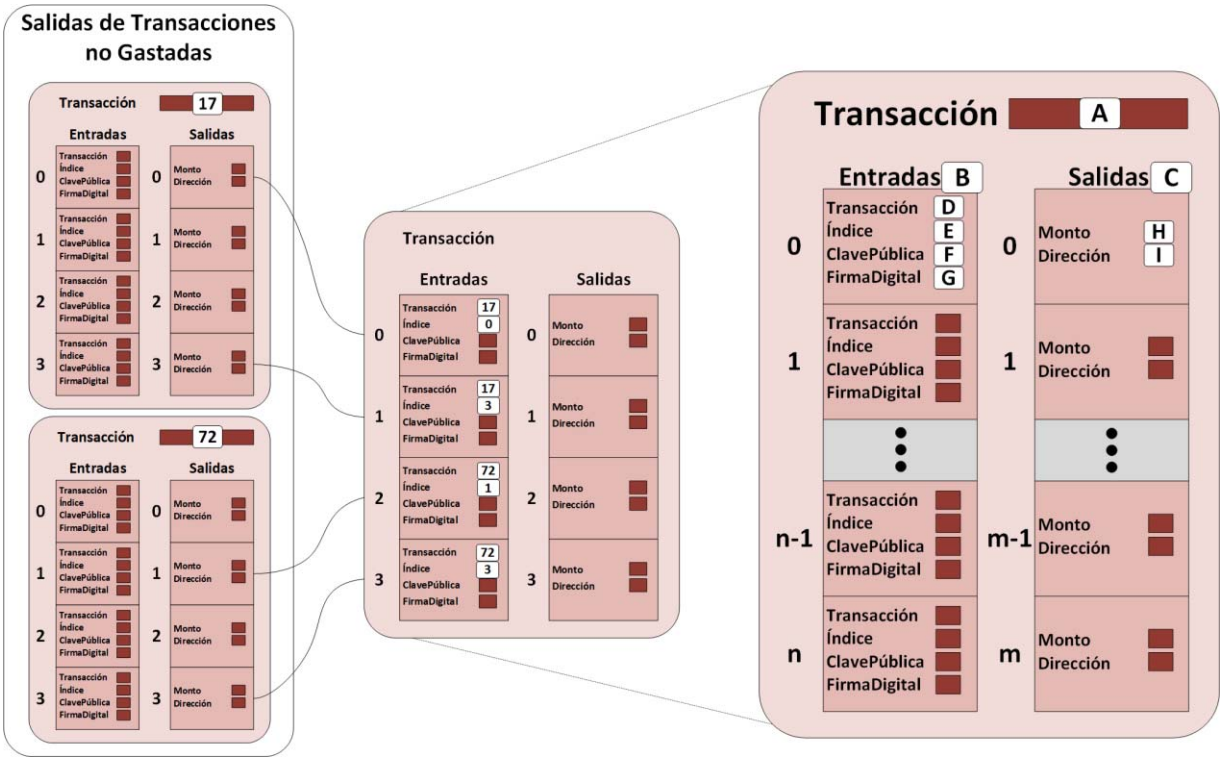

Figura 4.2: Esquema de Transacción en Bitcoin

En Ethereum las transacciones tienen dos campos más, el STARTGAS y el GASPRICE que son vitales para el funcionamiento de esta plataforma junto con el concepto de GAS. Este último representa el valor que tiene la ejecución de un paso computacional en una transacción. El primero es la cantidad máxima de pasos computacionales a ejecutar por la negociación, mientras que el segundo es el precio que se paga por cada uno de ellos.

4.4 Bloques

Los bloques son la columna vertebral de Blockchain y en dónde se guardan una cierta cantidad de transacciones ordenadas cronológicamente en una estructura llamada Merkle Tree. En este último cada una de las operaciones es una hoja del árbol y la raíz del mismo es la que se almacena en el bloque. Además del campo anterior, dentro del mismo se almacena otros datos, como una marca de tiempo, una dificultad, un campo de 32 bits llamado Nonce y el hash del bloque anterior. Este último nombrado es el hash de la combinación de todos los campos del bloque anterior vital para el funcionamiento de la cadena de bloques, dado que como se puede observar en la Figura 4.3 es un hash acumulativo y para realizar alguna modificación en alguna transacción de un bloque más antiguo se tendrá que reconstruir toda la cadena hasta llegar a la actual.

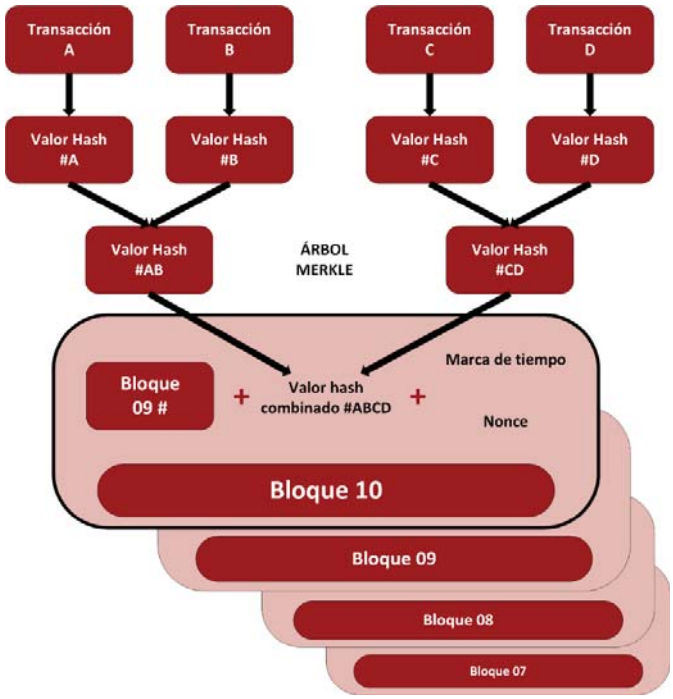

Figura 4.3: Estructura de los Bloques

4.5 Mineros

Los mineros son las entidades que se dedican a recibir transacciones, validar que sean correctas y retransmitirlas a sus pares para que toda la red se mantenga actualizada en cuanto a operaciones realizadas en el sistema. Además, estas entidades están encargadas de la creación de nuevos bloques, para realizar esta acción se debe encontrar el valor del hash del bloque actual para enlazarlo al siguiente, este debe comenzar con una cierta cantidad de bits ceros dada por la dificultad del mismo definida por el sistema, por lo que deben iterar sobre el valor del campo nonce. Lo anterior es llamado prueba de trabajo, los mineros compiten entre ellos en resolverla, el primero en hacerlo transmite su resultado a la red para que lo valide y gana una recompensa en criptomoneda, así el sistema se mantiene actualizado e incentiva a minar en la red aportando capacidad de computo en lugar de intentar modificar transacciones.

5 Documentación de la Aplicación

En esta sección el autor comienza exhibiendo la definición y especificación de requerimientos. Prosigue informando sobre la arquitectura del sistema, mostrando las distintas capas de la misma. Después se plantea el diseño del modelo de datos, presentando las estructuras de las entidades identificadas. Posteriormente se establecen los procesos y servicios que la plataforma indicando el flujo de datos principal. Para finalizar se da a conocer la documentación técnica ligada al código de la aplicación. En el Anexo A se enseñan las capturas de pantalla a las distintas funciones del software.

5.1 Definición y Especificación de Requerimientos

5.1.1 Definición General del Proyecto

La aplicación planteada en el presente documento mantendrá un registro de las propiedades mediante la Blockchain de Ethereum. Dentro de los propósitos que tiene la plataforma están reemplazar a los entes que intervienen actualmente en el proceso y generar un impacto económico y social en los individuos que hagan uso de esta. Los usuarios serán todas aquellas personas que sean propietarios de un bien raíz y tengan a su disposición una billetera con saldo en Ether.

5.1.2 Especificación de Requerimientos

5.1.2.1 Requisitos Funcionales

A nivel general el sistema permite además de la inscripción de propiedades, la administración de estas, al poder colocarlas a la venta, transferir el dominio de la misma a otra persona y gestionar los contratos de compraventa. Además de las funcionalidades nombradas existen otras que se definirán para cada tipo de usuario con el propósito de una mejor comprensión. En las siguientes secciones se presentan los casos de uso que representan todas las funcionalidades que la plataforma ofrece a propietarios, compradores, usuario general, tasadores y el creador del contrato.

Antes de poder utilizar la aplicación se debe realizar el registro de usuario de modo obligatorio para obtener la identidad real del individuo. Como lo muestra la Figura 5.1 se debe validar la información ingresada por la persona y pedir la clave del sistema para guardar la información en la Blockchain. Una vez registrado este no necesitará de un ingreso con clave ya que sus datos están guardados en el sistema y siempre disponibles para su uso, asociando su billetera con estos.

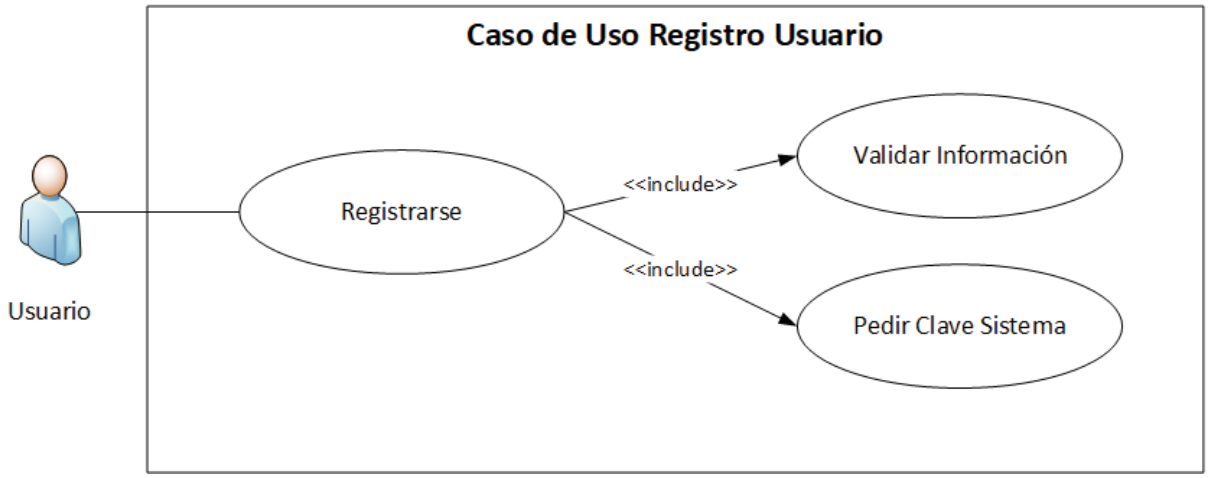

Figura 5.1: Caso de Uso Registro Usuario

El creador del contrato es la persona quién monta el código base de la aplicación (contrato inteligente) en la Blockchain, o sea, es como el administrador. Las funciones que tiene este tipo de usuario son más administrativas y no tiene, en ningún caso, poder sobre los datos almacenados en la cadena de bloques si no es el real propietario de un bien raíz. A continuación, se detallan las tareas que éste podrá realizar tal y como muestra la Figura 5.2:

1. Cambiar Clave del Sistema: el creador podrá cambiar la clave del sistema por otra generada automáticamente por la plataforma.
2. Agregar Tasador: el creador será el único con la potestad de dar permisos de tasador a una persona.
3. Ver Balance Sistema: el creador podrá ver la recaudación de la plataforma.
4. Retirar Balance Sistema: el creador podrá hacer retiros de lo recaudado en la aplicación.

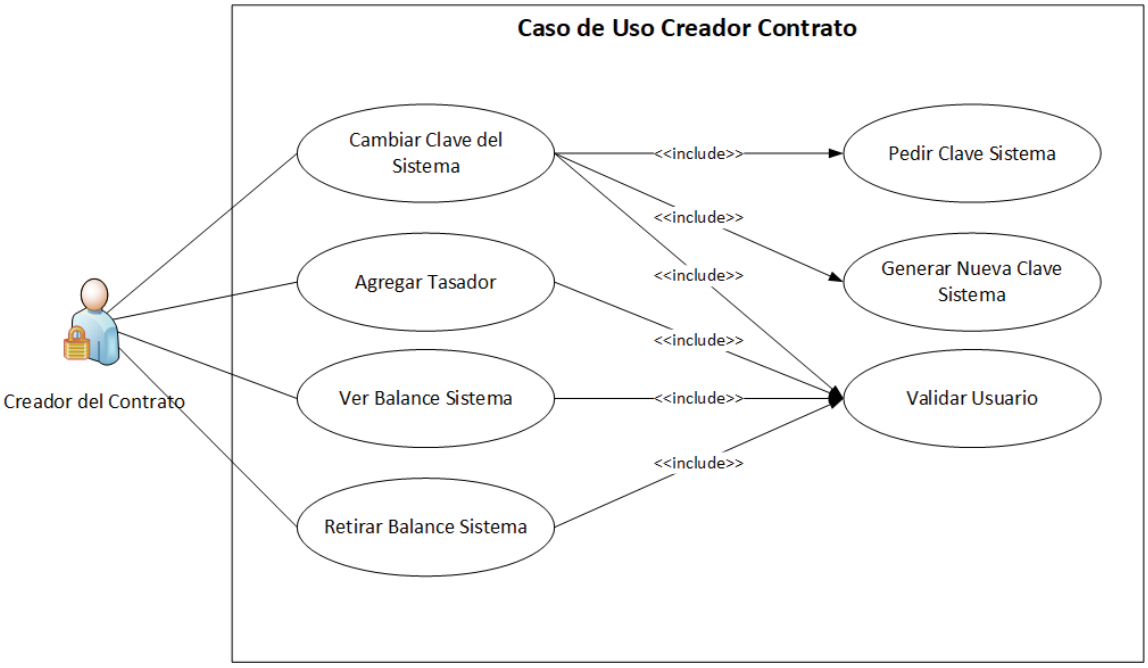

Figura 5.2: Caso de Uso Creador Contrato

El usuario general es la persona que ya se ha registrado en la aplicación independientemente si pertenece a alguna otra categoría de usuario. Las funcionalidades de este son de acceso universal, es decir, no tienen restricciones de ejecución más allá de la identificación inicial del mismo y los permisos correspondientes para ver información sensible. A continuación, se detallan las tareas presentadas en la Figura 5.3 que se podrán realizar luego del registro:

1. Agregar propiedad: una persona puede agregar una propiedad de su dominio.
2. Ver Propiedad: un individuo puede ver la información de alguna propiedad.
3. Ver Precio Trámite: el usuario podrá ver el precio del trámite.
4. Actualizar Precio Trámite: Por la variación del precio del Ether se podrá actualizar el precio del trámite.
5. Ver Detalle Venta: una persona podrá ver la información de una venta si es que tiene los permisos.

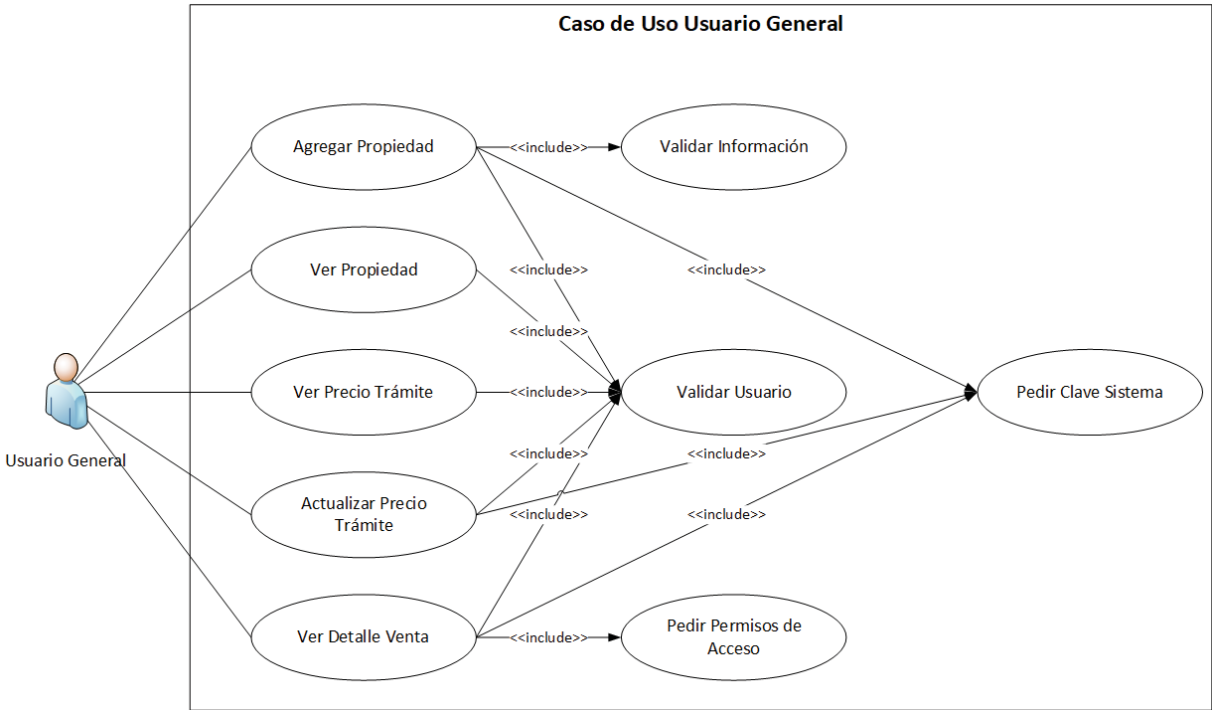

Figura 5.3: Caso de Uso Usuario General

Una persona adquiere la categoría de propietario luego de la inscripción de un bien raíz de su dominio en el sistema. Las funcionalidades están limitadas sólo a las propiedades de las que son dueños y a los contratos en los que ellos sean partícipes. Las tareas que puede realizar este tipo de usuario se muestran en la Figura 5.4 y se detallan a continuación de forma complementaria:

1. Poner Propiedad a la Venta: una persona podrá poner a la venta un bien raíz de su propiedad.
2. Cancelar Venta Propiedad: un propietario puede cancelar la venta de una propiedad sólo antes de dar permiso para que sea firmado.

3. Agregar Descripción Venta: los vendedores podrán agregar la descripción a un contrato de compraventa, siendo este el cuerpo del mismo.
4. Dar Permiso Firmar Contrato Venta: el vendedor tendrá que dar permiso a un RUT para firmar el contrato.
5. Ver Estado de Venta: el vendedor podrá ver el estado actual en el que se encuentra una venta.
6. Transferir Dominio Propiedad: el propietario de un bien raíz podrá transferir el dominio de éste a otra persona. Este trámite tiene un valor, equivalente en Ether, de cinco mil pesos chilenos.
7. Finalizar Venta Propiedad: el propietario podrá, una vez cumplido los pasos previos, finalizar la venta del bien, cambiando automáticamente de dueño.
8. Dar Permiso Para Ver Información de Venta: el propietario podrá dar permiso para ver el contrato a una persona ajena a este.

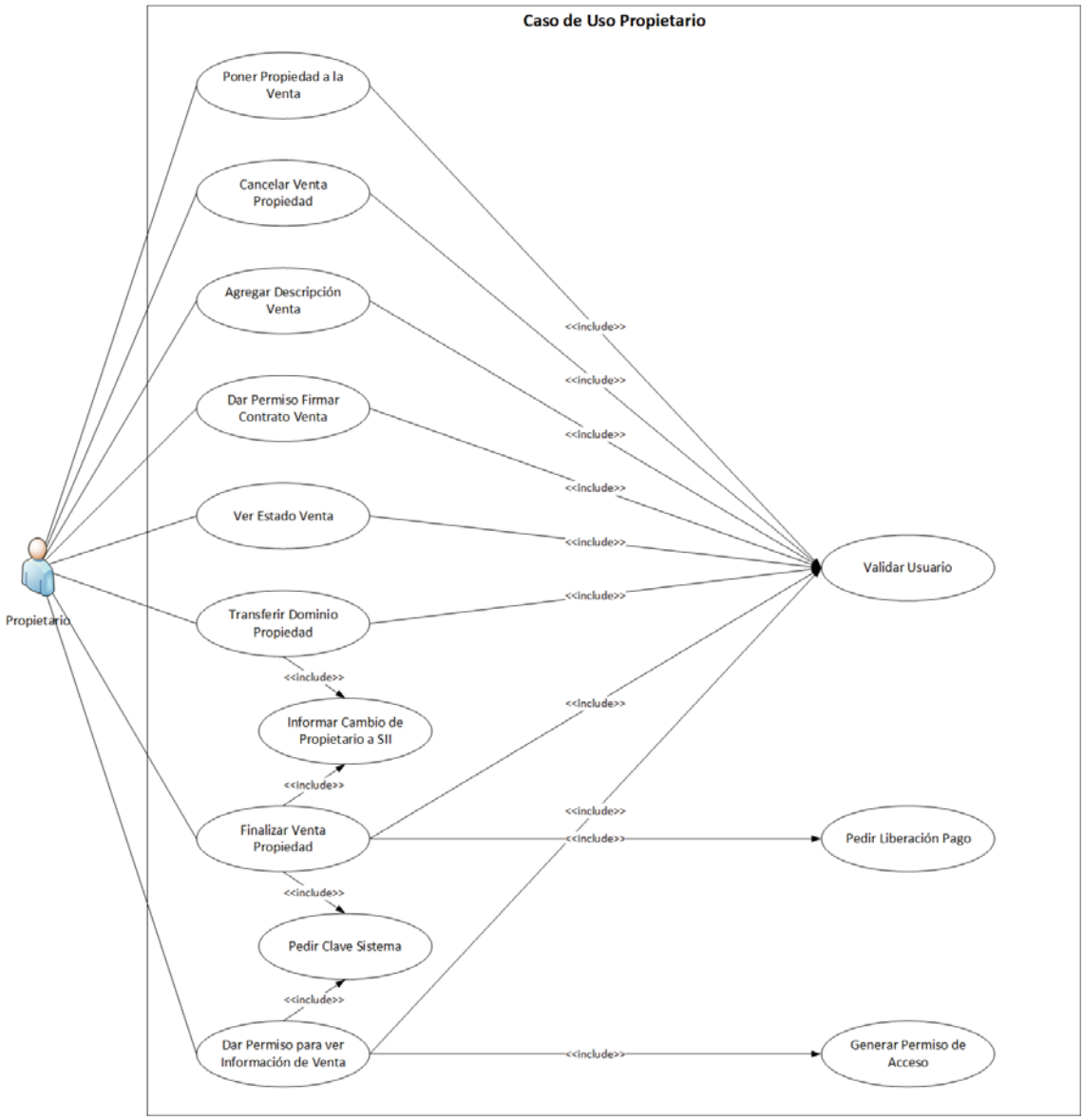

Figura 5.4: Caso de Uso Propietario

Un individuo obtiene la categoría de comprador cuando este se hace partícipe de un contrato de compraventa al momento que el propietario de un bien raíz lo exprese. Esto ocurre cuando la propiedad está a la venta, el interesado se contacta con el dueño y este último lo acepta como el comprador dándole permiso para firmar el documento. A continuación, se detallan las funcionalidades presentadas en la Figura 5.5:

1. Firmar Contrato de Venta: el comprador podrá firmar el contrato una vez lo haya leído.
2. Ver Estado de Venta: el comprador podrá ver el estado actual en el que se encuentra una venta.
3. Dar Permiso para Ver Información de Venta: el comprador podrá dar permiso para ver el contrato a una persona ajena a este.
4. Agregar Documento Custodia Bancaria: el comprador podrá agregar la información del documento de custodia bancaria realizado en la institución correspondiente.
5. Solicitar Devolución Custodia Bancaria: el comprador podrá solicitar la devolución de su dinero si la venta no se concreta.

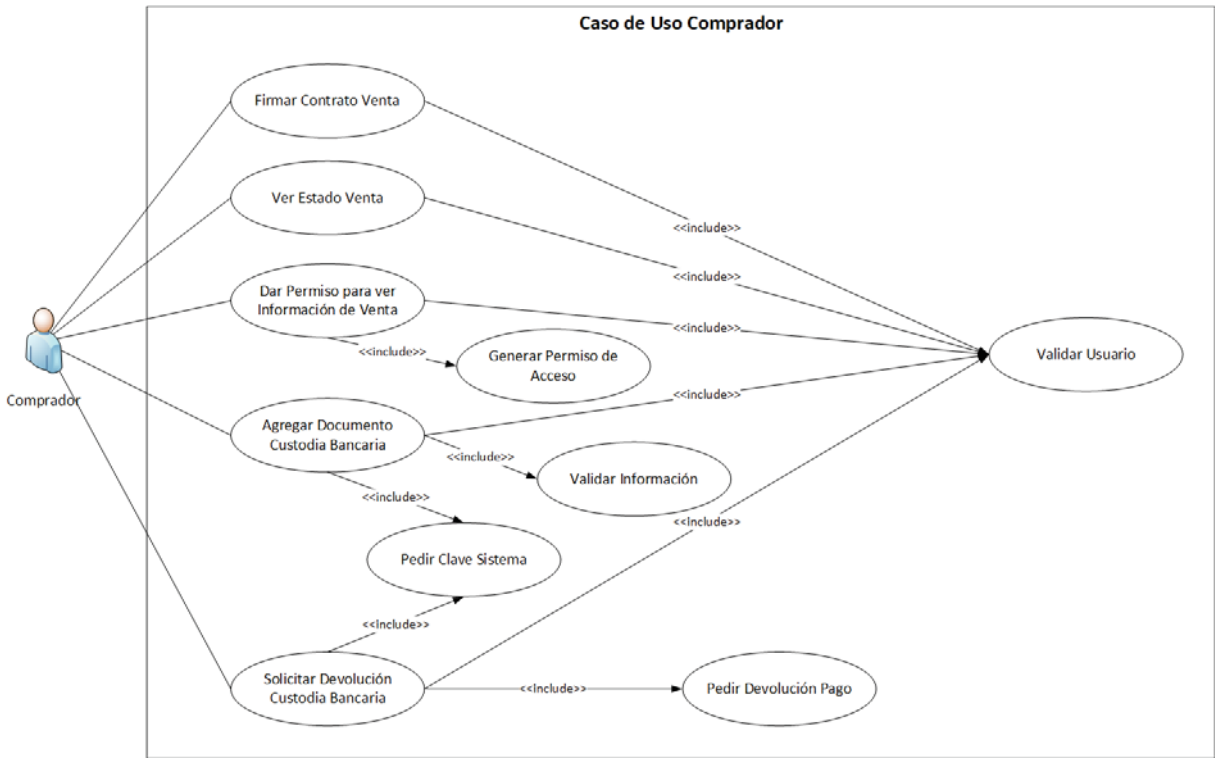

Figura 5.5: Caso de Uso Comprador

Una persona adquiere privilegios de tasador cuando el creador del contrato se los otorgue. Este tendrá una única función en el sistema, pero muy importante para el funcionamiento de la plataforma. Como se muestra en la Figura 5.6 este tipo de usuario tendrá que realizar el cambio de valor de un bien raíz producto de la revalorización que la propiedad tiene a lo largo del tiempo.

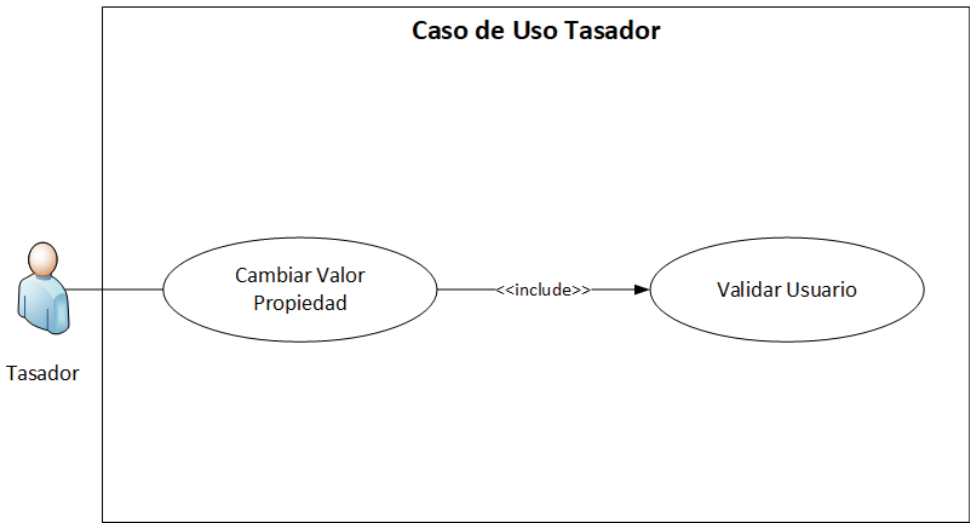

Figura 5.6 Caso de Uso Tasador

5.1.2.2 Alcance

El alcance del sistema se compone de todas las tareas que son necesarias para completar con la realización de la misma. Para ello se considera sólo las fases del proceso de desarrollo de software dado que este apartado es para una parte de la tesis que es la aplicación. En la Figura 5.7 se muestra la estructura de desglose de trabajo o EDT de la plataforma.

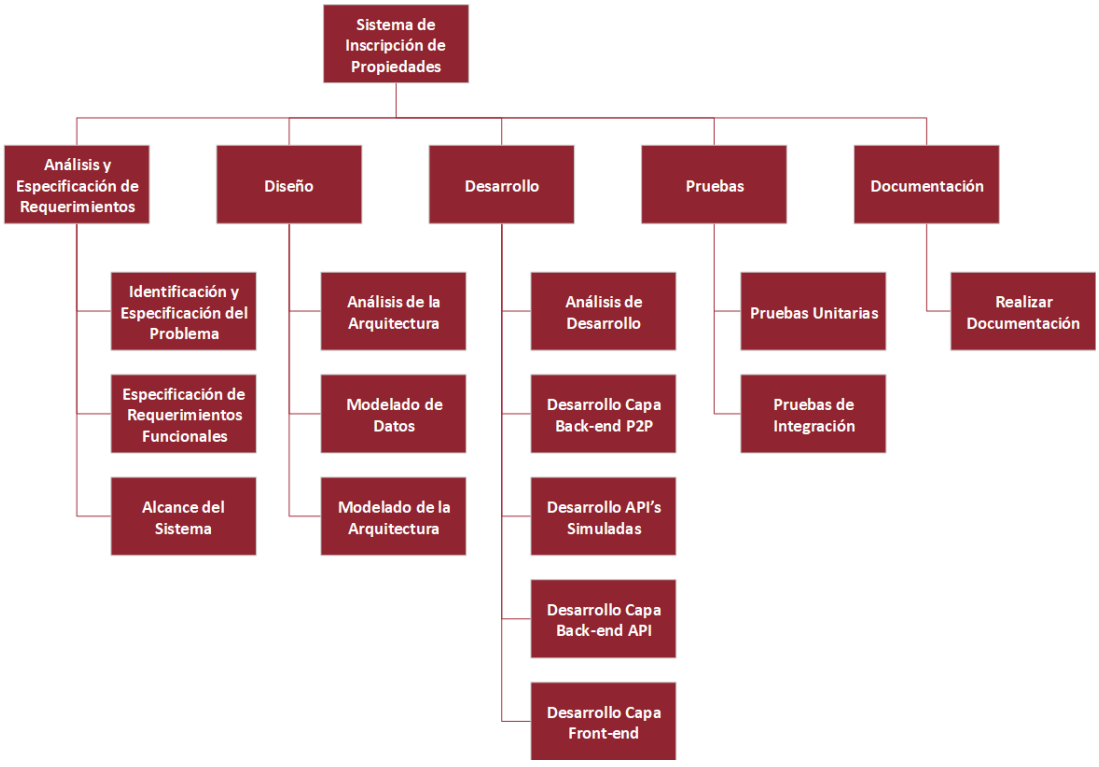

Figura 5.7: EDT de la Plataforma

5.3.1 Especificación de Procedimientos

Durante las distintas fases del proceso de desarrollo de software se utilizaron distintos instrumentos para apoyar a las tareas definidas. Dado que la Blockchain de Ethereum tiene un número acotado de herramientas para crear aplicaciones basadas en esta, no existen muchas opciones de desarrollo para las capas que la involucran. La Tabla 5.1 muestra los softwares utilizados, señalando con una equis (X) si se ha usado en la etapa correspondiente y en caso contrario con un guion (-).

Tabla 5.1: Herramientas Utilizadas en el Proceso de Desarrollo

Herramientas	Fase del proceso de desarrollo de Software					Detalle
	Análisis y Especificación de Requerimientos	Diseño	Desarrollo	Pruebas	Documentación	
Microsoft Visio	X	X	-	-	-	Software de dibujo vectorial de Microsoft
Microsoft Word	-	-	-	X	X	Software de procesamiento de textos de Microsoft
Remix Solidity	-	-	X	X	-	Entorno de desarrollo integrado online de Solidity de Ethereum
Sublime Text	-	-	X	X	-	Editor de texto y código fuente
Truffle	-	-	X	X	-	Framework de aplicaciones web descentralizadas en Ethereum escrita en JavaScript
Ruby On Rails	-	-	X	X	-	Framework de aplicaciones web escrito en Ruby
Ganache	-	-	X	X	-	Software que provee de una Blockchain local
MetaMask	-	-	X	X	-	Nodo de Ethereum para navegadores web

En cuanto a la metodología de desarrollo utilizada durante el proyecto el autor opta por usar el modelo iterativo como se muestra en la Figura 5.8. Al no tener los requerimientos claros es el ideal para su uso en la realización de la plataforma, dado que se van refinando los mismos en las iteraciones. Así se logra que la aplicación ofrezca las funcionalidades que necesitan los usuarios y el problema a resolver.

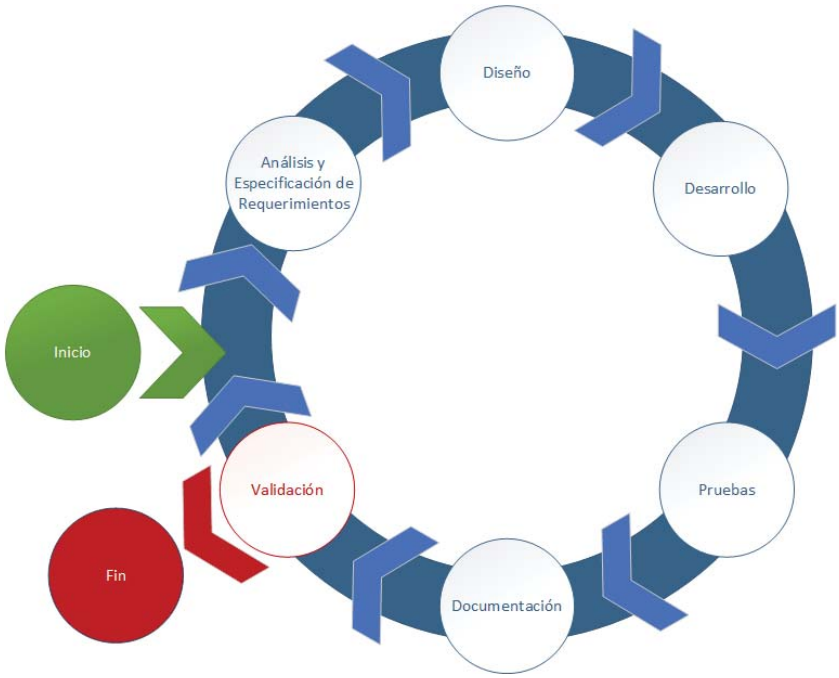

Figura 5.8: Metodología de Desarrollo de la Plataforma

5.2 Arquitectura del Sistema

La aplicación propuesta en este documento se compone de tres capas, una de Front-End y dos de Back-End. Todos los módulos son independientes uno del otro en términos físicos, pero se necesitan comunicar entre ellos para que el sistema otorgue a los usuarios las funcionalidades. La Figura 5.9 muestra la arquitectura de la plataforma con las capas nombradas y las entidades externas necesarias para el funcionamiento de misma. En las siguientes secciones se describen más en detalle cada una de las partes.

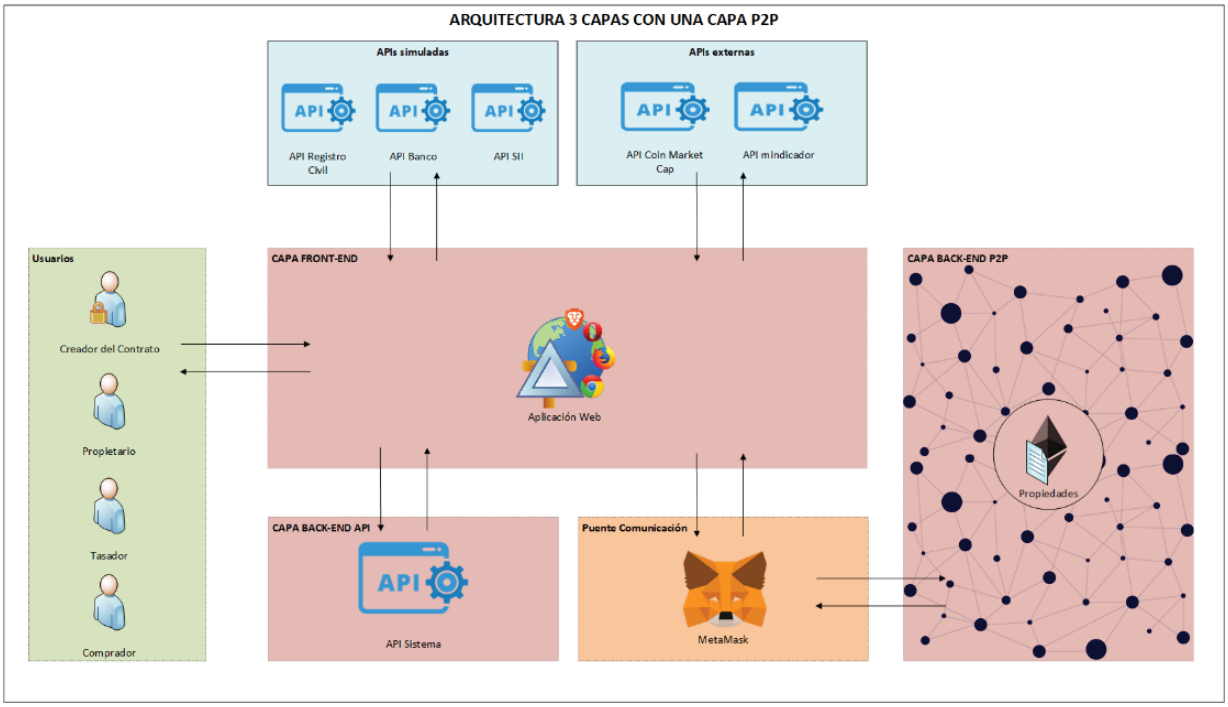

Figura 5.9: Arquitectura del Sistema

5.2.1 Capa Front-end

La capa Front-end es la parte del sistema que va a interactuar con los usuarios mediante una interfaz familiar y entendible para los últimos. Además, por un lado, es en donde se realizará la recolección de los datos de entrada a la aplicación ajustándolos para que sean procesados más adelante. Por otro, será el receptor de las respuestas de los distintos componentes para presentarlos a los usuarios de una manera comprensible para ellos.

Esta capa está implementada con Truffle [10], un framework de JavaScript puro, por lo que es completamente parte del cliente y no del servidor. En este sentido no se podrá generar ninguna comunicación con alguna base de datos para realizar operaciones de ingreso, modificación y eliminación de datos en este módulo. Así toda información que requiera esta capa se tendrá que hacer comunicándose con las otras partes del sistema.

5.2.2 Capa Back-end API

La capa Back-end API es la encargada de administrar las claves y los permisos de acceso a la información de la Blockchain. Estas estarán almacenadas de manera cifrada con el fin de mantener estos datos seguros en caso de que un atacante esté escuchando la comunicación entre esta y la capa anterior. La presente debe restringir que sólo el Front-end pueda establecer una comunicación mediante llamadas POST.

Como se menciona en la sección 5.2.1 el Front-end no puede guardar datos en alguna base de datos. Para suplir esta función está la presente dado que se requiere una clave almacenada tanto en la Blockchain como en la aplicación web. Con esto la plataforma puede validar que las funciones de la capa Back-end P2P sean llamadas exclusivamente desde el Front-end del sistema.

Esta capa está programada en el lenguaje de programación Ruby utilizando el framework Rails. Se utiliza esta tecnología por la cantidad de herramientas que otorga y ser de una rápida codificación. Con esto permite realizar de manera sencilla una API y generar un firewall interno.

5.2.3 Capa Back-end P2P

Esta capa es el núcleo de la plataforma de inscripción de propiedades que ofrece mediante los Smart Contracts distintos tipos de datos y funciones. Toda la información de las personas, propiedades y contratos de compraventa generadas en el Front-end llegarán a través de MetaMask [11] en forma de transacciones. Por lo que el Back-end P2P permitirá procesar y almacenar de manera descentralizada y segura la información ingresada por el usuario.

La presente capa está codificada en Solidity, el lenguaje de programación de Ethereum para la creación de contratos inteligentes. Las mencionadas se ocupan al ser totalmente distribuida e ideal para este tipo de plataforma. Además, hereda todas las propiedades de la tecnología Blockchain como lo es la inmutabilidad y la descentralización.

5.2.4 Puente de Comunicación

MetaMask [11] es una extensión para distintos navegadores web, entre los que se encuentran Chrome, Mozilla Firefox, Opera y Brave, que proporciona un nodo Ethereum funcionando en el mismo. Es la encargada de realizar la comunicación entre el Front-end y el Back-end P2P mediante transacciones. Con esto la aplicación podrá acceder a los contratos inteligentes almacenados en la Blockchain de Ethereum.

5.2.5 APIs Simuladas

Dentro de la aplicación se deben validar distintos datos que se necesitan almacenar en la Blockchain de Ethereum. Entre lo que se necesita legitimar está la información de los usuarios, de las propiedades y de las custodias bancarias. Para ello se crean tres APIs que realizarán tales acciones Registro Civil, Servicio de Impuestos Internos y el sistema bancario.

Estas APIs están programadas en lenguaje de programación Ruby utilizando el framework Rails. Estas se utilizan al ser una tecnología de rápida codificación y por la

cantidad de herramientas que otorga. Así en un breve tiempo se pueden generar este tipo de servicios con un bajo nivel de dificultad.

5.2.6 APIs Externas

El sistema requiere de los valores tanto del Ether como del Dólar actual para poder usarlos internamente. Para ello se utilizarán las APIs que proporcionan CoinMarketCap y mindicador.cl. El primero, mantiene un registro constante del precio en dólares no tan sólo del ETH sino de una gran cantidad de criptomonedas. La segunda, realiza un mapeo constante del Banco Central de Chile para entregar información creíble y actualizada de los indicadores económicos del país. En definitiva, el autor escoge estas dos fuentes de información al ser confiables en la entrega de la misma y por no tener trabas en el acceso a los datos.

5.3 Diseño del Modelo de Datos

La aplicación no guardará ningún dato de las propiedades, las ventas, los usuarios y las custodias en bases de datos tradicionales, sino que lo hará en contratos inteligentes. Para el modelado se utilizará UML ya que es ideal para representar esta parte del sistema. La Figura 5.10 muestra las distintas estructuras que representan a las entidades que se han identificado para solucionar el problema.

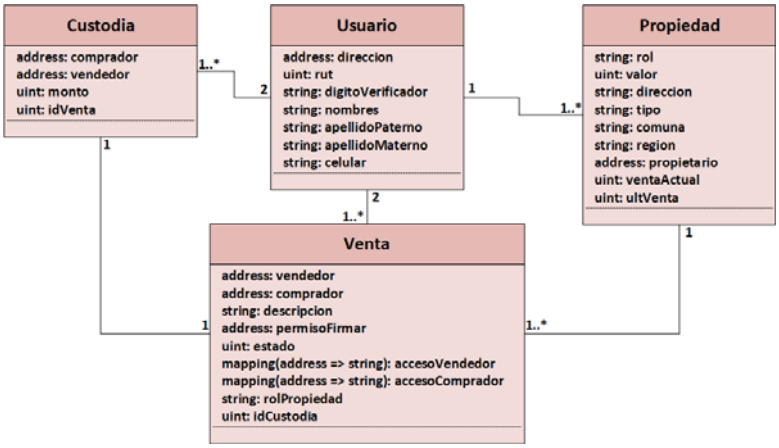

Figura 5.10: Modelo de Datos Back-end P2P

Por otro lado, la API de la aplicación se encargará de guardar ciertas claves y permisos necesarios para el funcionamiento de la plataforma. Estos se almacenarán en una base de datos dentro de la capa de Back-end API que sólo podrá ser accedida por la aplicación. La Figura 5.11 muestra las tablas utilizadas para este propósito.

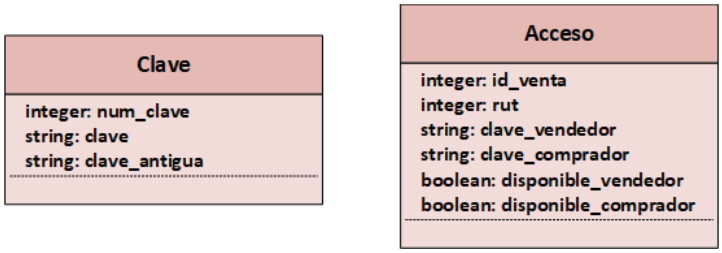

Figura 5.11: Modelo de Datos Back-end API

5.4 Procesos y Servicios Ofrecidos

La aplicación ofrece variadas funcionalidades a los distintos usuarios que tendrá la misma. En las siguientes secciones se describe el conjunto de acciones más importante de la plataforma, las que son necesarias para comprar/vender una propiedad. La totalidad de procesos ofrecidos por el sistema se encuentra en el Anexo B: Flujo de Datos.

5.4.1 Paso 1: Ingresar la Propiedad al Sistema

Antes de realizar cualquier acción la propiedad debe estar registrada en el sistema, cuya acción debe ser ejecutada por el propietario real de la misma. Cuando se ingresa el bien en la plataforma se valida con el Servicio de Impuestos Internos el dominio real del mismo. En este sentido ninguna persona podrá inscribir una propiedad sin ser el dueño.

Por un lado, la Figura 5.12 muestra el flujo que tendrá la información a través del tiempo en la realización del ingreso de una propiedad. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para obtener un mayor detalle de la ejecución de la presente ir a la sección Anexo B.4: Flujo de Datos Usuario General en la Tabla B.7.

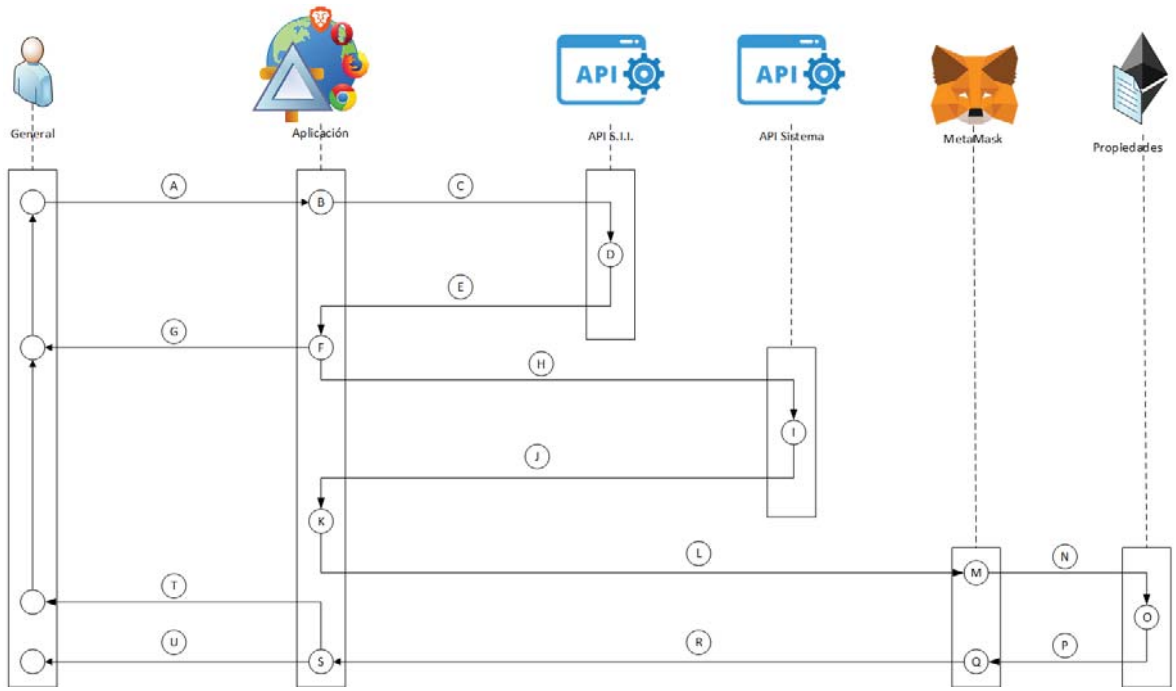

Figura 5.12: Diagrama de Flujo Agregar Propiedad

5.4.2 Paso 2: Poner la Propiedad a la Venta

Una vez que la propiedad sea ingresada en el sistema, el propietario de esta podrá colocarla a la venta si este lo desea. Al realizar esta acción en la plataforma se creará un contrato de compraventa que se debe llenar más adelante al tener la información del futuro comprador. Además, al ejecutar este acto se mostrará el número de celular del propietario en la información de la propiedad, cuya información se ocultará al ingresar el contrato, esto porque ya hay un comprador seguro.

Por un lado, la Figura 5.13 muestra el flujo que tendrá la información a través del tiempo al momento de poner una propiedad a la venta. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para más información dirigirse al desglose en la sección Anexo B.5: Flujo de Datos Propietario en la Tabla B.12.

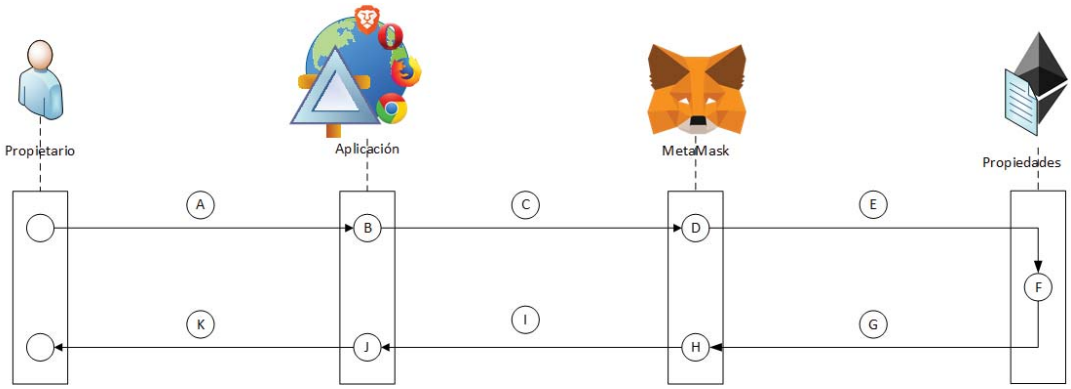

Figura 5.13: Diagrama de Flujo Poner Propiedad a la Venta

5.4.3 Paso 3: Interesado se Contacta con el Propietario

El interesado en ser el futuro dueño de la propiedad deberá ver la información de la misma en el sistema, en donde, si la propiedad está en venta y no hay ninguna persona ya con contrato, podrá ver el número de celular del propietario del bien. Luego, el interesado se tendrá que contactar externamente con la persona que tiene el dominio de la propiedad. Para finalizar, este último podrá realizar el contrato de compraventa con la información del comprador.

Por un lado, la Figura 5.14 muestra el flujo que tendrá la información a través del tiempo para ver la información de una propiedad. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Visitar la sección Anexo B.4: Flujo de Datos Usuario General en la Tabla B.8 para más detalles al respecto.

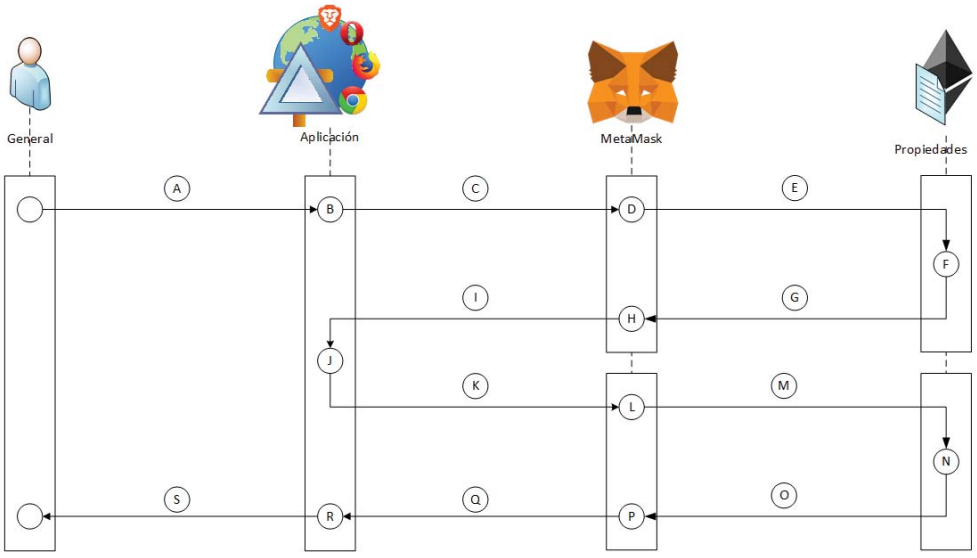

Figura 5.14: Diagrama de Flujo Ver Propiedad

5.4.4 Paso 4: Agregar Contrato de Compraventa

Una vez el propietario tenga la información suficiente para realizar el contrato de compraventa este tendrá que ingresar el convenio en el sistema. Al realizar esta acción ningún interesado más se podrá contactar con su persona, esto al ocultarse el número de celular, información personal que sólo debe mostrarse cuando sea realmente necesario. Cuando el vendedor ingresa este documento al sistema automáticamente quedará firmado con el hash de su clave pública, atributo que es inherente al propietario, quien es el único que puede ejecutar esta función.

Por un lado, la Figura 5.15 muestra el flujo que tendrá la información a través del tiempo en la realización del ingreso del contrato de compraventa. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para obtener un mayor detalle de la ejecución de la presente ir a la sección Anexo B.5: Flujo de Datos Propietario en la Tabla B.14.

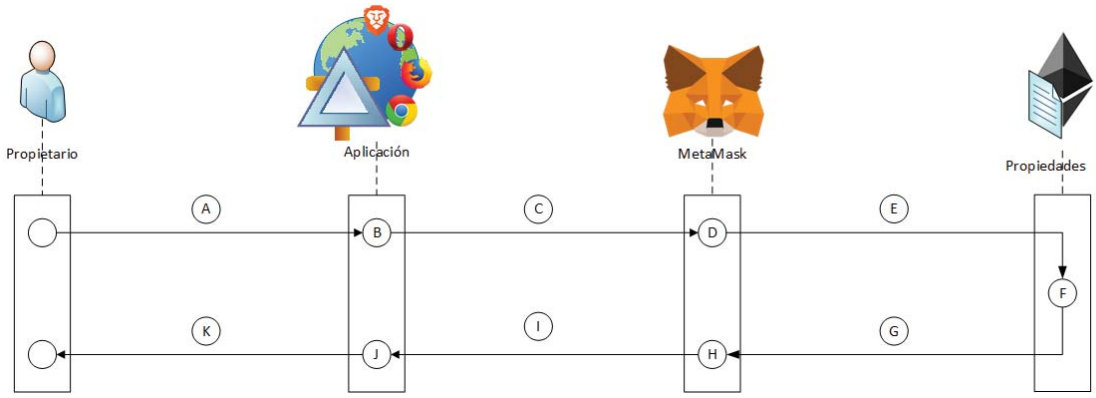

Figura 5.15: Diagrama de Flujo Agregar Contrato de Compraventa

5.4.5 Paso 5: Dar Permiso para Firmar al Comprador

Ya con el contrato de compraventa ingresado en la plataforma el propietario podrá otorgar el permiso para firmar al comprador. Aquí el vendedor debe indicar el RUT del usuario a dar el permiso para que sólo él pueda realizar esta acción. Este paso debe existir dado que cualquier persona podría firmar el convenio sin ser realmente el interesado en comprar la propiedad.

Por un lado, la Figura 5.16 muestra el flujo que tendrá la información a través del tiempo para dar el permiso para firmar al comprador. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para mayor información de la ejecución de este proceso ir a la sección Anexo B.5: Flujo de Datos Propietario en la Tabla B.15.

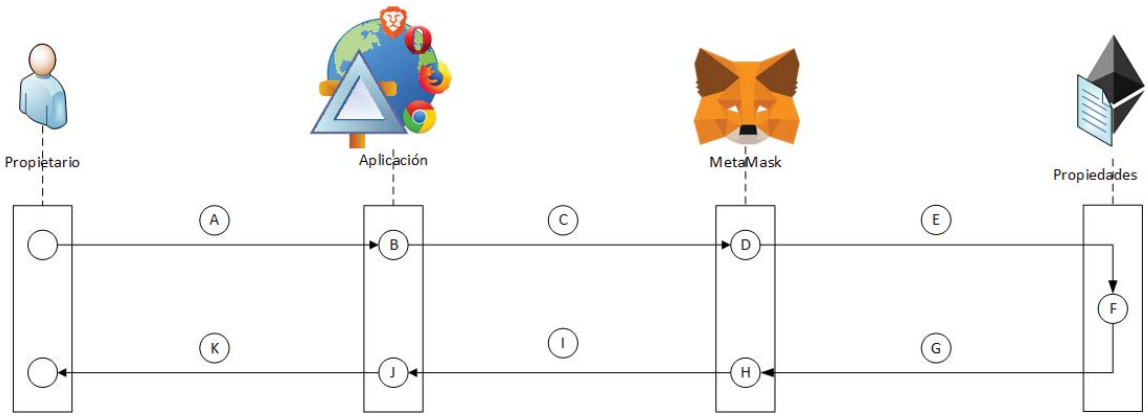

Figura 5.16: Diagrama de Flujo Dar Permiso Firmar Contrato

5.4.6 Paso 6: Firmar el Contrato de Compraventa

Una vez que el interesado consigue el permiso para firmar por parte del vendedor, el primero podrá colocar su firma en el contrato de compraventa. El contrato se firmará internamente en la capa Back-end P2P con el hash de la clave pública del comprador, quien es el único que puede llamar a esta función. Previo a esta acción la persona debió haber leído el contrato dentro de la plataforma.

Por un lado, la Figura 5.17 muestra el flujo que tendrá la información a través del tiempo para firmar el contrato de compraventa. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para obtener un mayor detalle de la ejecución ir a la sección Anexo B.6: Flujo de Datos Comprador en la Tabla B.18.

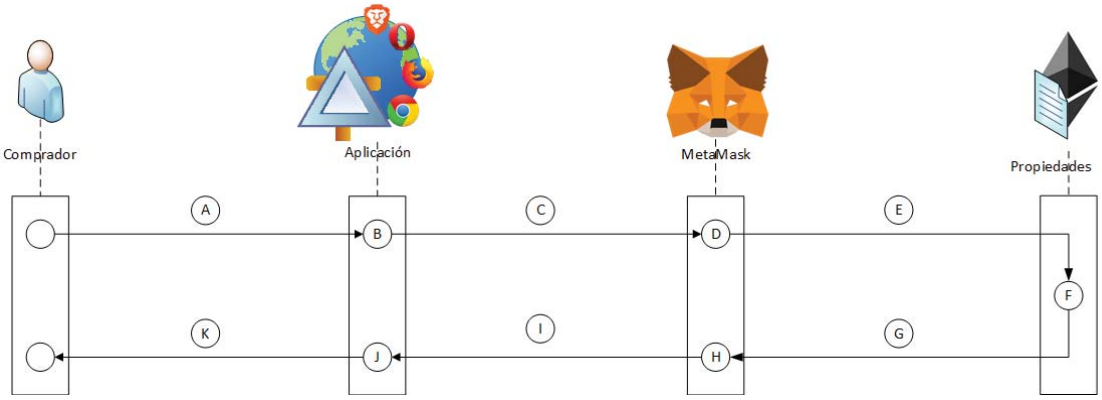

Figura 5.17: Diagrama de Flujo Firmar Contrato de Compraventa

5.4.7 Paso 7: Realizar Custodia Bancaria

Ya con el contrato de compraventa firmado el comprador deberá ir al banco y dejar el dinero de la operación en custodia de este último. El monto de la custodia bancaria debe ascender al valor que tiene la propiedad en la plataforma. Una vez realizado esto el documento quedará ingresado en los sistemas del banco en cuestión.

5.4.8 Paso 8: Ingresar Custodia Bancaria

Una vez realizada la custodia en el banco el comprador deberá ingresar el documento en la plataforma. El comprador deberá indicar la información del documento y el sistema revisará con el banco la validez del mismo. Una vez autenticada la custodia bancaria se ingresará al sistema, más específicamente en la capa Back-end P2P.

Por un lado, la Figura 5.18 muestra el flujo que tendrá la información a través del tiempo en la realización del ingreso de la custodia bancaria. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para más información dirigirse al desglose en la sección Anexo B.6: Flujo de Datos Comprador en la Tabla B.19.

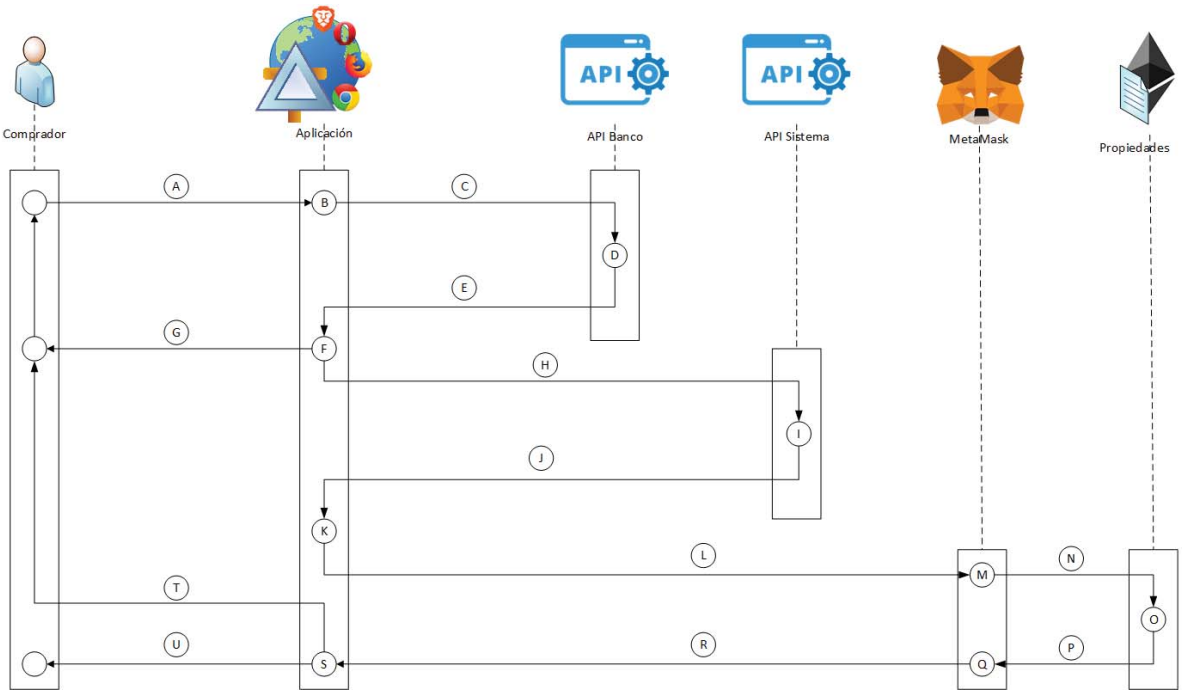

Figura 5.18: Diagrama de Flujo Ingresar Custodia Bancaria

5.4.9 Paso 9: Finalizar Venta de Propiedad

Ya completados todos los pasos anteriores el vendedor podrá finalizar la venta de la propiedad. El propietario deberá indicar, entre otros campos, el rol de la propiedad, el número de documento de venta y el número de custodia bancaria para poder realizar esta función. Luego el sistema internamente realizará varias acciones, dentro de las que se destacan la liberación del pago por parte del comprador hacia el vendedor.

Por un lado, la Figura 5.19 muestra el flujo que tendrá la información a través del tiempo para finalizar la venta de una propiedad. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para obtener un mayor detalle de la ejecución de la presente ir a la sección Anexo B.5: Flujo de Datos Propietario en la Tabla B.16.

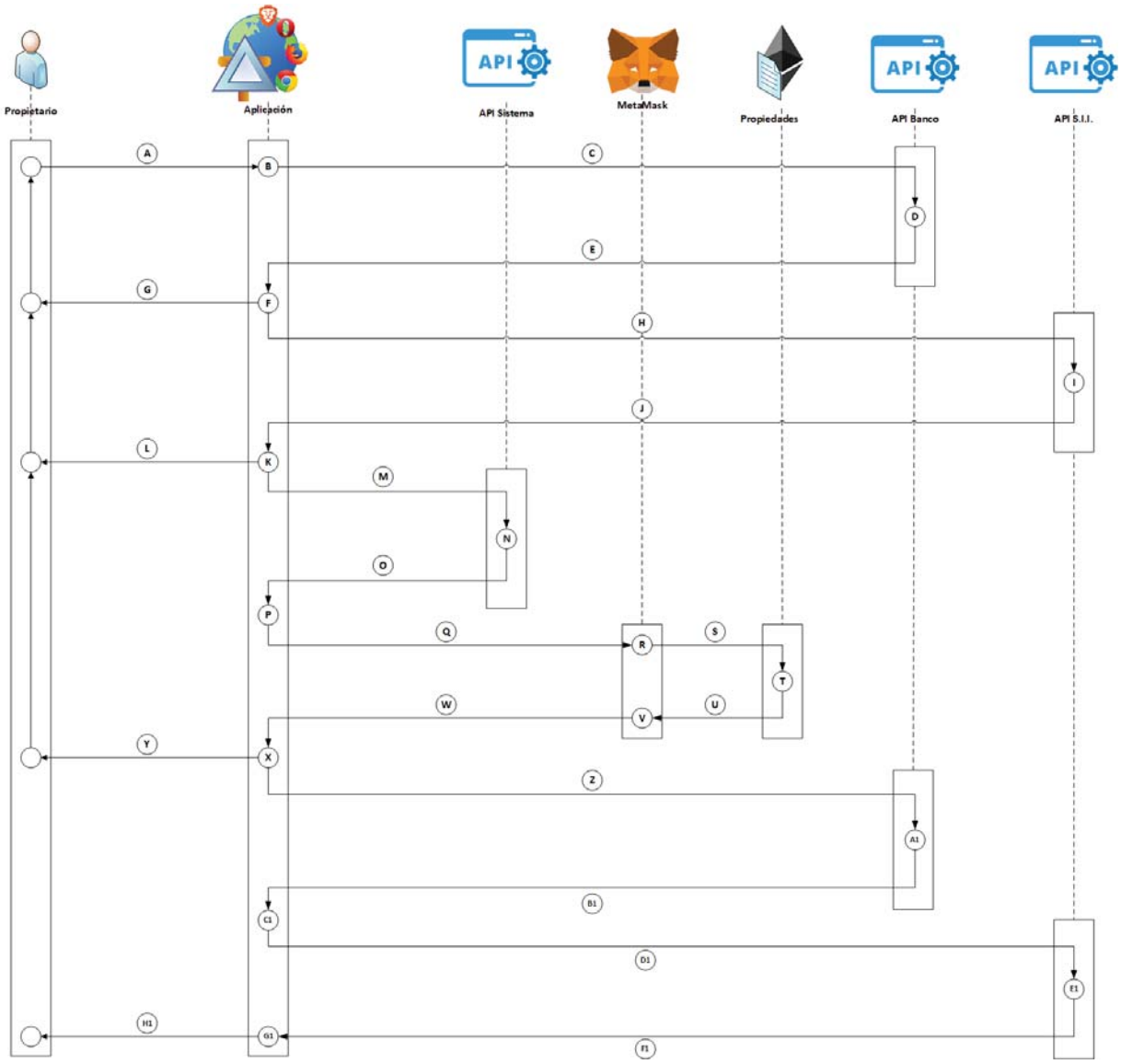

Figura 5.19: Diagrama de Flujo Finalizar Venta de Propiedad

5.4.10 Paso 10: Solicitar Devolución de Custodia Bancaria

En caso de que se hayan completado todos los pasos menos el finalizar la venta de la propiedad el comprador podrá solicitar, luego de un tiempo, la devolución de la custodia bancaria. El usuario deberá indicar, entre otros campos, el rol de la propiedad, el número de documento de venta y el número de custodia bancaria para poder realizar esta función. Con esto el sistema podrá comunicarse con el banco y ejecutar las acciones correspondientes.

Por un lado, la Figura 5.20 muestra el flujo que tendrá la información a través del tiempo en la realización del ingreso de una propiedad. Por otro, presenta las distintas entidades que involucra esta funcionalidad del sistema. Para verificar con mayor detalle el proceso presente ir a la sección Anexo B.6: Flujo de Datos Comprador en la Tabla B.20.

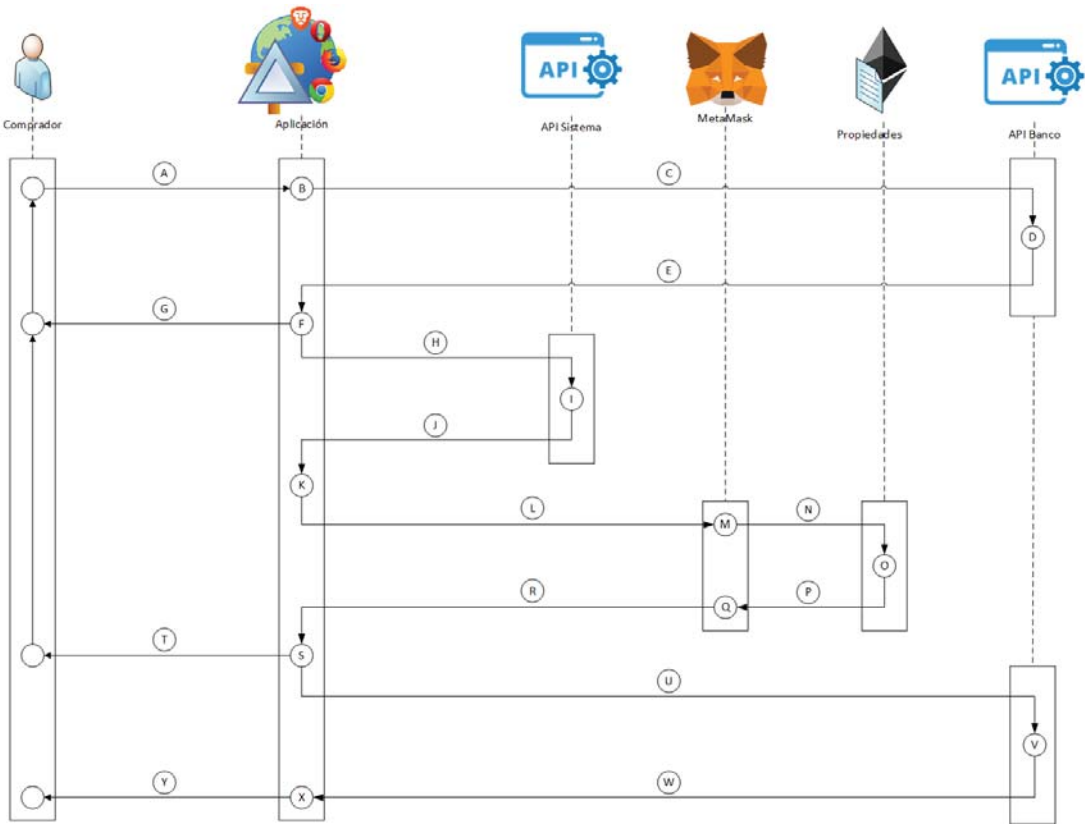

Figura 5.20: Diagrama de Flujo Solicitar Devolución de Custodia Bancaria

5.5 Documentación Técnica

Dentro de la capa Back-end P2P se han definido distintas funciones para satisfacer los requerimientos derivados del problema. En esta sección se muestran estas funcionalidades divididas por tipo de usuario y rol que cumplen en la aplicación. En la Figura 5.21 se presenta el código encargado de registrar un usuario en el sistema, la totalidad del código del contrato inteligente se expone en el Anexo C: Código de las Funciones del Sistema.

```
/*
Descripción:
    - Agrega un usuario al sistema con todos los valores correspondientes.
Parámetros de Entrada:
    - string     _clave
    - string     _nombres
    - string     _apell_pat
    - string     _apell_mat
    - uint       _rut
    - string     _digito
    - string     _celular
Parámetros de Salida:
*/
function agregarUsuario(string _clave, string _nombres, string _apell_pat, string _apell_mat, uint _rut, string _digito, string _celular) public
soloClave(_clave)
soloUsuarioNoRegistrado
soloRutNoRegistrado(_rut) {
    Usuario memory user;
    user.direccion = msg.sender;
    user.nombres = _nombres;
    user.apellido_paterno = _apell_pat;
    user.apellido_materno = _apell_mat;
    user.rut = _rut;
    user.digitoVerificador = _digito;
    user.celular = _celular;

    cantidadUsuarios += 1;

    usuarios[_rut] = user;
    addRut[msg.sender] = _rut;
    userRegistrado[msg.sender] = true;
    rutRegistrado[_rut] = true;
}
```

Figura 5.21: Función Agregar Usuario

Por tipo de usuario se presentan las funcionalidades para creador (Tabla 5.2), tasador (Tabla 5.3), usuario general (Tabla 5.4 y 5.5), propietario (Tabla 5.6), comprador (Tabla 5.7) y las que comparten vendedor y comprador (Tabla 5.8). Se detalla para cada una de ellas los parámetros de entrada y salida, como también una descripción de la acción que realiza internamente. Además, se muestran las distintas restricciones de ejecución, indicando datos de entrada y el detalle de lo que efectúa.

Tabla 5.2: Documentación Funciones Creador

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
agregarTasador	uint	_rut	-	-	Ingresa el RUT del tasador en la variable "_rut" y le otorga los permisos correspondientes.
	-	-	-	-	
	-	-	-	-	
cambiarClaveSistema	string	_clave	-	-	Cambia la clave antigua asignadole el valor de "_clave_nueva".
	string	_clave_nueva	-	-	
	-	-	-	-	
verBalance	-	-	uint	balance	Retorna el valor del balance del sistema.
	-	-	-	-	
retirarBalance	-	-	-	-	Transfiere todo el balance del sistema al creador del contrato.
	-	-	-	-	
	-	-	-	-	

Tabla 5.3: Documentación Funciones Tasador

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
cambiarValorPropiedad	string	_rolProp	-	-	Cambia el valor de la propiedad con rol "_rolProp" a "_valorPesos".
	uint	_valorPesos	-	-	
	-	-	-	-	

Tabla 5.4: Documentación Funciones Usuario General Parte 1

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
agregarPropiedad	string	clave	-	-	Agrega una propiedad con todos los valores correspondientes.
	string	rolProp	-	-	
	string	tipo	-	-	
	uint	valorPesos	-	-	
	string	direccion	-	-	
	string	comuna	-	-	
	string	region	-	-	
agregarUsuario	string	clave	-	-	Agrega un usuario al sistema con todos los valores correspondientes.
	string	nombres	-	-	
	string	apell_pat	-	-	
	string	apell_mat	-	-	
	uint	rut	-	-	
	string	digito	-	-	
	string	celular	-	-	
verPropiedad	string	rolProp	uint	valor	Función que retorna la información de la propiedad con rol "_rolProp".
	-	-	string	direccion	
	-	-	string	tipo	
	-	-	string	comuna	
	-	-	string	region	
	-	-	bool	enVenta	
	-	-	uint	ventaActual	

Tabla 5.5: Documentación Funciones Usuario General Parte 2

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
verPropietario	string	rolProp	string	nombres	Función que retorna la información del propietario de la propiedad con rol "_rolProp". El valor de "celular" se mostrará sólo cuando sea necesario.
	-	-	string	apellido_paterno	
	-	-	string	apellido_materno	
	-	-	uint	rut	
	-	-	string	celular	
verDescripcionVenta	string	_clave	string	descripcion	Función que retorna el contrato de compra venta. Requiere permisos.
	uint	idVenta			
	string	_claveAccesoVendedor			
	string	_claveAccesoComprador			
verPrecioTramite	-	-	uint	precioTramite	Función que retorna el precio del trámite.
actualizarPrecioTramite	string	_clave	-	-	Actualiza el precio del trámite asignándole el valor "_valorWei".
	uint	_valorWei	-	-	

Tabla 5.6: Documentación Funciones Propietario

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
ponerEnVentaPropiedad	string	_rolProp	-	-	Se pondrá en venta la propiedad con rol "_rolProp".
	-	-	-	-	
	-	-	-	-	
	-	-	-	-	
cancelarVentaPropiedad	string	_rolProp	-	-	Se cancelará la venta de la propiedad con rol "_rolProp".
	-	-	-	-	
	-	-	-	-	
	-	-	-	-	
agregarDescripcionVenta	uint	idVenta	-	-	Se agregará la descripción "_descripcion" de la venta con id "_idVenta".
	string	_descripcion	-	-	
	-	-	-	-	
agregarPermisoParaFirmarVenta	uint	_idVenta	-	-	Se dará permiso a una persona con RUT "_rut" para firmar la venta con id "_idVenta".
	uint	_rut	-	-	
	-	-	-	-	
	-	-	-	-	
finalizarVentaPropiedad	string	_clave	-	-	Se finalizará la venta con id "_idVenta" de la propiedad con rol "_rolProp". El bien cambiará de dueño y se liberará el pago de la custodia con id "_idCustodia".
	string	_rolProp	-	-	
	uint	_idVenta	-	-	
	uint	_idCustodia	-	-	
	-	-	-	-	
	-	-	-	-	
trasferenciaPropiedad	string	_clave	-	-	Se transferirá el dominio de la propiedad con rol "_rolProp" a la persona con RUT "_rut".
	string	_rolProp	-	-	
	uint	_rut	-	-	
	-	-	-	-	
	-	-	-	-	
	-	-	-	-	
	-	-	-	-	

Tabla 5.7: Documentación Funciones Comprador

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
firmarCompraPropiedad	uint	_idVenta	-	-	Se firmará la venta con id "_idVenta" con el hash de la clave pública del usuario que llama a la función.
	-	-	-	-	
	-	-	-	-	
	-	-	-	-	
agregarDocumentoCustodia	string	_clave	-	-	Se agrega un documento de custodia con id "_idCustodia" junto con información relevante para la misma.
	uint	_idCustodia	-	-	
	uint	_idVenta	-	-	
	uint	_rutVend	-	-	
	uint	_monto	-	-	
	-	-	-	-	
	-	-	-	-	
devolverPagoCustodia	string	_clave	-	-	Se finaliza con error la venta correspondiente a la custodia con id "_idCustodia", la propiedad correspondiente no estará a la venta luego de esta acción.
	uint	_idCustodia	-	-	
	-	-	-	-	
	-	-	-	-	
	-	-	-	-	

Tabla 5.8: Documentación Funciones Vendedor y Comprador

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
darPermisoTemporalVenta	string	_clave	-	-	Se dará permiso a una persona de RUT "_rut" para poder ver la información de la venta con id "_idVenta". Se guardará el acceso para dicha persona "_claveAcceso" dependiendo de quien llame a la función (vendedor o comprador).
	uint	_idVenta	-	-	
	uint	_rut	-	-	
	string	_claveAcceso	-	-	
	-	-	-	-	
verEstadoVenta	uint	_idVenta	string	estado	Se retornará el estado y el siguiente paso a seguir de una venta con id "_idVenta".
	-	-	string	siguientePaso	
	-	-	-	-	
soyEnCustodia	uint	_idCustodia	string	tipoUsuario	Se retornará el tipo de usuario (vendedor o comprador) correspondiente a la custodia con id "_idCustodia". El valor depende de quien llame a la función.
	-	-	-	-	
	-	-	-	-	

Por rol que cumplen en la aplicación se muestra las funcionalidades de consulta del sistema (Tabla 5.9) y las funciones internas (Tabla 5.10), estas últimas se caracterizan por invocarse exclusivamente por otras funciones del contrato inteligente. Se precisa para cada una de ellas los parámetros de entrada y salida, como también una descripción de la acción que realiza internamente. Además, se muestran las distintas restricciones de ejecución, indicando datos de entrada y el detalle de lo que efectúa.

Tabla 5.9: Documentación Funciones Consulta del Sistema

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
soyCreador	-	-	bool	soyCreador	Se retornará "true" o "false" dependiendo si el usuario que llama a la función es creador o no.
soyTasador	-	-	bool	soyTasador	Se retornará "true" o "false" dependiendo si el usuario que llama a la función es tasador o no.
soyPropietario	-	-	bool	soyPropietario	Se retornará "true" o "false" dependiendo si el usuario que llama a la función es propietario o no.
soyUsuario	-	-	bool	soyUsuario	Se retornará "true" o "false" dependiendo si la persona que llama a la función es usuario o no.
getNombreUsuario	-	-	string	nombres	Se retornará el nombre completo del usuario que llama a la función.
getApellidoPaternoUsuario	-	-	string	apellidoPaterno	Se retornará el apellido paterno del usuario que llama a la función.
getApellidoMaternoUsuario	-	-	string	apellidoMaterno	Se retornará el apellido materno del usuario que llama a la función.
getRutUsuario	-	-	uint	rut	Se retornará el RUT sin dígito verificador del usuario que llama a la función.
getDigitoUsuario	-	-	string	digitoVerificador	Se retornará el dígito verificador del RUT del usuario que llama a la función.
soyEnVenta	uint	_idVenta	string	soyEnVenta	Se retornará si un usuario es "vendedor", "comprador" o "externo" en una venta con id "_idVenta"
	-	-	-	-	
rutEstaRegistrado	uint	_rut	bool	rutRegistrado	Se retornará "true" o "false" si el RUT indicado en "_rut" esta registrado o no en la plataforma.

Tabla 5.10: Documentación Funciones Internas

Función	Parámetros de Entrada		Parámetros de Salida		Descripción
	Tipo	Nombre	Tipo	Nombre	
agregarVenta	string	_rolProp	uint	idVenta	Crea una nueva venta conectándola con la propiedad con rol "_rolProp" y con el vendedor con address "_vendedor".
	address	_vendedor	-	-	
agregarCustodia	uint	_idCustodia	-	-	Crea una nueva custodia con id "_idCustodia", añadiendo al vendedor a partir de "_rutVend", al comprador de forma interna y asignando un valor "_monto".
	uint	_rutVend	-	-	
	uint	_monto	-	-	
getNumeroPropietario	string	_rolProp	string	celular	Si la propiedad con rol "_rolProp" está a la venta y la venta correspondiente aún no tiene un comprador se retornará el "celular" del propietario, en caso contrario se devolverá "NO DISPONIBLE".

6 Conclusiones

Esta investigación presentó una plataforma de inscripción de inmuebles que permite además la venta de los mismos. Se utilizarán los sistemas API de Registro Civil, Servicio de Impuestos Internos y los Bancos para comprobar la veracidad de los datos que se ingresarán a la aplicación. Al no existir las APIs de estos entes necesarios se desarrollaron las propias para simular completamente el funcionamiento de la aplicación.

Blockchain proporciona inmutabilidad y validez instantánea de los datos almacenados de manera descentralizada en ella, por lo que sólo las personas que son dueños de los mismos podrán modificarlos. Es por esto que la información personal de los usuarios ingresada en la plataforma estará resguardada y quien desee verla deberá tener los permisos correspondientes. Al gestionar las propiedades y los contratos de compraventa en la cadena de bloques se reemplazarán del proceso actual a los conservadores de bienes raíces y a los notarios. En definitiva, se genera un impacto económico en la sociedad, disminuyendo notoriamente los montos desembolsados en el procedimiento actual.

Como se ha exhibido en el documento en la literatura disponible no de aborda el uso de la Blockchain de Ethereum en la inscripción de propiedades, pero si en otros ámbitos. Uno de ellos es la gobernanza organizacional y toma de decisiones planteado por The DAO [12]. El otro es Gnosis que otorga un lugar para la creación de aplicaciones descentralizadas para el mercado predictivo [13].

En definitiva, la aplicación propuesta resuelve los problemas de centralización de los datos y los importantes desembolsos de dinero que deber realizar las personas tanto a conservador de bienes raíces como a los notarios. El primero, generado por el sistema actual en el cual se llevan los registros de propiedades [2][3]. El segundo, se gesta de los altos cobros que realizan las notarías según lo indicado en un estudio de julio de 2018 de la Fiscalía Nacional Económica [5].

Blockchain es una tecnología disruptiva surgida en el año 2009 a través de Bitcoin [8] mediante un sistema de transacciones sin la dependencia de un banco como validador de las mismas. Ethereum agregó a la cadena de bloques normal un lenguaje de programación creando una plataforma para desarrollar aplicaciones descentralizadas, ampliando más el uso de esta. Si bien actualmente esta técnica esta siendo un gran aporte a nivel mundial aún no alcanza su máximo potencial y está en constante evolución.

Dentro de los usos que se le puede dar a Blockchain están la democracia, utilizándose en votaciones totalmente independientes y transparentes, identidad digital, en el ámbito del IoT, entre otras. La cadena de bloques tiene el potencial de cambiar el Internet como se conoce actualmente, en donde las empresas son dueñas de los datos, y generar una red que otorga a los usuarios el control sobre su información personal. Es por esto que se invita al lector a seguir el desarrollo de esta tecnología ya que va a revolucionar el ciberespacio.

Referencias

[1] Yoshinori Matsunobu. *Migrating a database from InnoDB to MyRocks*. Core Data, Data Infrastructure. Facebook. Disponible vía web en https://code.fb.com/data-infrastructure/migrating-a-database-from-innodb-to-myrocks/. Revisada por última vez el 03 de diciembre de 2018.

[2] Sebastián Vedoya M. *Conservadores lanzan web para trámites online*. La Tercera. Disponible vía web en https://www.latercera.com/nacional/noticia/conservadores-lanzan-web-tramites-online/221826. Revisada por última vez el 03 de diciembre de 2018.

[3] Cristóbal Torres. *Conservador de Bienes Raíces de Santiago lanza plataforma digital para realizar trámites*. Bío-bío Chile. Disponible vía web en https://www.biobiochile.cl/noticias/nacional/region-metropolitana/2018/06/27/conservador-de-bienes-raices-de-santiago-lanza-plataforma-digital-para-realizar-tramites.shtml. Revisada por última vez el 03 de diciembre de 2018.

[4] Edmundo Rojas García. *Aranceles de los Conservador de Bienes Raíces: Incidencia en el costo de operaciones inmobiliarias*. Revista Fojas. Disponible vía web en http://fojas.conservadores.cl/articulos/derechos-arancelarios-del-conservador-de-bienes-raices-y-su-incidencia-en-el-costo-de-las-operaciones-inmobiliarias. Revisada por última vez el 03 de diciembre de 2018.

[5] Sebastián Castro Quiroz, Felipe Castro Altamirano, Jacinta Diestre Jullian, Luis Muñoz Chaparro. *Estudio de Mercado sobre Notarios (EM02-2017)*. División Estudios de Mercado. Fiscalía Nacional Económica. Disponible vía web en http://www.fne.gob.cl/wp-content/uploads/2018/07/Informe-Final-optimizado.pdf. Revisada por última vez el 03 de diciembre de 2018.

[6] Ministerio de Justicia. *Código Orgánico de Tribunales Ley 7421*. Título XI: Los Auxiliares de la Administración de Justicia, § 8. Los Conservadores, Artículo 448, Inciso 5. Disponible vía web en https://www.leychile.cl/Navegar?idNorma=25563. Revisada por última vez el 03 de diciembre de 2018.

[7] Ramón Jesús Millán Tejedor. “Distribución de libros electrónicos en redes P2P”. ACTA y CEDRO, Abril 2006.

[8] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Disponible vía web en https://bitcoin.org/bitcoin.pdf. Revisada por última vez el 03 de diciembre de 2018.

[9] Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Disponible vía web en http://gavwood.com/paper.pdf. Revisada por última vez el 03 de diciembre de 2018.

[10] Truffle. *Sweet Tools for Smart Contracts*. Disponible vía web en https://truffleframework.com/docs/truffle/overview. Revisada por última vez el 03 de diciembre de 2018.

[11] MetaMask. *Brings Ethereum to your browser*. Disponible vía web en https://metamask.io. Revisada por última vez el 03 de diciembre de 2018.

[12] Christoph Jentzsch. *Decentralized Autonomous Organization to Automate Governance*. Disponible vía web en https://download.slock.it/public/DAO/WhitePaper.pdf. Revisada por última vez el 03 de diciembre de 2018.

[13] Martin Köppelmann, Stefan George, Dr. Friederike Ernst. *Gnosis: Crowdsourced Wisdom*. Disponible vía web en https://gnosis.pm/assets/pdf/gnosis-whitepaper.pdf. Revisada por última vez el 03 de diciembre de 2018.

Anexos

A: Capturas de Pantalla del Sistema

A.1: Registro de Usuario

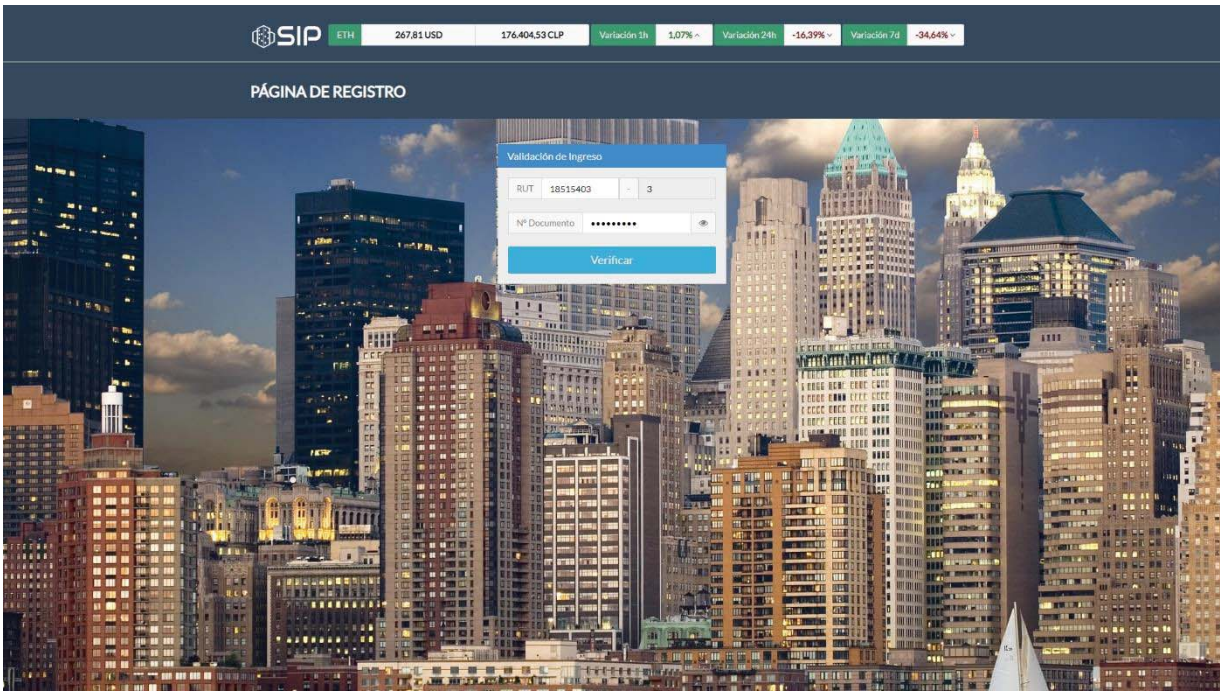

Figura A.1: Pantalla Registro de Usuario Validación

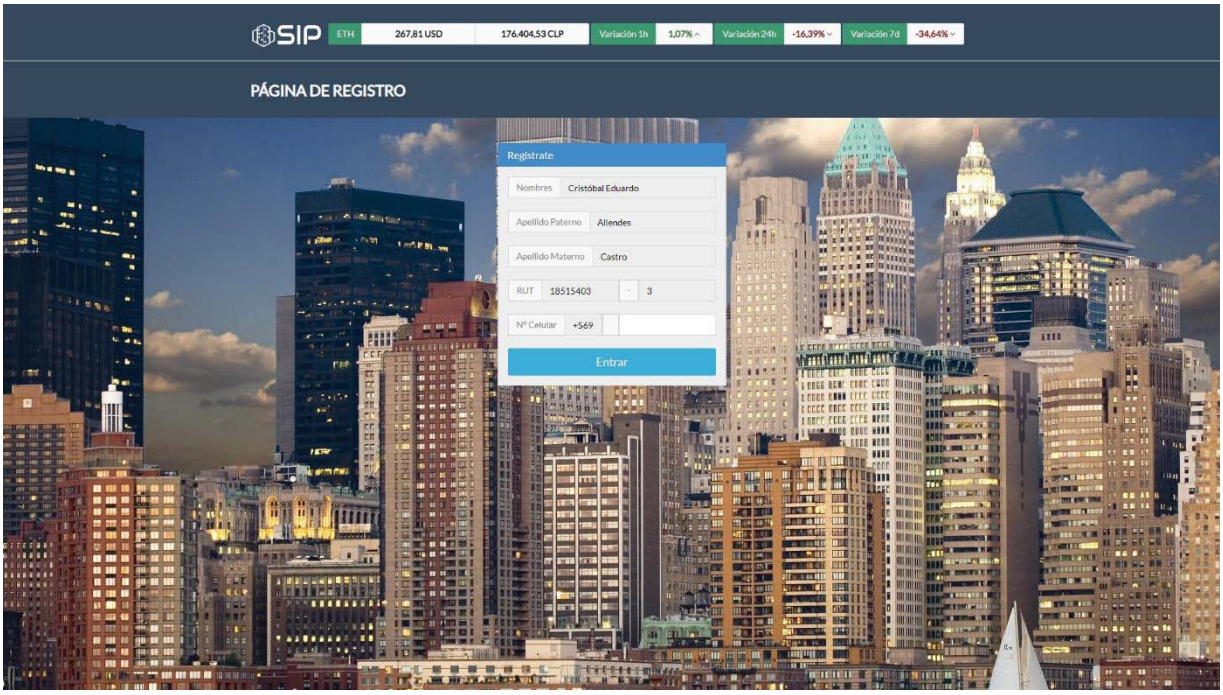

Figura A.2: Pantalla de Registro de Usuario Validado

A.2: Menú de Inicio de la Plataforma

Figura A.3: Pantalla Menú de Inicio de la Plataforma

A.3: Funciones Creador

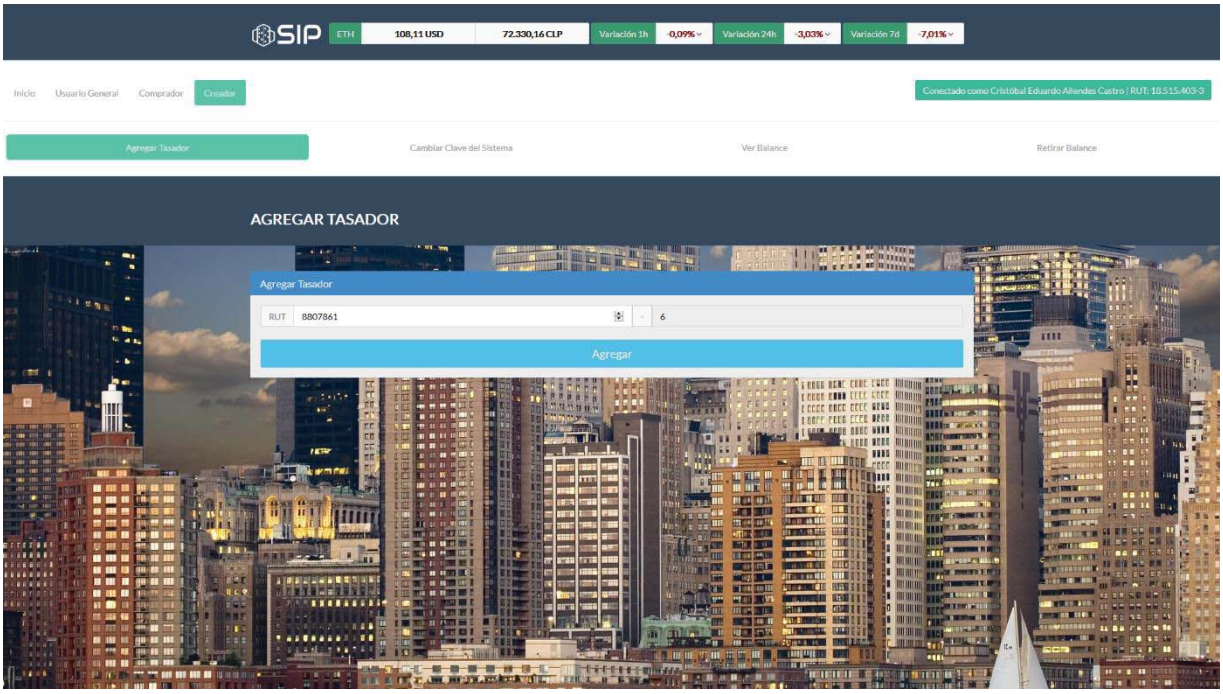

Figura A.4: Pantalla Agregar Tasador

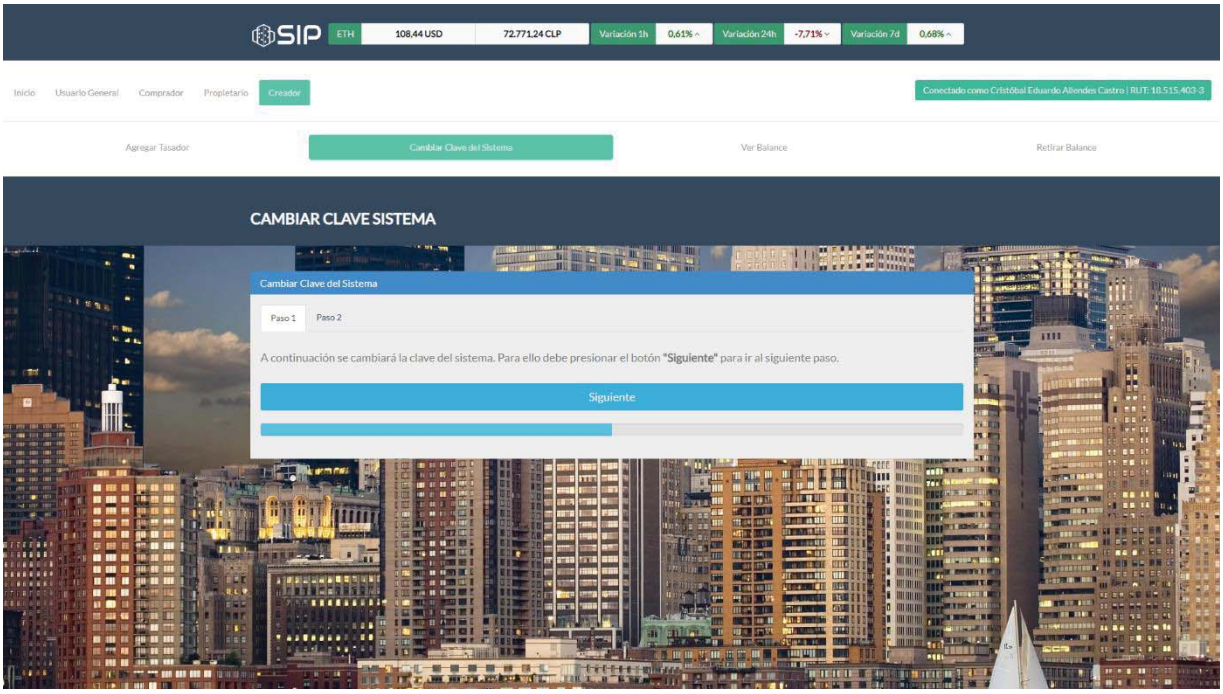

Figura A.5: Pantalla Cambiar Clave del Sistema Paso 1

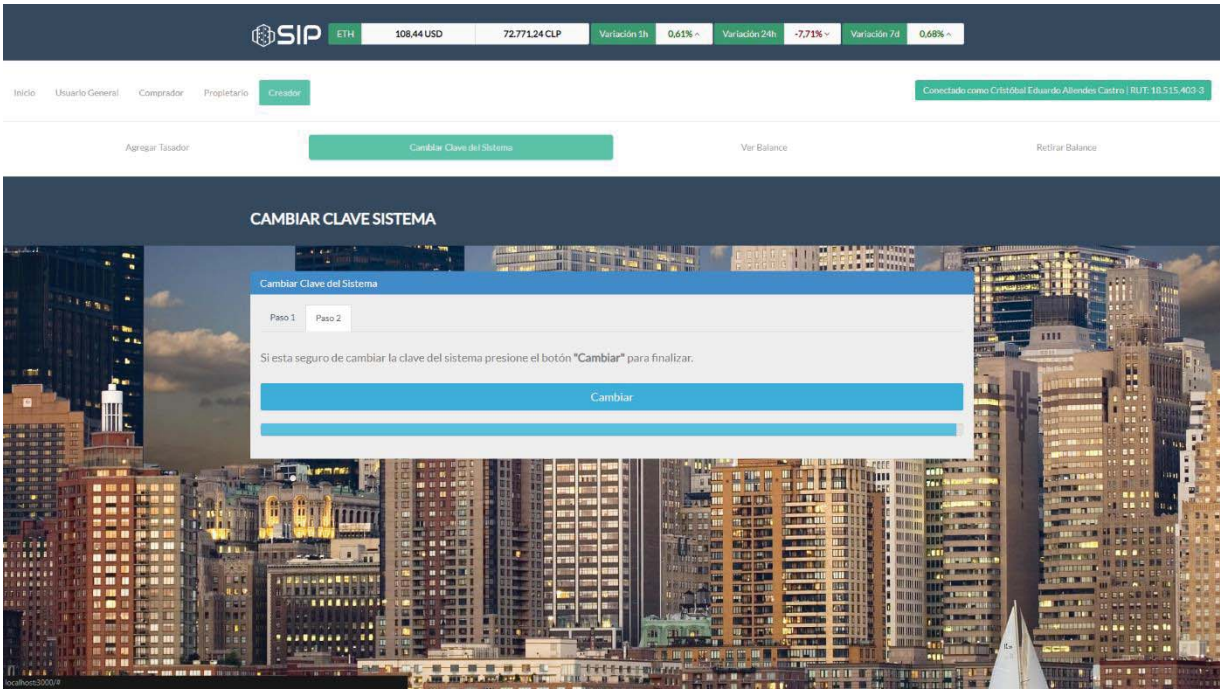

Figura A.6: Pantalla Cambiar Clave del Sistema Paso 2

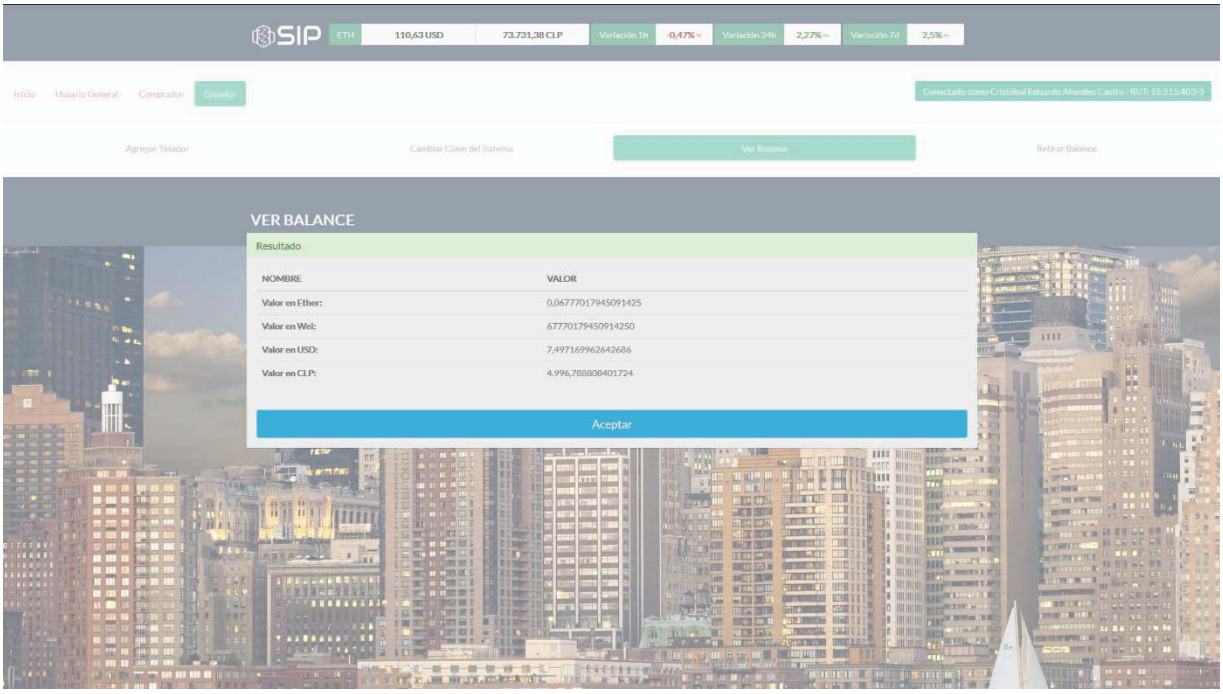

Figura A.7: Pantalla Ver Balance

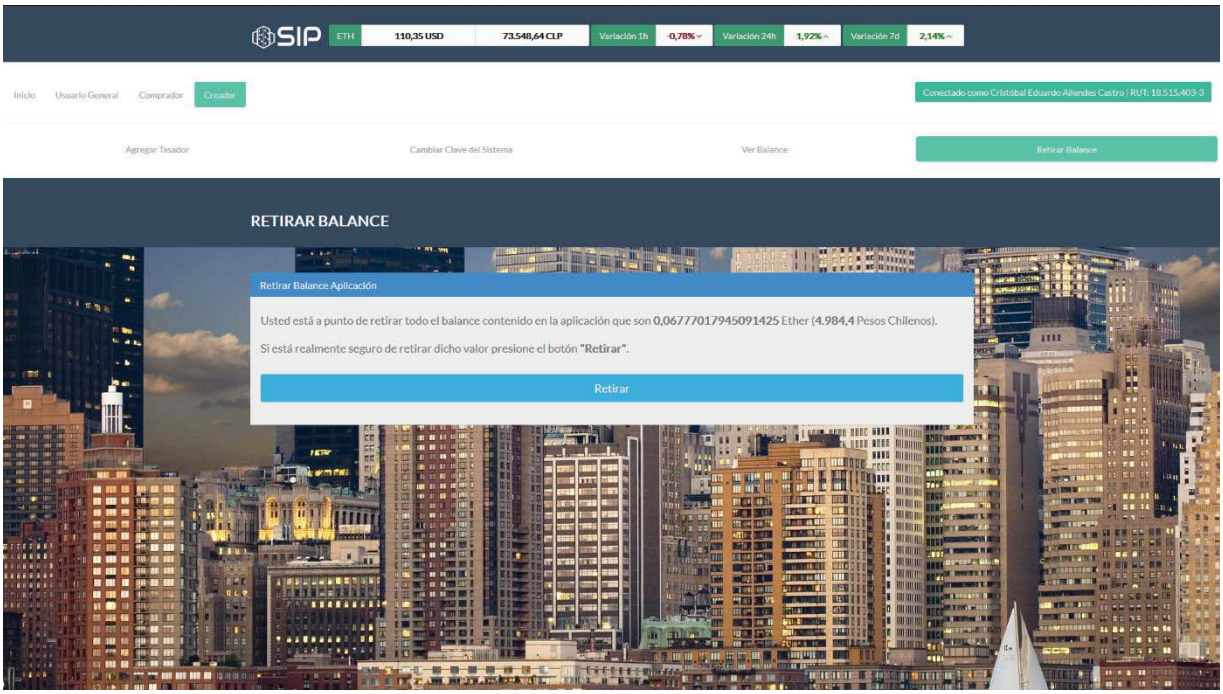

Figura A.8: Pantalla Retirar Balance

A.4: Funciones Tasador

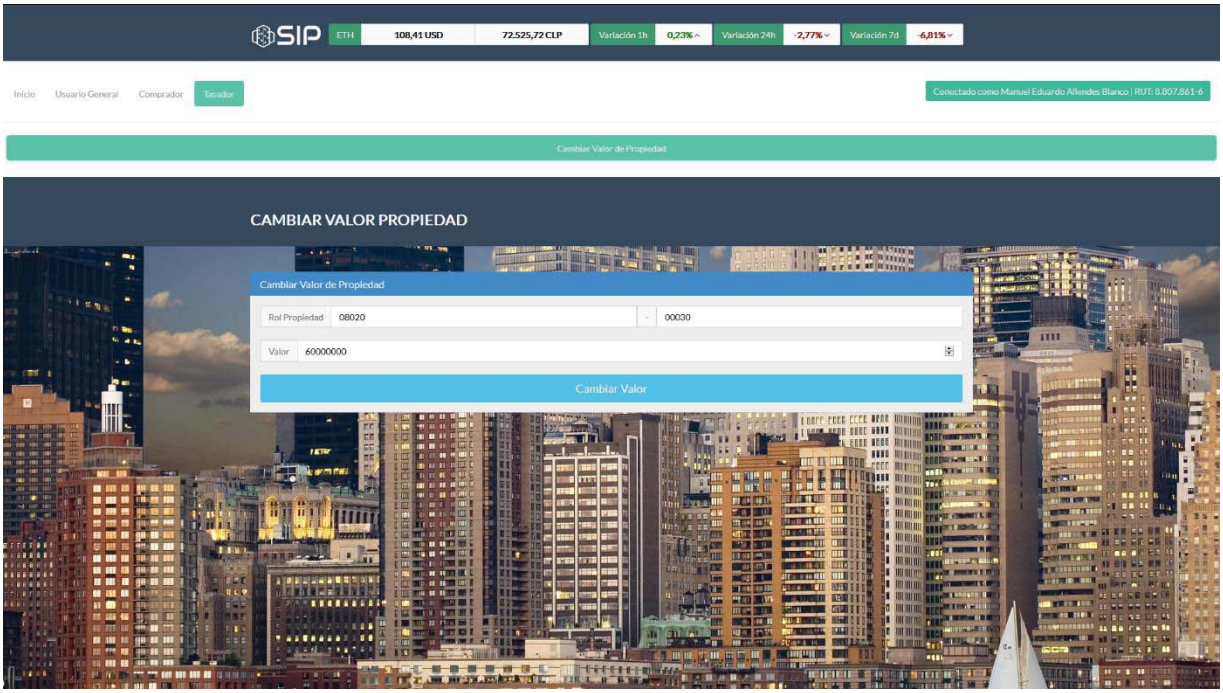

Figura A.9: Pantalla Cambiar Valor Propiedad

A.5: Funciones Usuario General

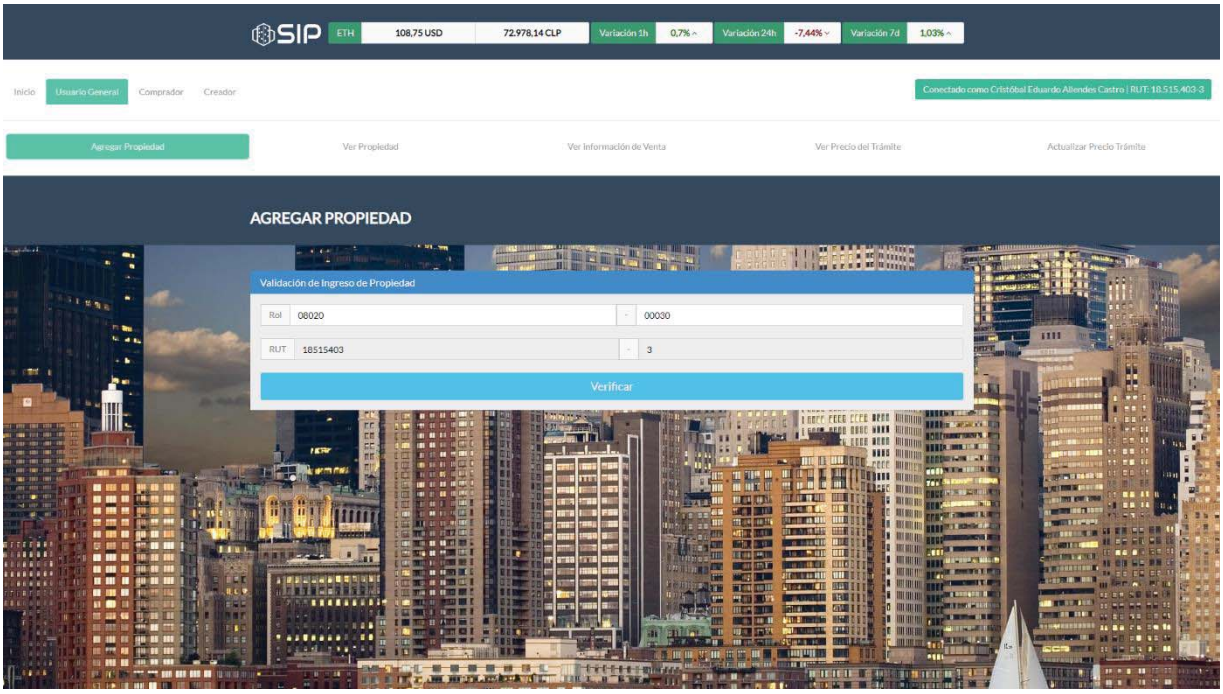

Figura A.10: Pantalla Agregar Propiedad Paso 1

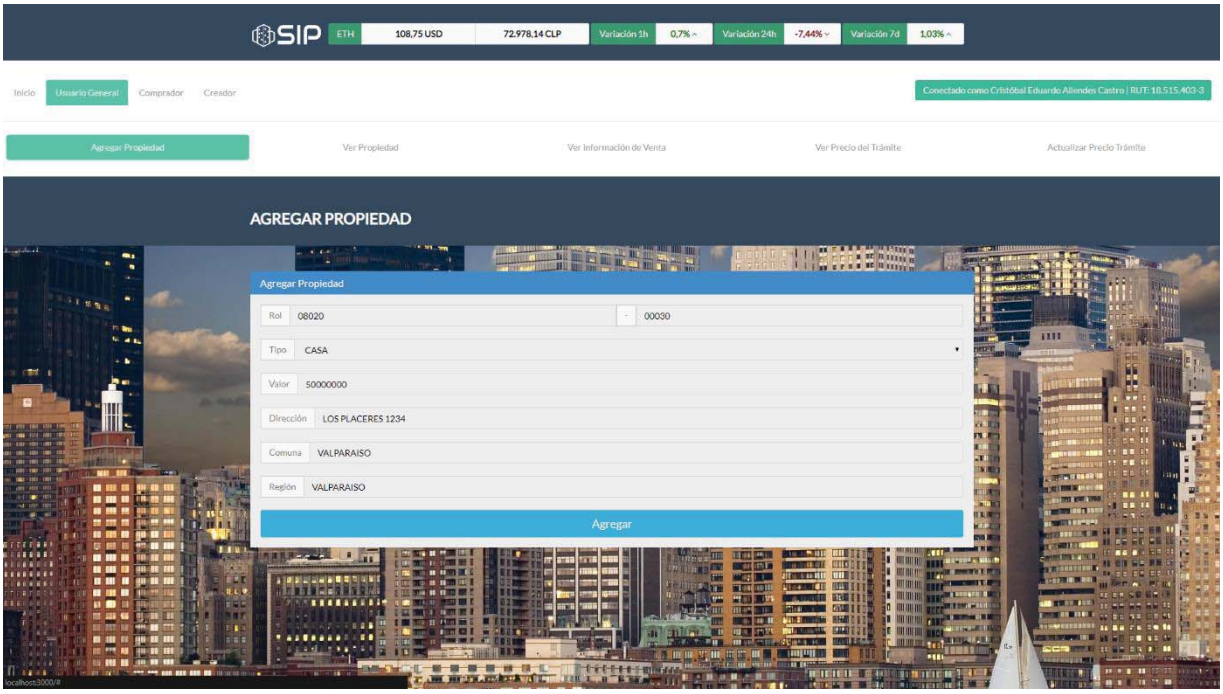

Figura A.11: Pantalla Agregar Propiedad Paso 2

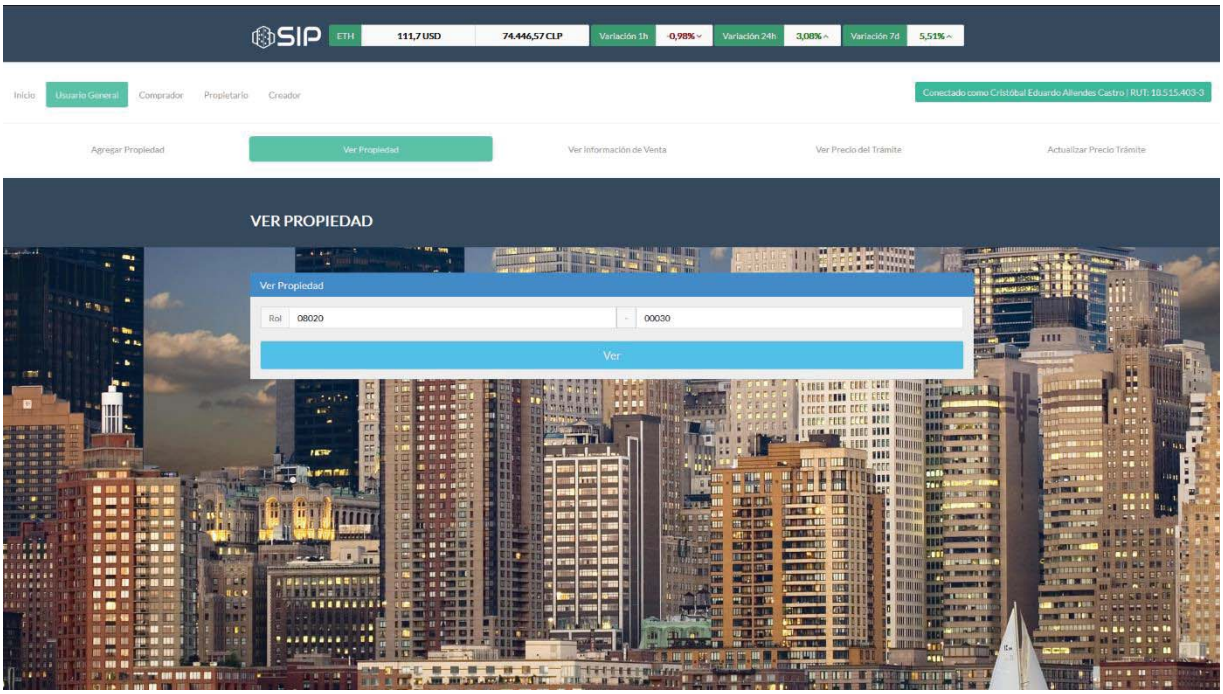

Figura A.12: Pantalla Ver Propiedad

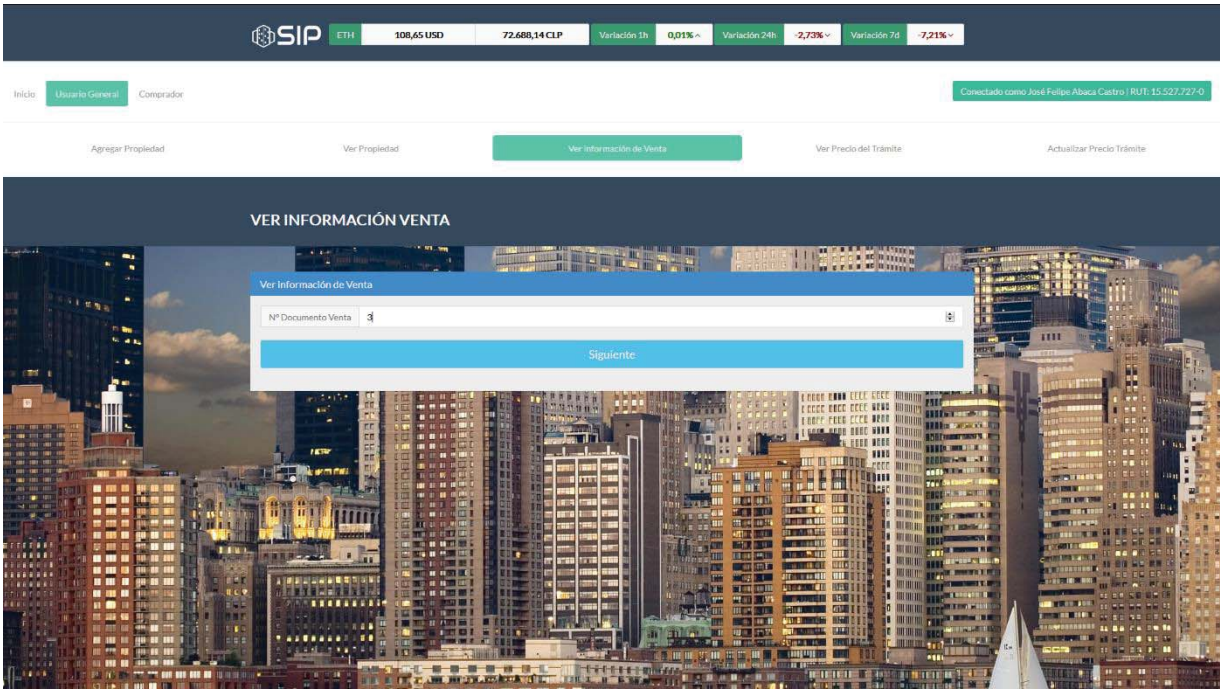

Figura A.13: Pantalla Ver Información de Venta

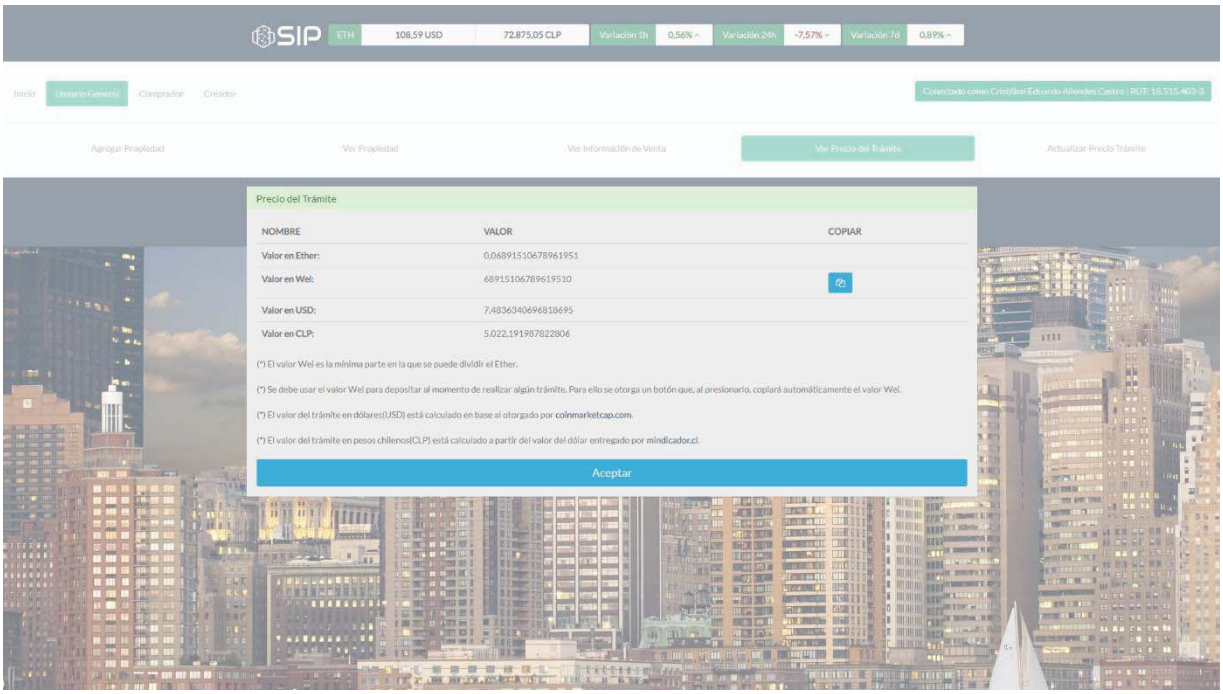

Figura A.14: Pantalla Ver Precio Trámite

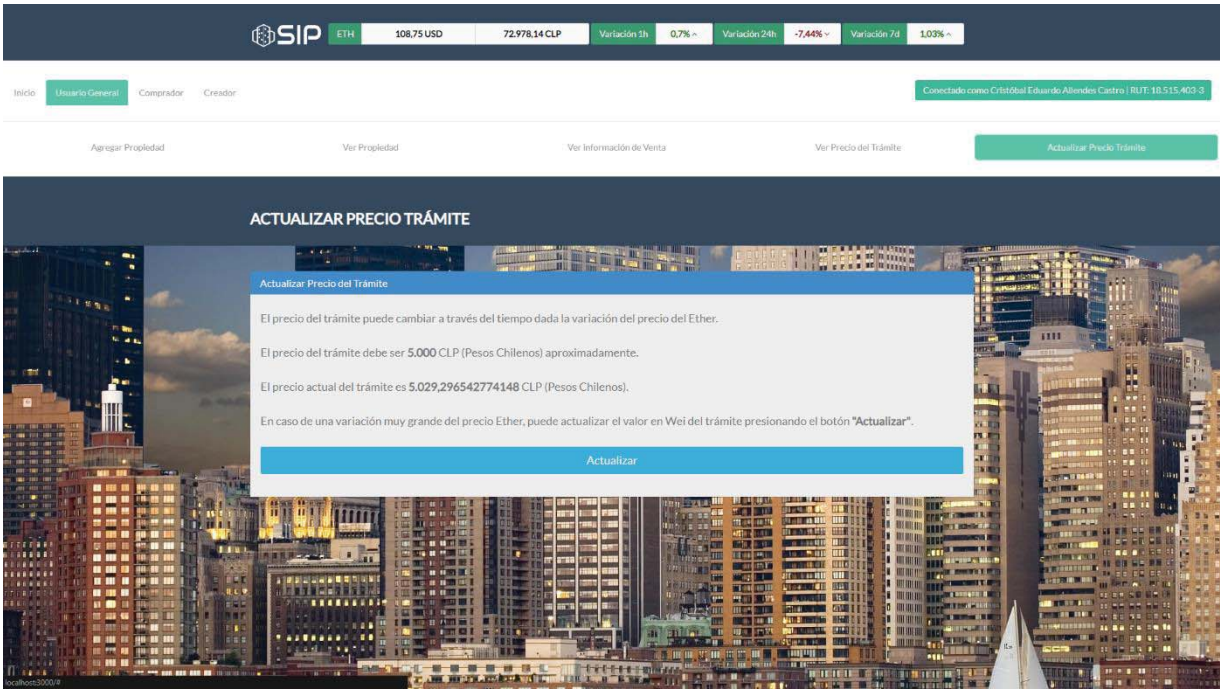

Figura A.15: Pantalla Actualizar Precio Trámite

A.6: Funciones Propietario

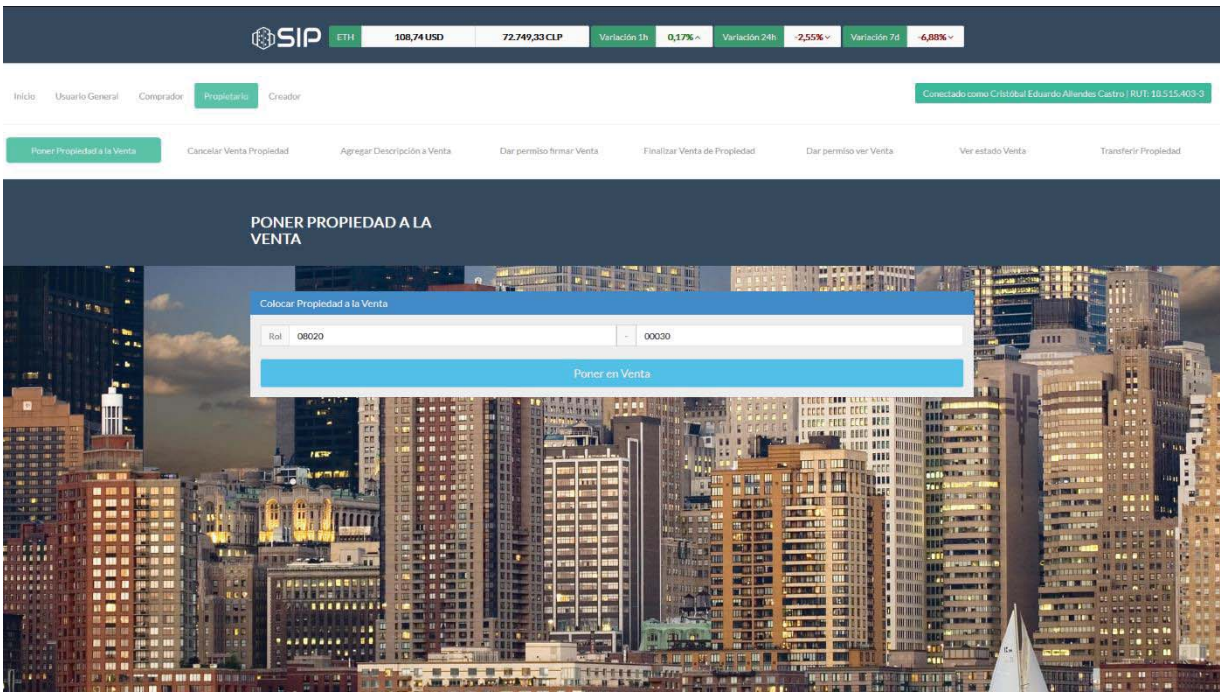

Figura A.16: Pantalla Poner Propiedad a la Venta

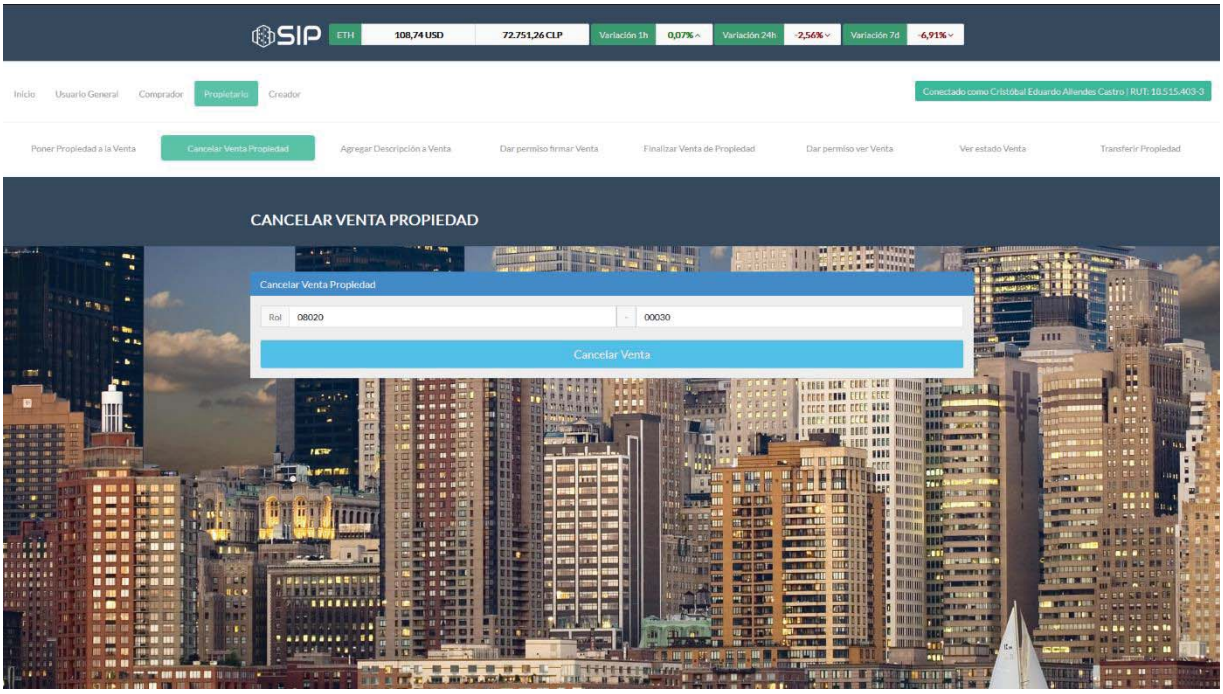

Figura A.17: Pantalla Cancelar Venta de Propiedad

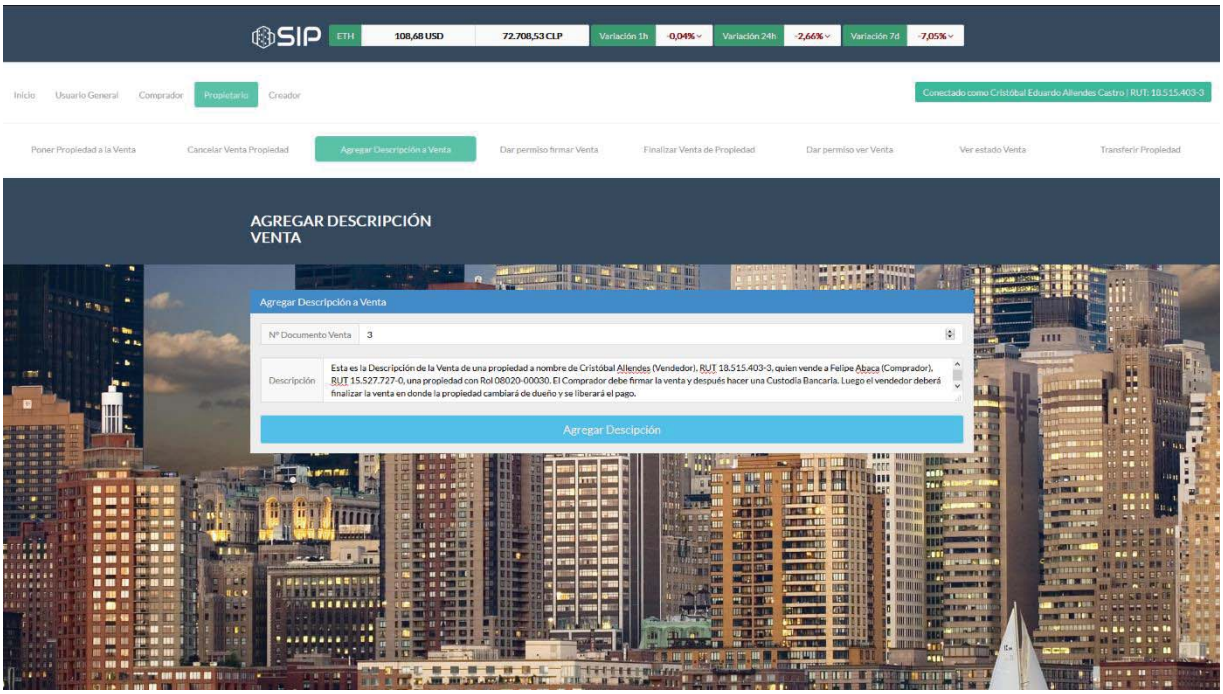

Figura A.18: Pantalla Agregar Contrato de Compraventa

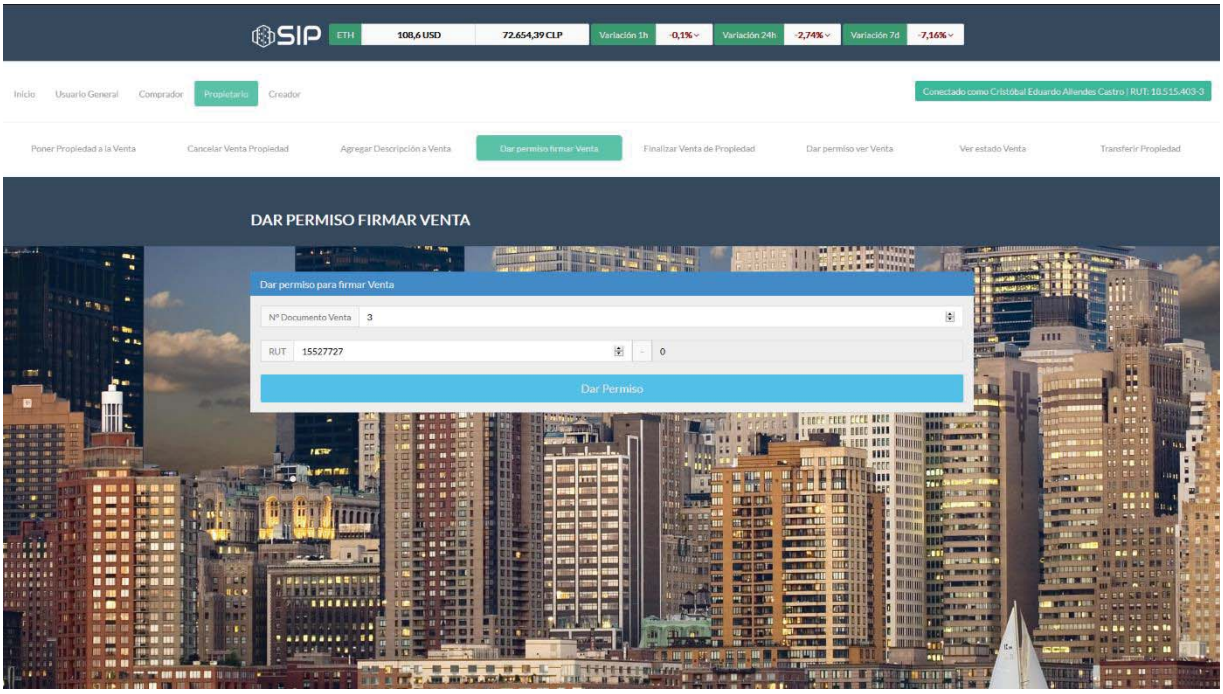

Figura A.19: Pantalla Dar Permiso Firmar Venta

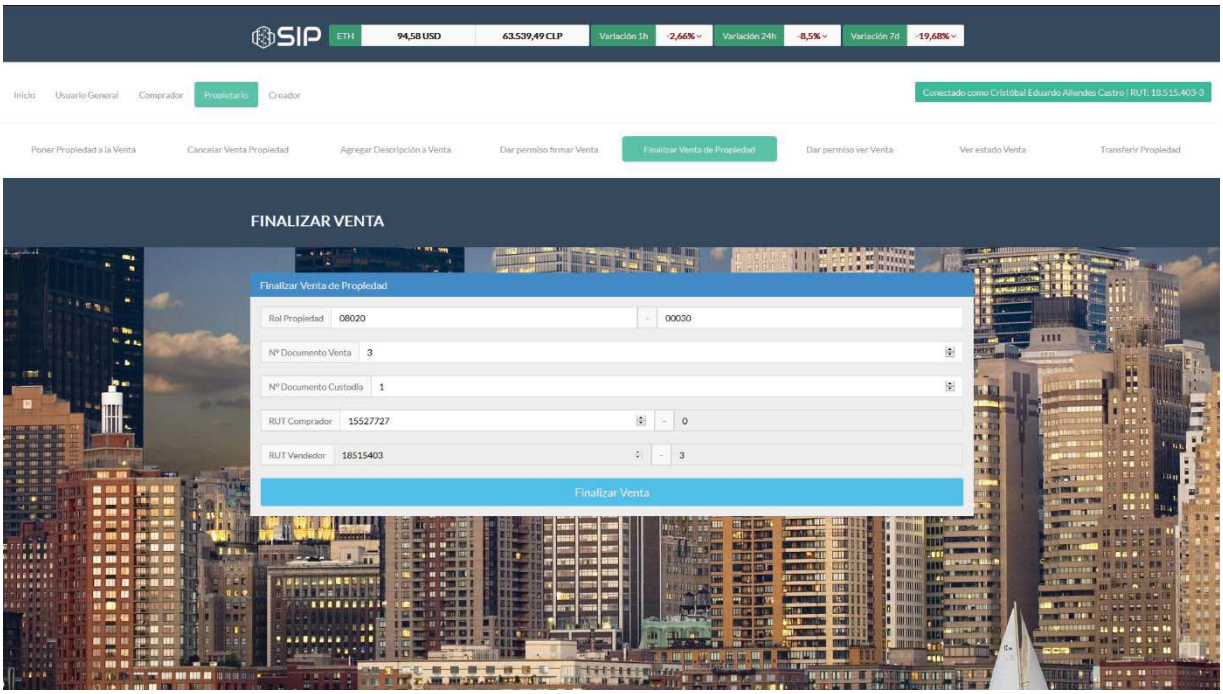

Figura A.20: Pantalla Finalizar Venta

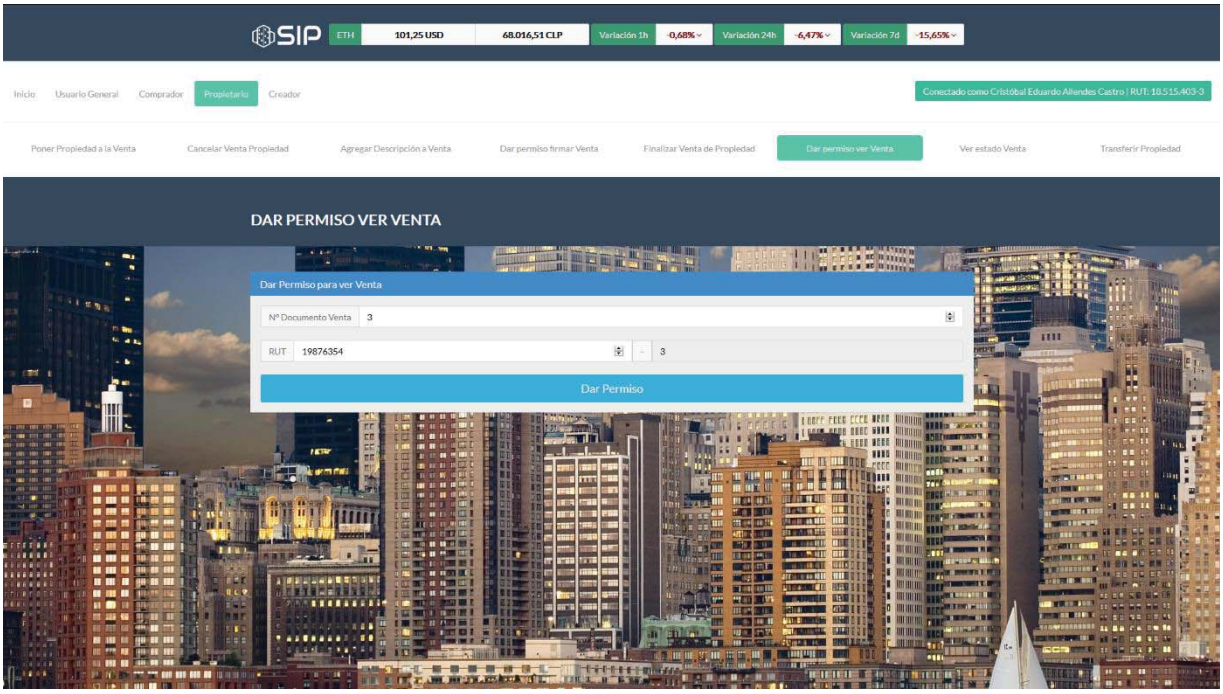

Figura A.21: Pantalla Dar Permiso Ver Venta

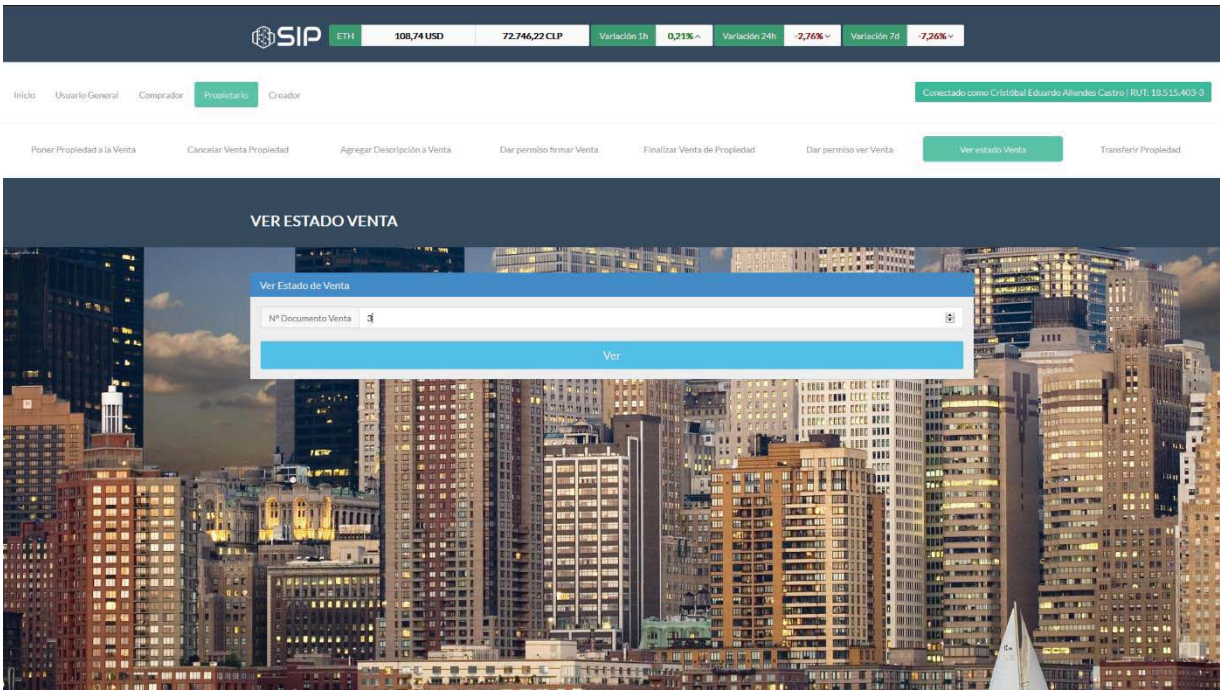

Figura A.22: Pantalla Ver Estado Venta

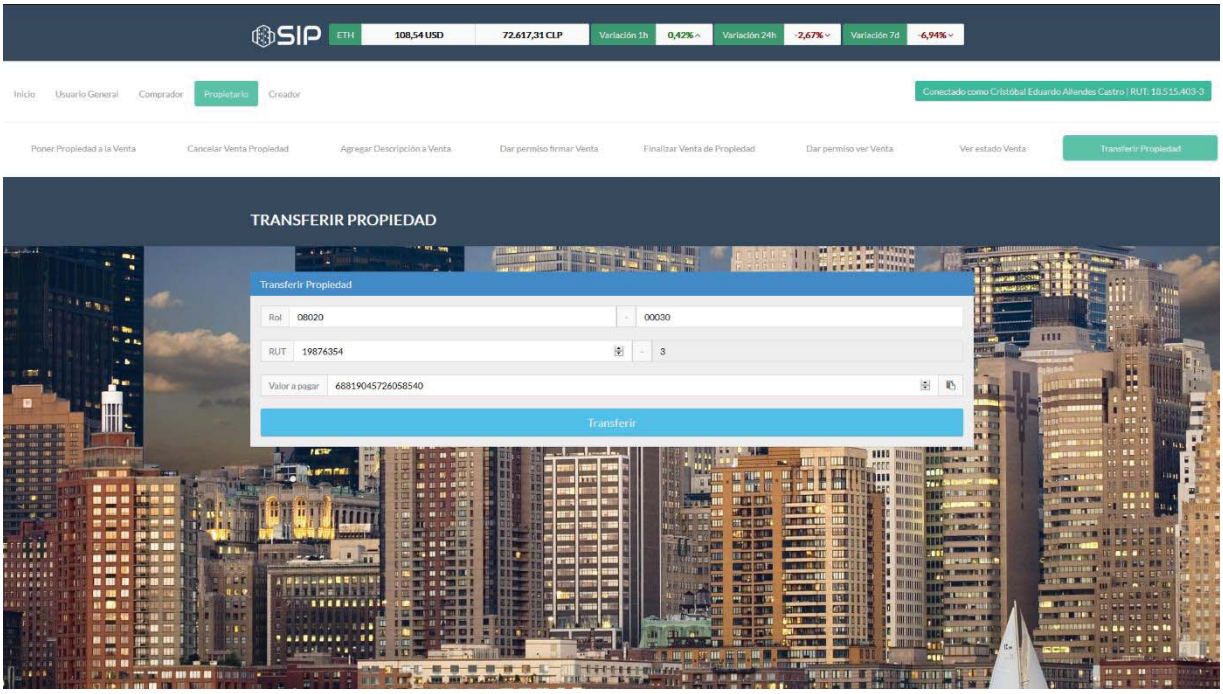

Figura A.23: Pantalla Transferir Propiedad

A.7: Funciones Comprador

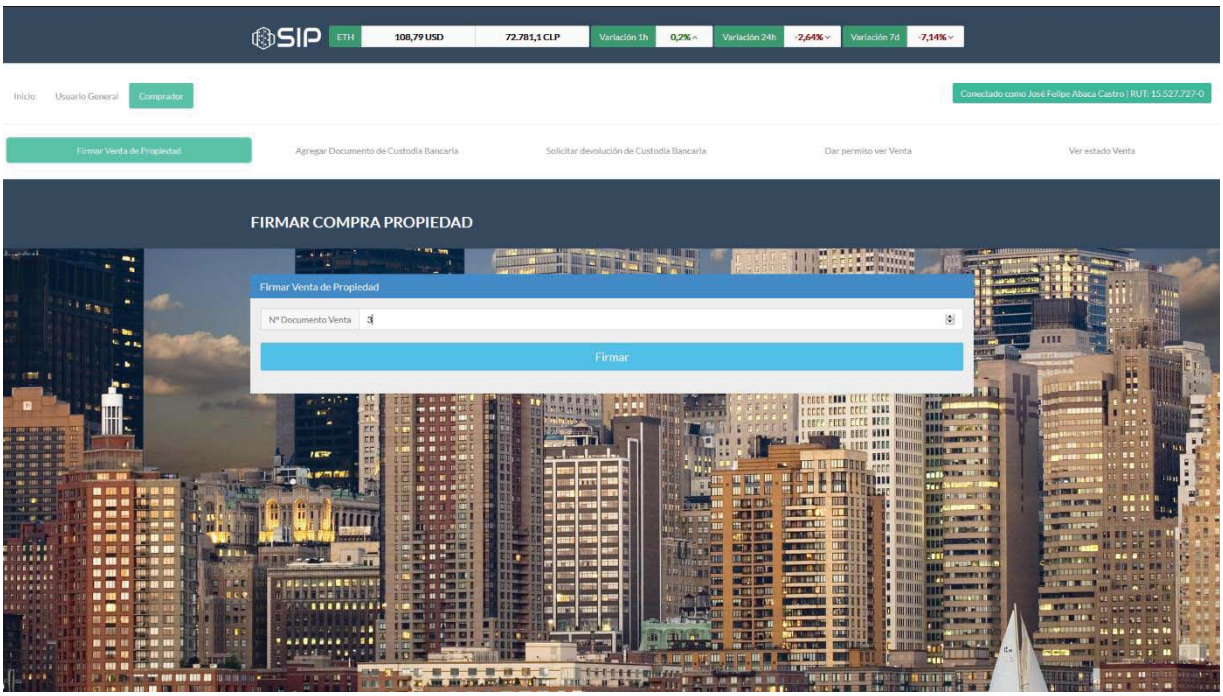

Figura A.24: Pantalla Firmar Contrato de Compraventa

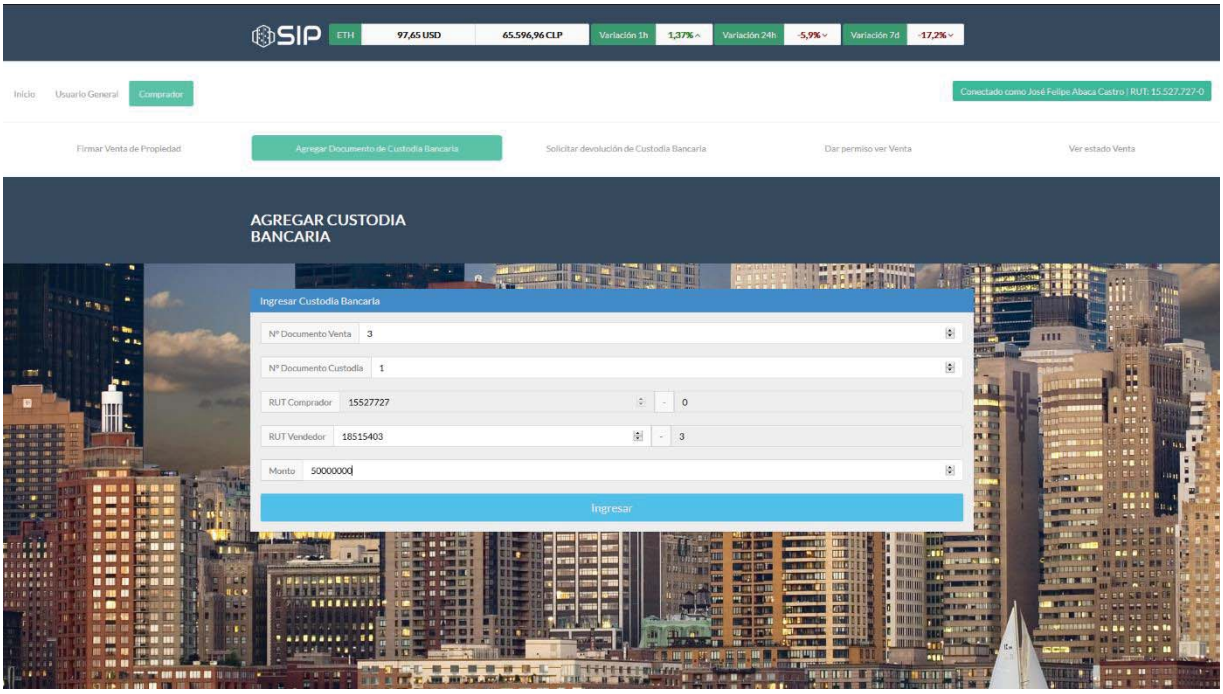

Figura A.25: Pantalla Agregar Custodia Bancaria

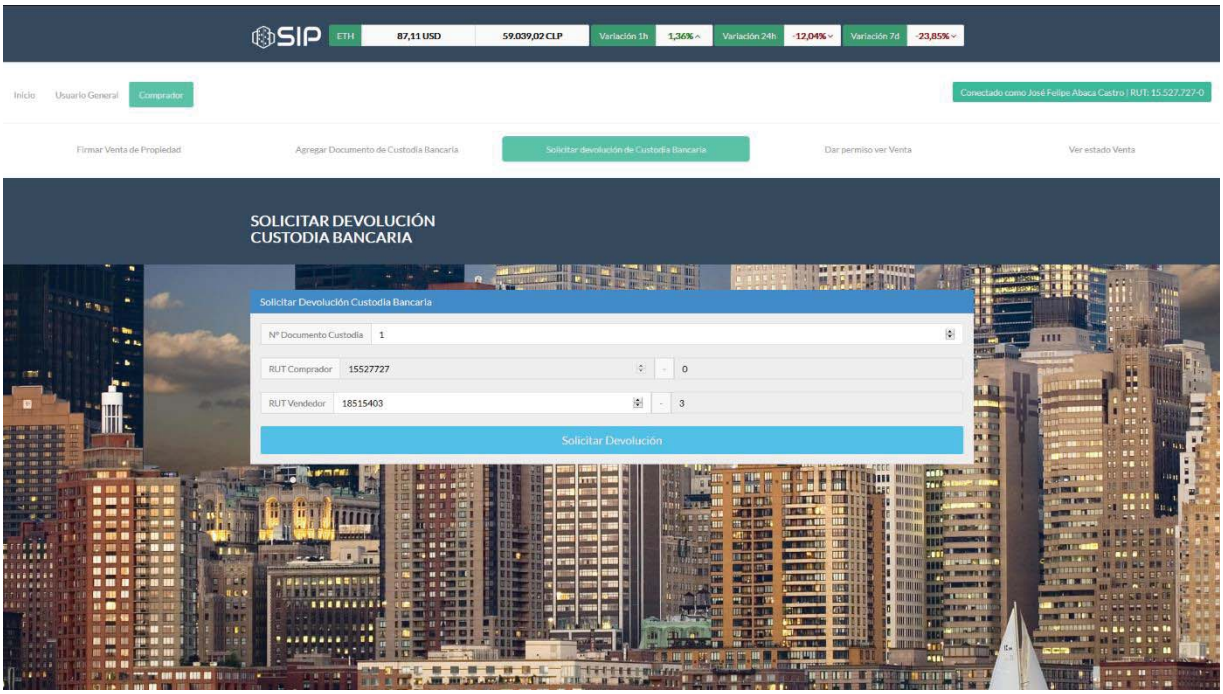

Figura A.26: Pantalla Solicitar Devolución Custodia Bancaria

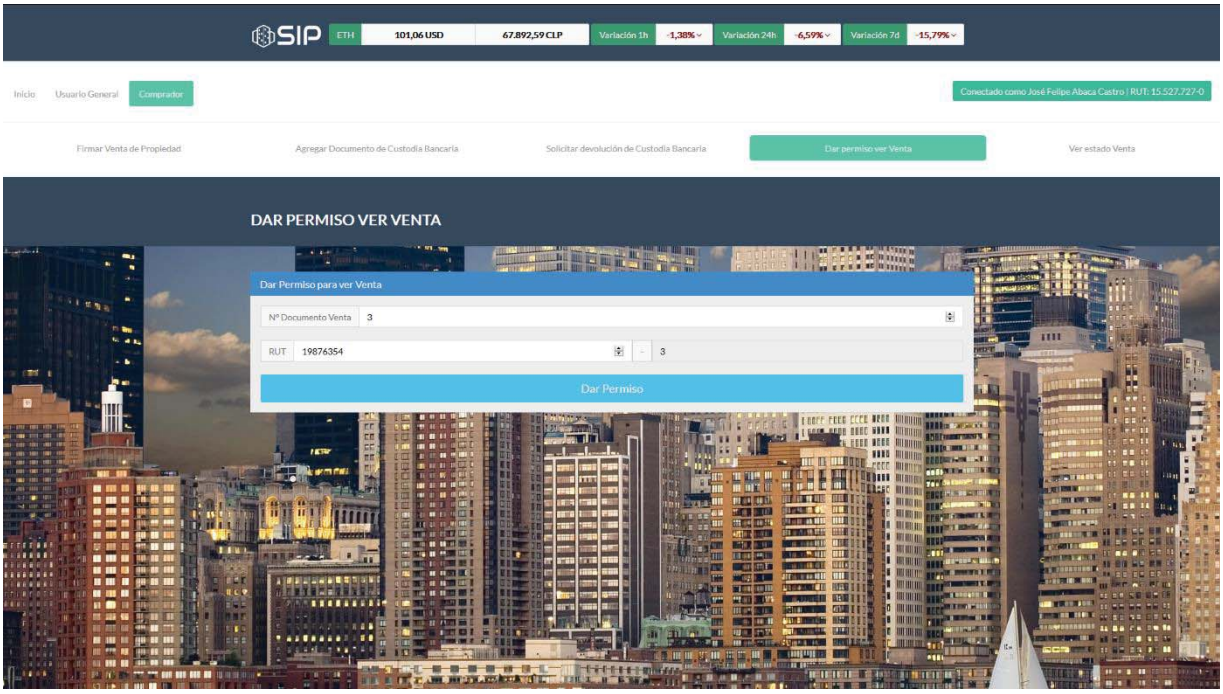

Figura A.27: Pantalla Dar Permiso Ver Venta

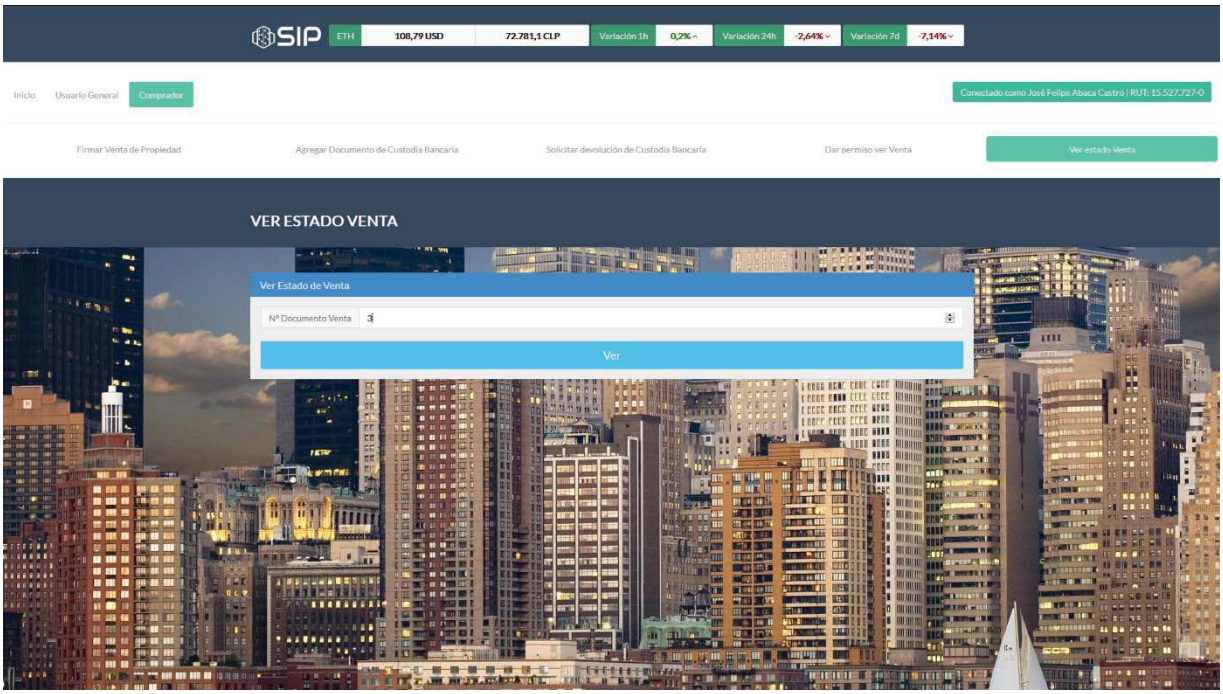

Figura A.28: Pantalla Ver Estado Venta

B: Flujo de Datos

B.1: Flujo de Datos Registro de Usuario

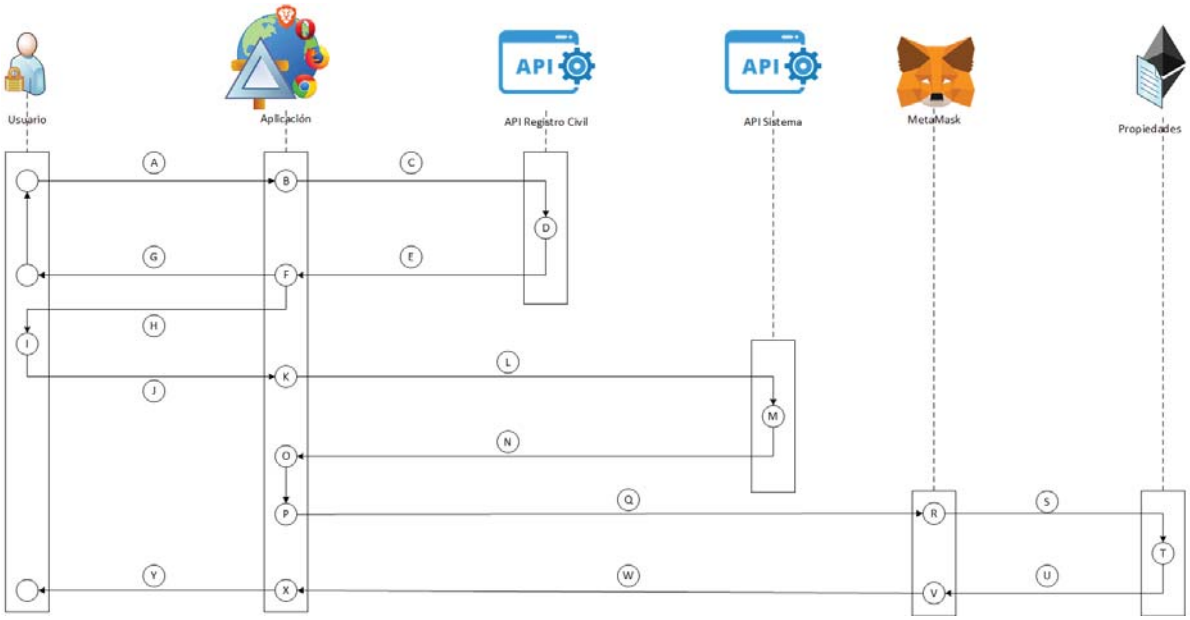

Figura B.1: Flujo de Datos Registro de Usuario

Tabla B.1: Explicación Flujo de Datos Registro de Usuario

Agregar Usuario	
Acción	**Descripción**
A	El Usuario ingresa su RUT y el número de documento de su cédula de identidad en el formulario
B	La Aplicación recibe el RUT y el número de documento
C	La Aplicación realiza una petición POST a la API del Registro Civil enviando el RUT y el hash del número del documento
D	La API del Registro Civil recibe la petición POST y procesa los datos recibidos
E	Si los datos recibidos son correctos la API del Registro Civil retornará un mensaje de éxito, el nombre completo, el apellido paterno, el apellido materno y el RUT correspondiente en formato JSON Si los datos recibidos son incorrectos la API del Registro Civil retornara un mensaje de error en formato JSON
F	La Aplicación recibe los datos del JSON
G	En caso de que el estado recibido sea un error la Aplicación retornará un mensaje de error al Usuario y éste tendrá que volver a realizar A
H	En caso de que la información sea correcta se mostrará un formulario al Usuario previamente completado con los datos recibidos en F
I	El Usuario ingresa el número de celular y presiona el botón para registrarse
J	El Usuario envía los datos del formulario a la Aplicación
K	La Aplicación recibe los datos del formulario
L	La Aplicación realiza una petición POST a la API del Sistema para recibir la clave encriptada
M	La API del Sistema recibe la petición POST y procesa los datos recibidos
N	La API del Sistema retorna la clave encriptada en formato JSON
O	La Aplicación recibe la clave encriptada y la desencripta
P	La Aplicación prepara los datos recibidos en K
Q	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
R	MetaMask recibe la petición y prepara la transacción
S	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
T	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
U	La red Ethereum retorna un mensaje de éxito o error según corresponda
V	MetaMask recibe el mensaje
W	MetaMask retorna el mensaje a la Aplicación
X	La Aplicación recibe el mensaje de MetaMask y lo procesa
Y	La Aplicación muestra al Usuario si la operación fue exitosa o si no se pudo realizar

B.2: Flujo de Datos Creador

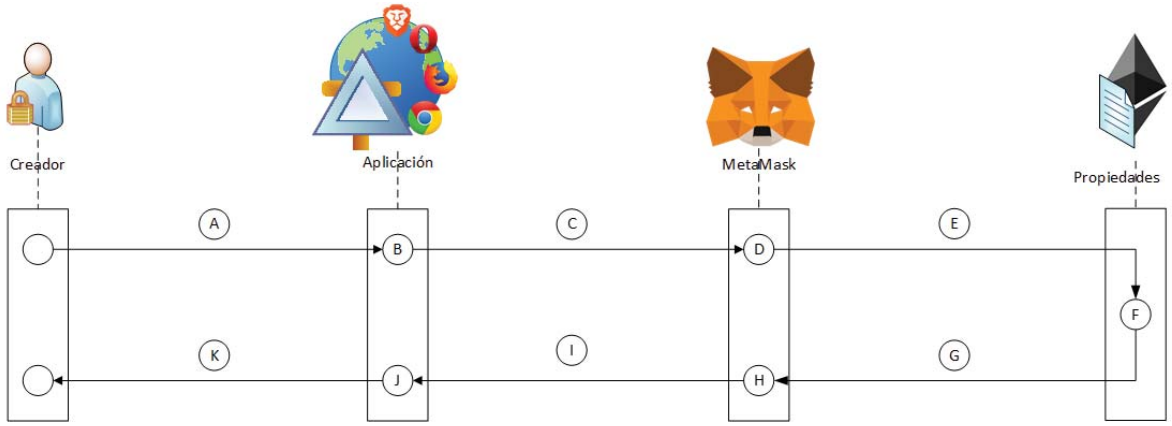

Figura B.2: Flujo de Datos Agregar Tasador

Tabla B.2: Explicación Flujo de Datos Agregar Tasador

Agregar Tasador	
Acción	**Descripción**
A	El Creador ingresa su RUT del Tasador
B	La Aplicación recibe el RUT
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Creador el resultado de la operación (éxito o error)

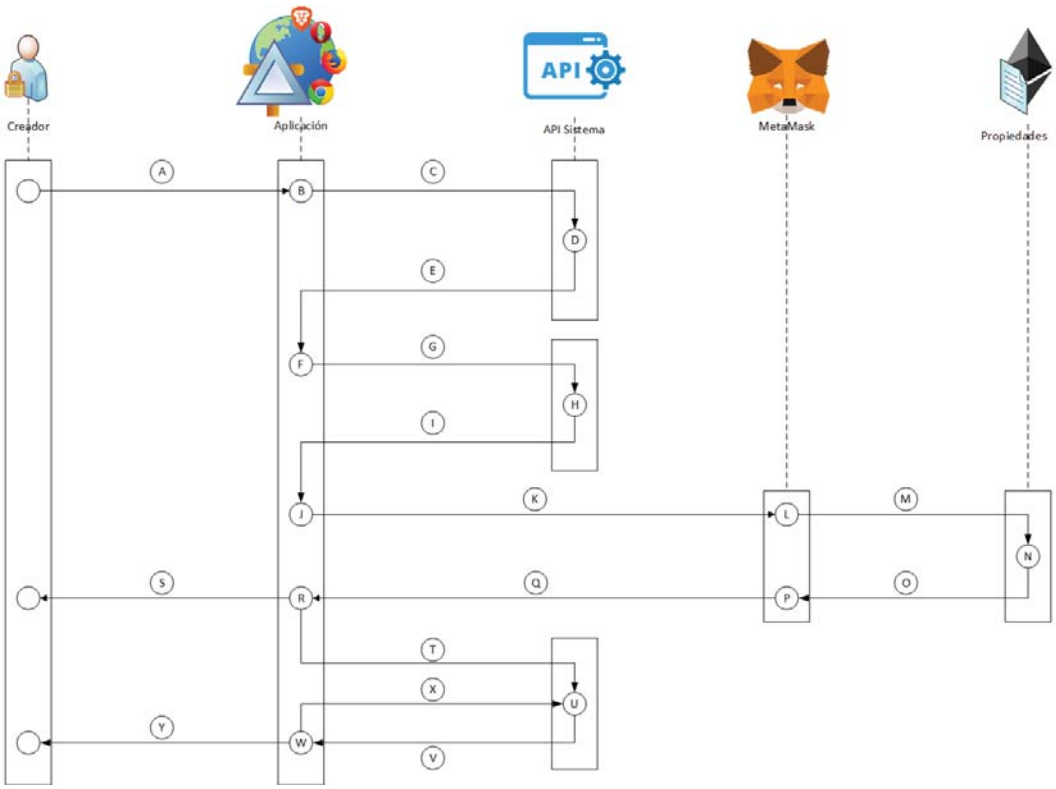

Figura B.3: Flujo de Datos Cambiar Clave del Sistema

Tabla B.3: Explicación Flujo de Datos Cambiar Clave del Sistema

Cambiar Clave del Sistema	
Acción	**Descripción**
A	El Creador pide el cambio de la clave del sistema
B	La Aplicación recibe la acción y se prepara para realizar petición
C	La Aplicación realiza una petición POST a la API para pedir la clave encriptada
D	La API de la Aplicación recibe la petición
E	La API de la Aplicación retorna la clave encriptada en formato JSON
F	La Aplicación recibe la clave encriptada y se prepara para realizar petición
G	La Aplicación realiza una petición POST a la API para cambiar la clave
H	La API de la Aplicación recibe la petición, realiza el cambio de clave y encripta la misma
I	La API de la Aplicación retorna la nueva clave encriptada en formato JSON
J	La Aplicación recibe la nueva clave encriptada y desencripta ambas claves
K	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
L	MetaMask recibe la petición y prepara la transacción
M	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
N	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
O	La red Ethereum retorna un mensaje de éxito o error según corresponda
P	MetaMask recibe el mensaje
Q	MetaMask retorna el mensaje a la Aplicación
R	La Aplicación recibe el mensaje de MetaMask y lo procesa
S	Si la operación fue exitosa la Aplicación muestra al Creador un mensaje de éxito
T	Si la operación finalizó con error la Aplicación realiza un petición POST a la API para pedir que la clave se restaure
U	La API de la Aplicación recibe la petición y restaura la clave
V	La API de la Aplicación retorna un mensaje de éxito o error según corresponda en formato JSON
W	La Aplicación recibe el mensaje de la API
X	Si hay un error la aplicación vuelve a realizar la petición para restaurar la clave
Y	Si se realizó exitosamente se mostrará el mensaje correspondiente

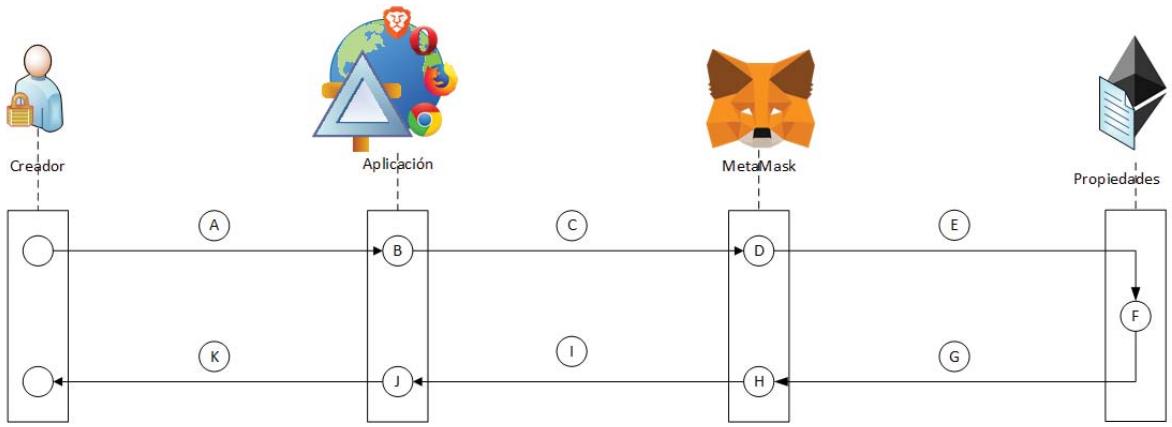

Figura B.4: Flujo de Datos Ver Balance

Tabla B.4: Explicación Flujo de Datos Ver Balance

Ver Balance	
Acción	**Descripción**
A	El Creador pide ver el balance del sistema
B	La Aplicación recibe la acción
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código de la función correspondiente
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna el balance del sistema
H	MetaMask recibe el balance
I	MetaMask retorna el balance a la Aplicación
J	La Aplicación recibe el balance de MetaMask y lo procesa
K	La Aplicación muestra al Creador el balance del sistema

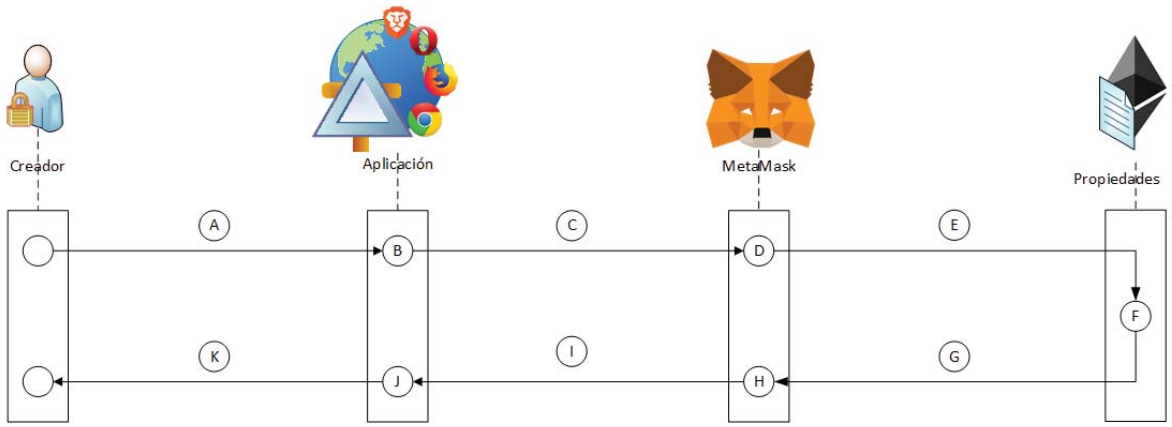

Figura B.5: Flujo de Datos Retirar Balance

Tabla B.5: Explicación Flujo de Datos Retirar Balance

Retirar Balance	
Acción	**Descripción**
A	El Creador pide retirar el balance del sistema
B	La Aplicación recibe la acción
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código de la función correspondiente
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Creador el resultado de la operación (éxito o error), si fue exitosa la persona podrá ver un aumento en el saldo de su billetera

B.3: Flujo de Datos Tasador

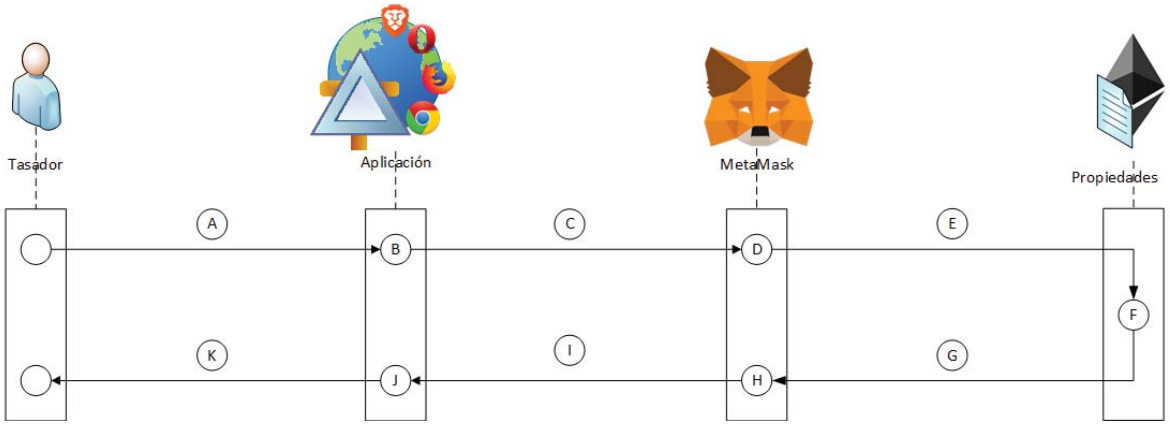

Figura B.6: Flujo de Datos Cambiar Valor Propiedad

Tabla B.6: Explicación Flujo de Datos Cambiar Valor Propiedad

Cambiar Valor de la Propiedad	
Acción	**Descripción**
A	El Tasador ingresa el rol de la propiedad y su nuevo valor
B	La Aplicación recibe los datos y los procesa
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Tasador el resultado de la operación (éxito o error)

B.4: Flujo de Datos Usuario General

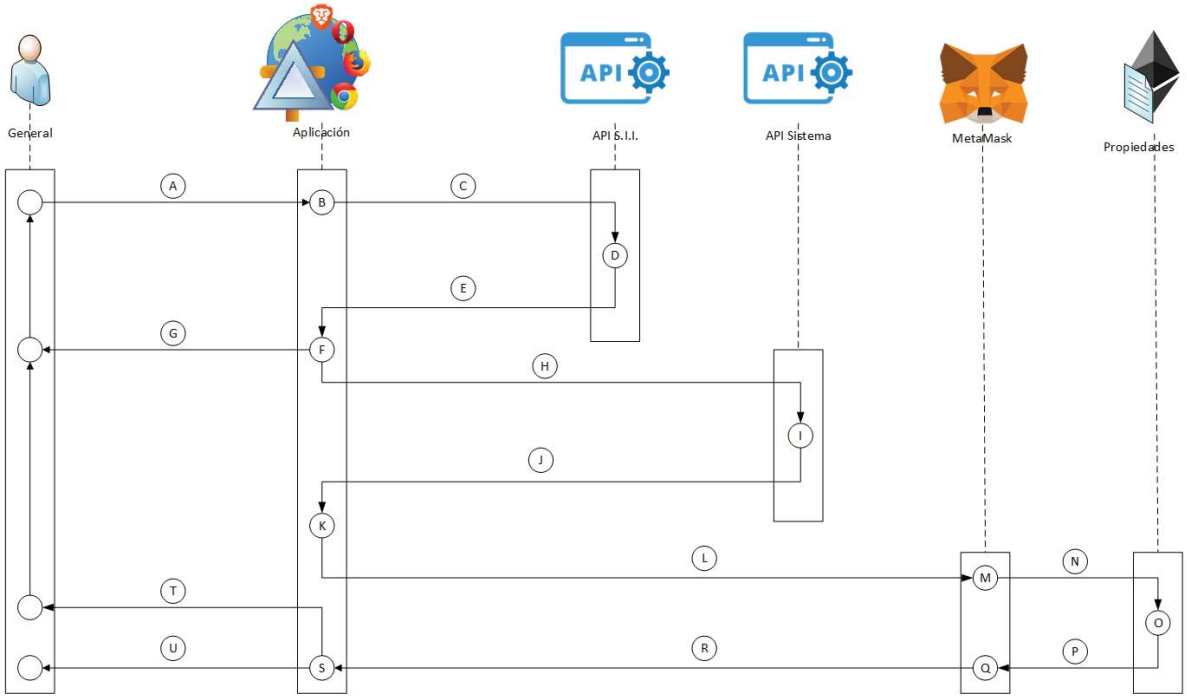

Figura B.7: Flujo de Datos Agregar Propiedad

Tabla B.7: Explicación Flujo de Datos Agregar Propiedad

Agregar Propiedad	
Acción	**Descripción**
A	El Usuario ingresa el rol de la propiedad y su RUT
B	La Aplicación recibe los datos y los procesa
C	La Aplicación realiza una petición POST a la API del S.I.I. para validar la información
D	La API del S.I.I. recibe la petición
E	La API del S.I.I. retorna un mensaje de error en formato JSON si la persona no es dueño de la propiedad
	La API del S.I.I. retorna la información de la propiedad en formato JSON si la persona es dueño de la propiedad
F	La Aplicación recibe la respuesta de la API del S.I.I.
G	Si la persona no es dueño de la propeidad la Aplicación retornará un mensaje de error al Usuario
H	Si la persona es dueño de la propiedad la Aplicación realiza una petición POST a la API del sistema pidiendo la clave
I	La API del sistema recibe la petición y la procesa
J	La API del sistema retorna la clave encriptada en formato JSON
K	La Aplicación recibe la clave, la desencripta y prepara los datos para comunicarse con MetaMask
L	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
M	MetaMask recibe la petición y prepara la transacción
N	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
O	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
P	La red Ethereum retorna un mensaje de éxito o error según corresponda
Q	MetaMask recibe el mensaje
R	MetaMask retorna el mensaje a la Aplicación
S	La Aplicación recibe el mensaje de MetaMask y lo procesa
T	Si la respuesta es un error la Aplicación muestra al Usuario el mensaje correspondiente
U	Si la respuesta es exitosa la Aplicación muestra al Usuario un mensaje de éxito

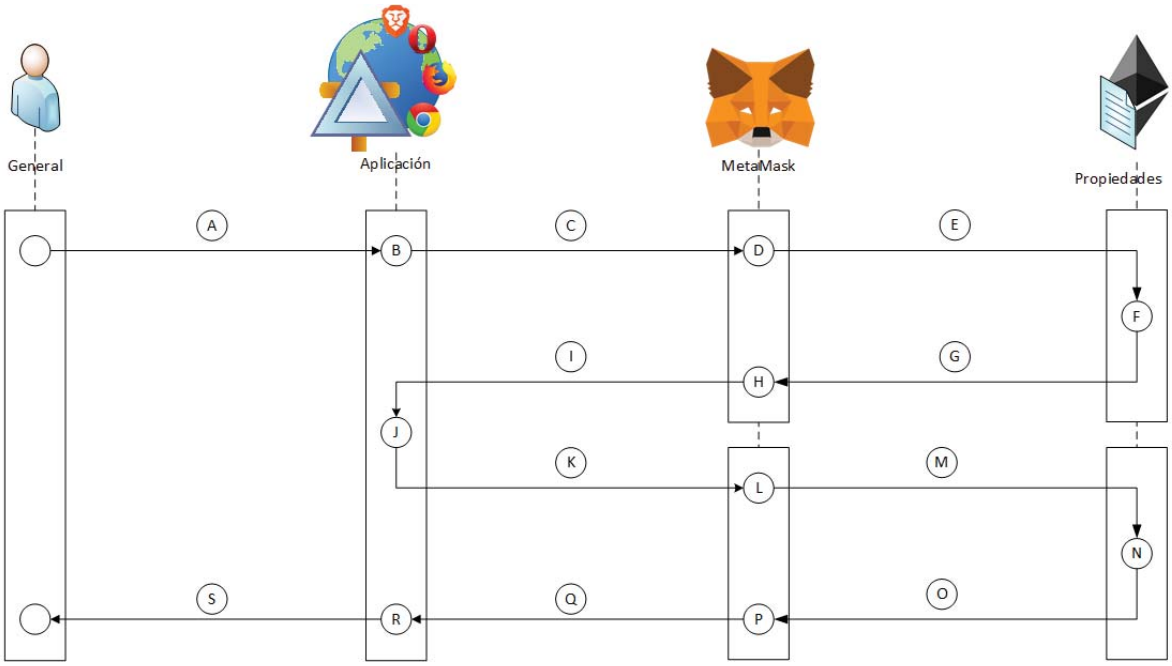

Figura B.8: Flujo de Datos Ver Propiedad

Tabla B.8: Explicación Flujo de Datos Ver Propiedad

Ver Propiedad	
Acción	**Descripción**
A	El Propietario ingresa el rol de la propiedad
B	La Aplicación recibe el rol de la propiedad y lo procesa
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna los datos de la propiedad
H	MetaMask recibe los datos
I	MetaMask retorna los datos a la Aplicación
J	La Aplicación recibe los datos de MetaMask y los guarda
K	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
L	MetaMask recibe la petición y prepara la transacción
M	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
N	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
O	La red Ethereum retorna los datos del propietario de la propiedad
P	MetaMask recibe los datos
Q	MetaMask retorna los datos a la Aplicación
R	La Aplicación recibe los datos de MetaMask y procesa la información obtenida en este paso y en J
S	La Aplicación muestra al Usuario la información de la propiedad

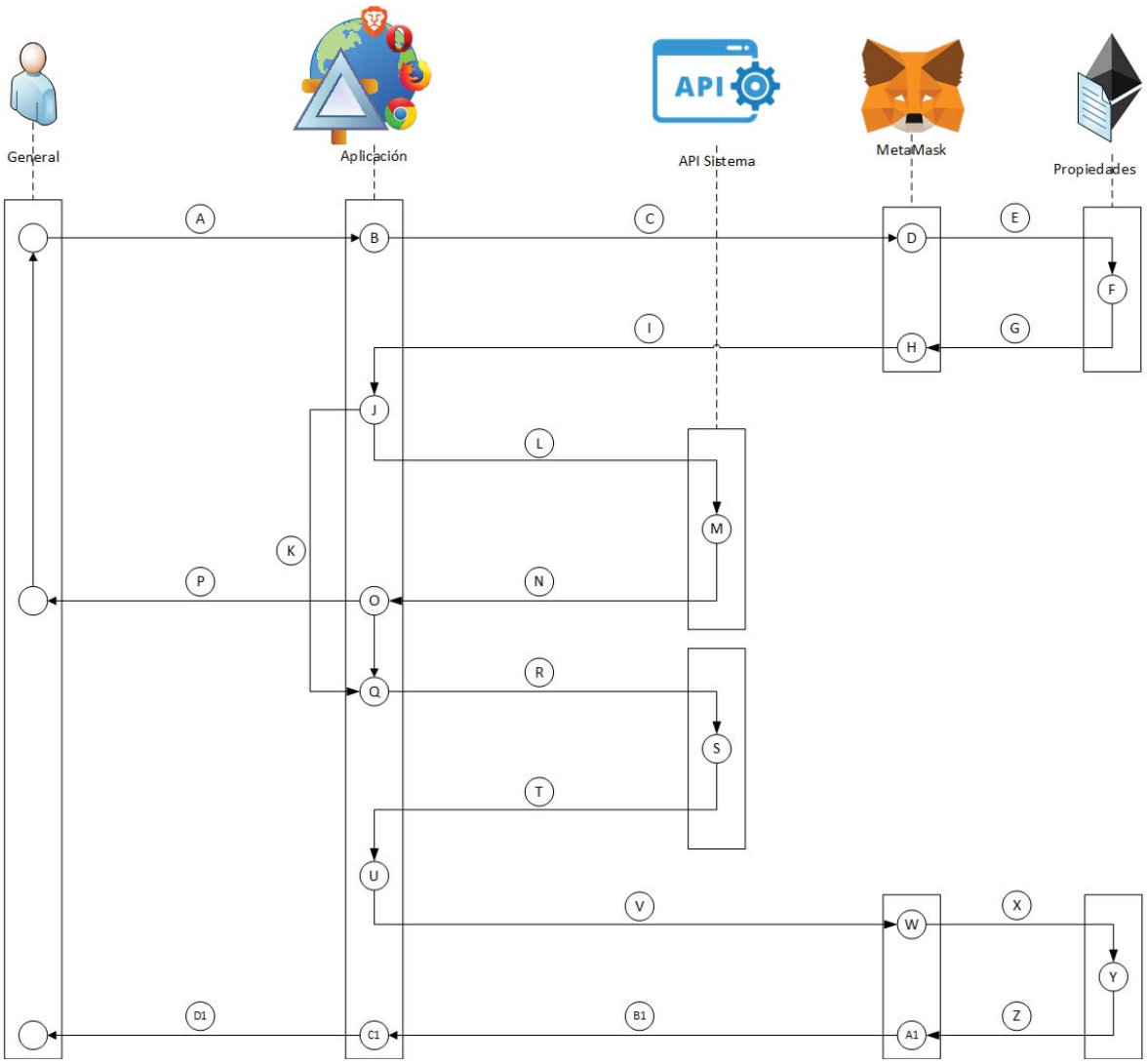

Figura B.9: Flujo de Datos Ver Información Venta

Tabla B.9: Explicación Flujo de Datos Ver Información Venta

Ver Información de la Venta	
Acción	**Descripción**
A	El Usuario ingresa el número de documento de venta
B	La Aplicación recibe el número de documento de venta y lo procesa
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código para ver la relación con el contrato
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna la relación que tiene el Usuario con el contrato de venta
H	MetaMask recibe los datos
I	MetaMask retorna los datos a la Aplicación
J	La Aplicación recibe los datos de MetaMask y los procesa
K	Si el Usuario es vendedor o comprador en el contrato la Aplicación saltará al paso Q
L	La Aplicación realiza una petición POST a la API del sistema para pedir los permisos de acceso
M	La API del sistema recibe la petición y la procesa
N	Si el Usuario no tiene los permisos retornara un mensaje de error en formato JSON
	Si el Usuario tiene los permisos retornara las claves de acceso encriptadas en formato JSON
O	La Aplicación recibe la respuesta de la API del sistema
P	Si la respuesta de la API del sistema es un error se mostrará un mensaje informando al Usuario
Q	Si la respuesta de la API del sistema son las claves de acceso encriptadas la Aplicación se prepara una nueva comunicación
R	La Aplicación realiza una petición POST a la API del sistema para pedir la clave de la plataforma
S	La API del sistema recibe la petición y la procesa
T	La API del sistema retorna la clave encriptada en formato JSON
U	La Aplicación recibe la clave y la desencripta junto con los accesos obtenidos en O
V	La Aplicación pide a MetaMask que realice una transacción de ejecución de código para ver información de la venta
W	MetaMask recibe la petición y prepara la transacción
X	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
Y	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
Z	La red Ethereum retorna la información del contrato de venta
A1	MetaMask recibe los datos
B1	MetaMask retorna los datos a la Aplicación
C1	La Aplicación recibe los datos de MetaMask y los procesa
D1	La Aplicación muestra la información de venta al Usuario

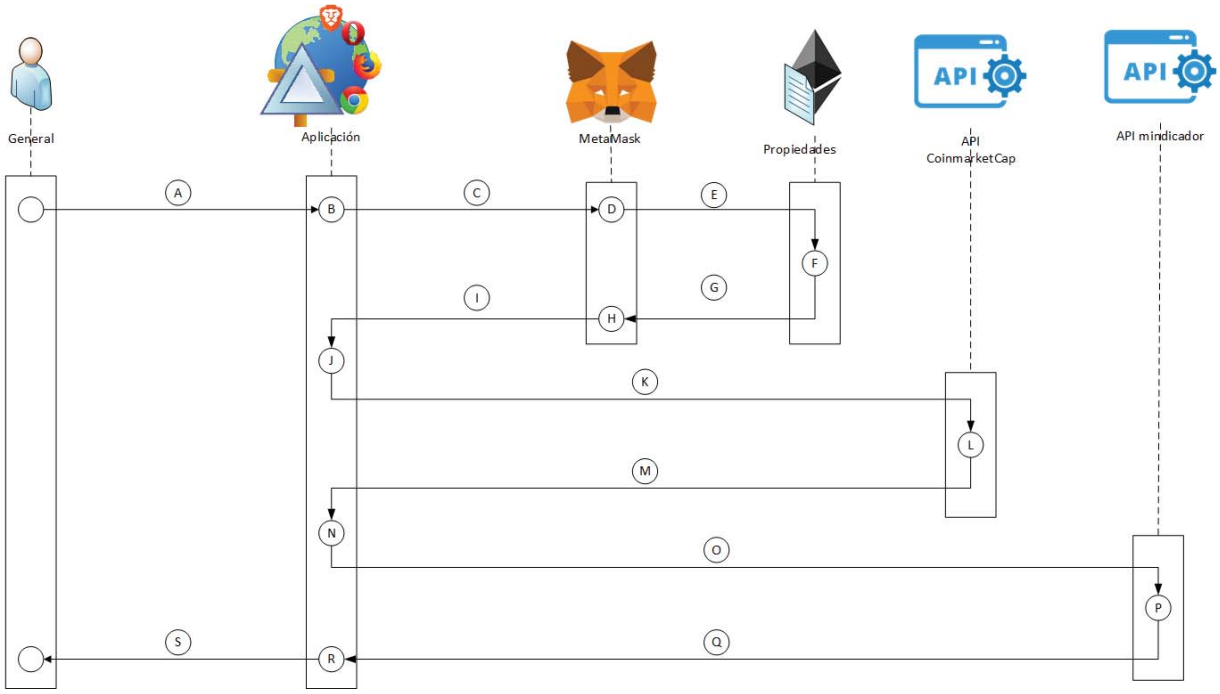

Figura B.10: Flujo de Datos Ver Precio Trámite

Tabla B.10: Explicación Flujo de Datos Ver Precio Trámite

Ver Precio del Trámite	
Acción	**Descripción**
A	El Usuario pide ver el precio del trámite
B	La Aplicación recibe la petición
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código para pedir el precio del trámite
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna el precio del trámite
H	MetaMask recibe el precio del trámite
I	MetaMask retorna el precio del trámite a la Aplicación
J	La Aplicación recibe el precio del trámite de MetaMask y lo procesa
K	La Aplicación realiza una petición GET a la API de CoinMarketCap pidiendo el precio del Ether
L	La API de CoinMarketCap recibe la petición y la procesa
M	La API de CoinMarketCap retorna el precio del Ether en dólares en formato JSON
N	La Aplicación recibe el precio del Ether en dólares y lo guarda
O	La Aplicación realiza una petición GET a la API de mindicador pidiendo el valor del dólar en pesos chilenos
P	La API de mindicador recibe la petición y la procesa
Q	La API de mindicador retorna el valor en pesos chilenos del dólar en formato JSON
R	La Aplicación recibe el precio del dólar en pesos chilenos y procesa la información recibida en J, en N y en este paso
S	La Aplicación muestra al Usuario el precio del trámite en Ether, en pesos chilenos y en dólares

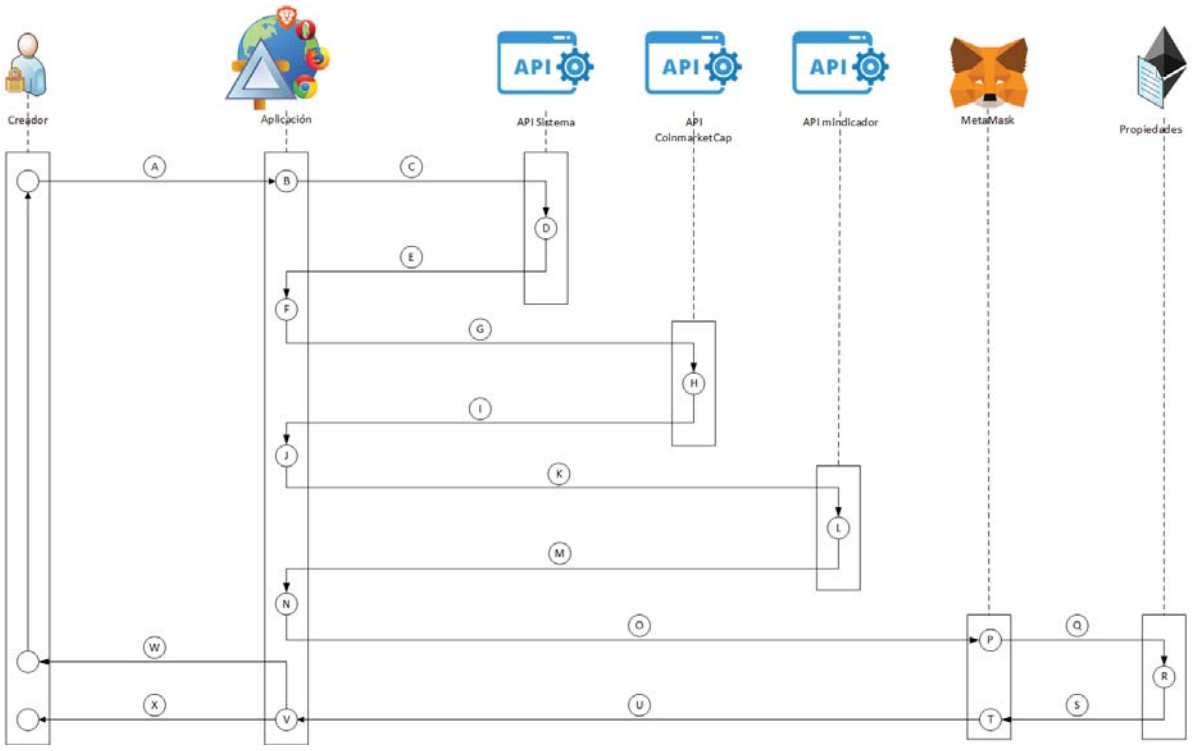

Figura B.11: Flujo de Datos Actualizar Precio Trámite

Tabla B.11: Explicación Flujo de Datos Actualizar Precio Trámite

Actualizar Precio del Trámite	
Acción	**Descripción**
A	El Usuario pide actualizar el precio del trámite
B	La Aplicación recibe la petición
C	La Aplicación realiza una petición POST a la API del sistema pidiendo la clave
D	La API del sistema recibe la petición y la procesa
E	La API del sistema retorna la clave encriptada en formato JSON
F	La Aplicación recibe la clave y la desencripta
G	La Aplicación realiza una petición GET a la API de CoinMarketCap pidiendo el precio del Ether
H	La API de CoinMarketCap recibe la petición y la procesa
I	La API de CoinMarketCap retorna el precio del Ether en dólares en formato JSON
J	La Aplicación recibe el precio del Ether en dólares y lo guarda
K	La Aplicación realiza una petición GET a la API de mindicador pidiendo el valor del dólar en pesos chilenos
L	La API de mindicador recibe la petición y la procesa
M	La API de mindicador retorna el valor en pesos chilenos del dólar en formato JSON
N	La Aplicación recibe el precio del dólar en pesos chilenos y procesa la información recibida en F, en J y en este paso
O	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
P	MetaMask recibe la petición y prepara la transacción
Q	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
R	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
S	La red Ethereum retorna un mensaje de éxito o error según corresponda
T	MetaMask recibe el mensaje
U	MetaMask retorna el mensaje a la Aplicación
V	La Aplicación recibe el mensaje de MetaMask y lo procesa
W	Si la respuesta es un error la Aplicación muestra al Usuario el mensaje correspondiente
X	Si la respuesta es exitosa la Aplicación muestra al Usuario un mensaje de éxito

B.5: Flujo de Datos Propietario

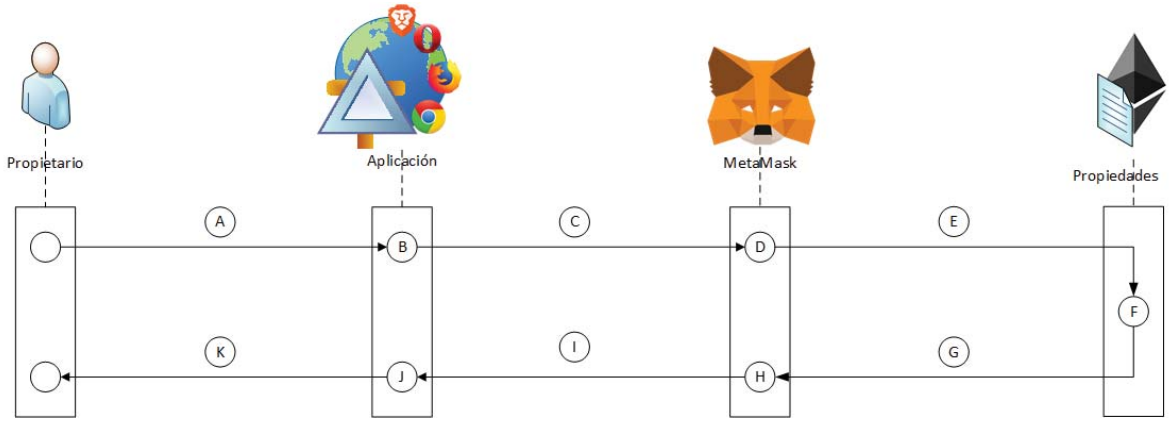

Figura B.12: Flujo de Datos Poner Propiedad a la Venta

Tabla B.12: Explicación Flujo de Datos Poner Propiedad a la Venta

Poner Propiedad a la Venta	
Acción	**Descripción**
A	El Propietario ingresa su el rol de la propiedad
B	La Aplicación recibe el rol de la propiedad
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Propietario el resultado de la operación (éxito o error)

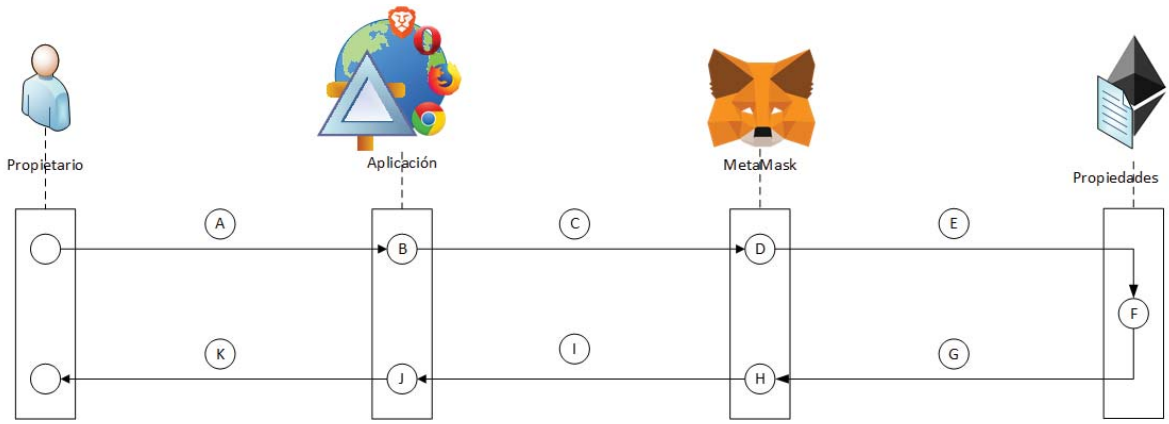

Figura B.13: Flujo de Datos Cancelar Venta Propiedad

Tabla B.13: Explicación Flujo de Datos Cancelar Venta Propiedad

Cancelar Venta de Propiedad	
Acción	**Descripción**
A	El Propietario ingresa su el rol de la propiedad
B	La Aplicación recibe el rol de la propiedad
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Propietario el resultado de la operación (éxito o error)

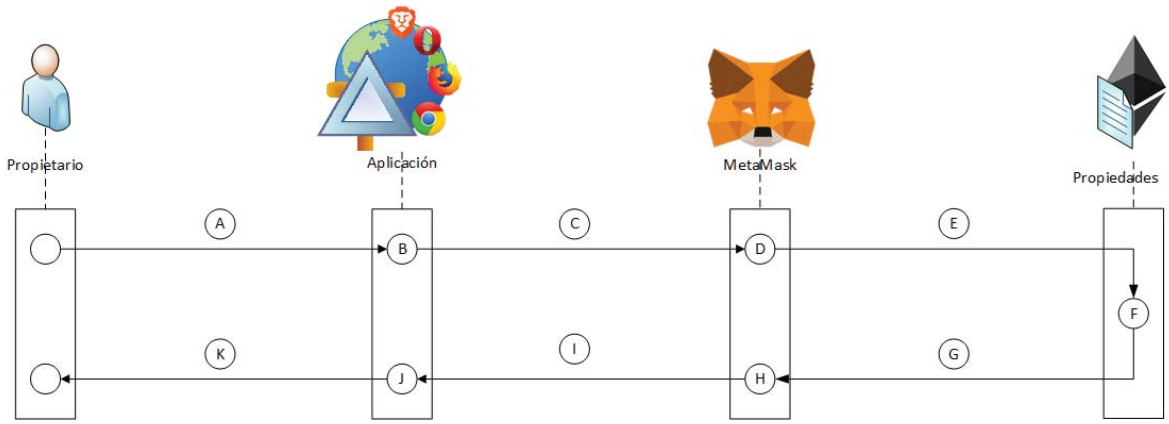

Figura B.14: Flujo de Datos Agregar Descripción Venta

Tabla B.14: Explicación Flujo de Datos Agregar Descripción Venta

Agregar Contrato de Compraventa	
Acción	**Descripción**
A	El Propietario ingresa el número de documento de venta y la descripción del contrato
B	La Aplicación recibe los datos y los procesa
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Propietario el resultado de la operación (éxito o error)

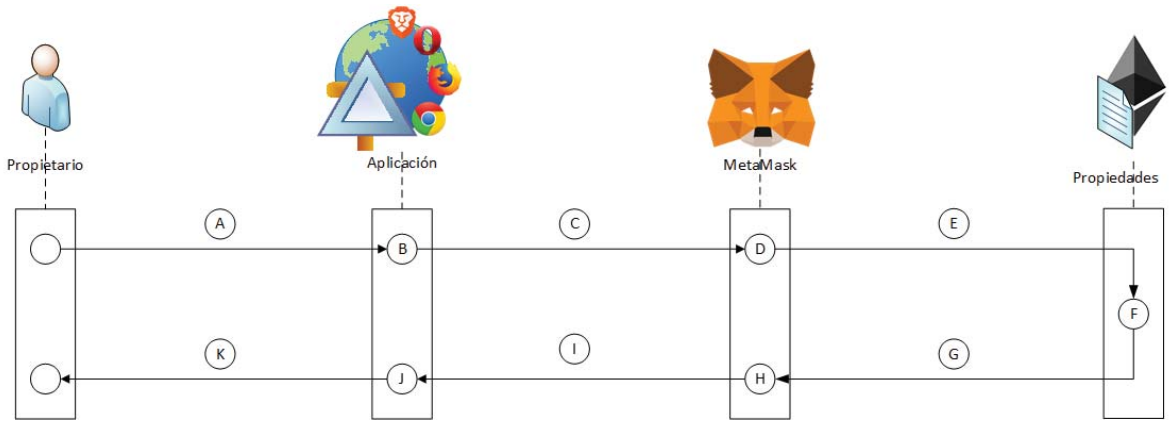

Figura B.15: Flujo de Datos Dar Permiso Firmar Venta

Tabla B.15: Explicación Flujo de Datos Dar Permiso Firmar Venta

Dar Permiso para Firmar Venta	
Acción	**Descripción**
A	El Propietario ingresa el número de documento de venta y el RUT de la persona a dar permiso
B	La Aplicación recibe los datos y los procesa
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Propietario el resultado de la operación (éxito o error)

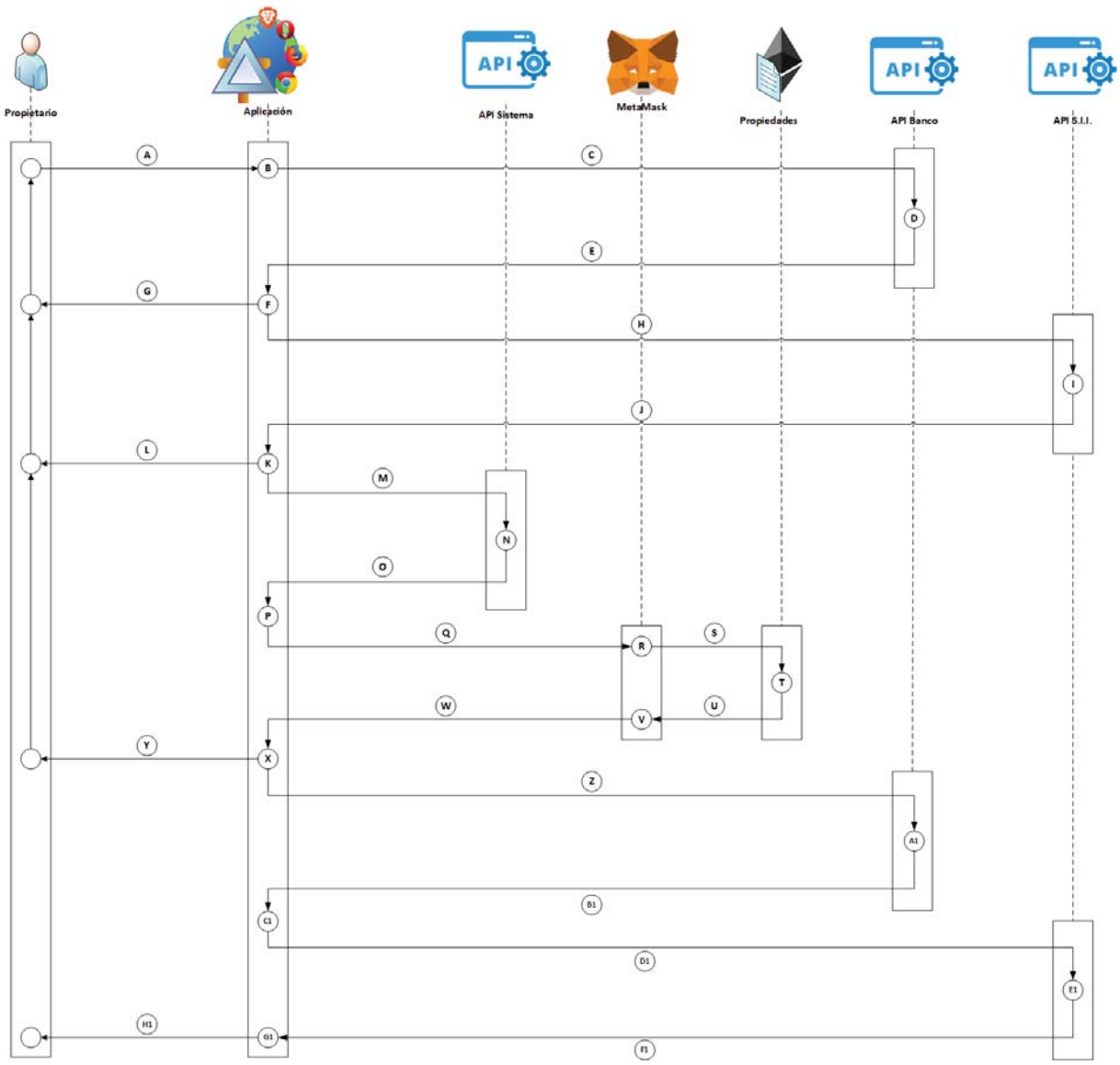

Figura B.16: Flujo de Datos Finalizar Venta

Tabla B.16: Explicación Flujo de Datos Finalizar Venta

Finalizar Venta	
Acción	**Descripción**
A	El Propietario ingresa el rol de la propiedad, el número de documento de venta, el número de custodia bancaria, el RUT del comprador y el RUT del vendedor
B	La Aplicación recibe los datos y los procesa
C	La Aplicación realiza una petición POST a la API del Banco para probar la disponibilidad de la misma
D	La API del Banco recibe la petición
E	La API del Banco retorna un mensaje de éxito en formato JSON
F	La Aplicación recibe la respuesta de la API del Banco
G	Si no hay conexión la Aplicación retornará un mensaje de error al Propietario
H	Si hay conexión con el Banco la Aplicación realiza una petición POST a la API del S.I.I. para probar disponibilidad
I	La API del S.I.I. recibe la petición
J	La API del S.I.I. retorna un mensaje de éxito en formato JSON
K	La Aplicación recibe la respuesta de la API del S.I.I.
L	Si no hay conexión la Aplicación retornará un mensaje de error al Propietario
M	Si hay conexión con el S.I.I. la Aplicación realiza una petición POST a la API del sistema pidiendo la clave
N	La API del sistema recibe la petición y la procesa
O	La API del sistema retorna la clave encriptada en formato JSON
P	La Aplicación recibe la clave, la desencripta y prepara los datos para comunicarse con MetaMask
Q	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
R	MetaMask recibe la petición y prepara la transacción
S	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
T	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
U	La red Ethereum retorna un mensaje de éxito o error según corresponda
V	MetaMask recibe el mensaje
W	MetaMask retorna el mensaje a la Aplicación
X	La Aplicación recibe el mensaje de MetaMask y lo procesa
Y	Si la respuesta es un error muestra al Propietario el mensaje correspondiente
Z	Si la respuesta es exitosa realiza una petición POST a la API del Banco para que liberé el pago
A1	La API del Banco recibe la petición y libera el pago
B1	La API del Banco retorna un mensaje de éxito en formato JSON
C1	La Aplicación recibe la respuesta
D1	La Aplicación realiza una petición POST a la API del S.I.I. para comunicar cambio de propietario
E1	La API del S.I.I. recibe la petición y la procesa
F1	La API del S.I.I. retorna un mensaje de éxito en formato JSON
G1	La Aplicación recibe la respuesta
H1	La Aplicación muestra al Propietario el resultado de la operación (éxito o error)

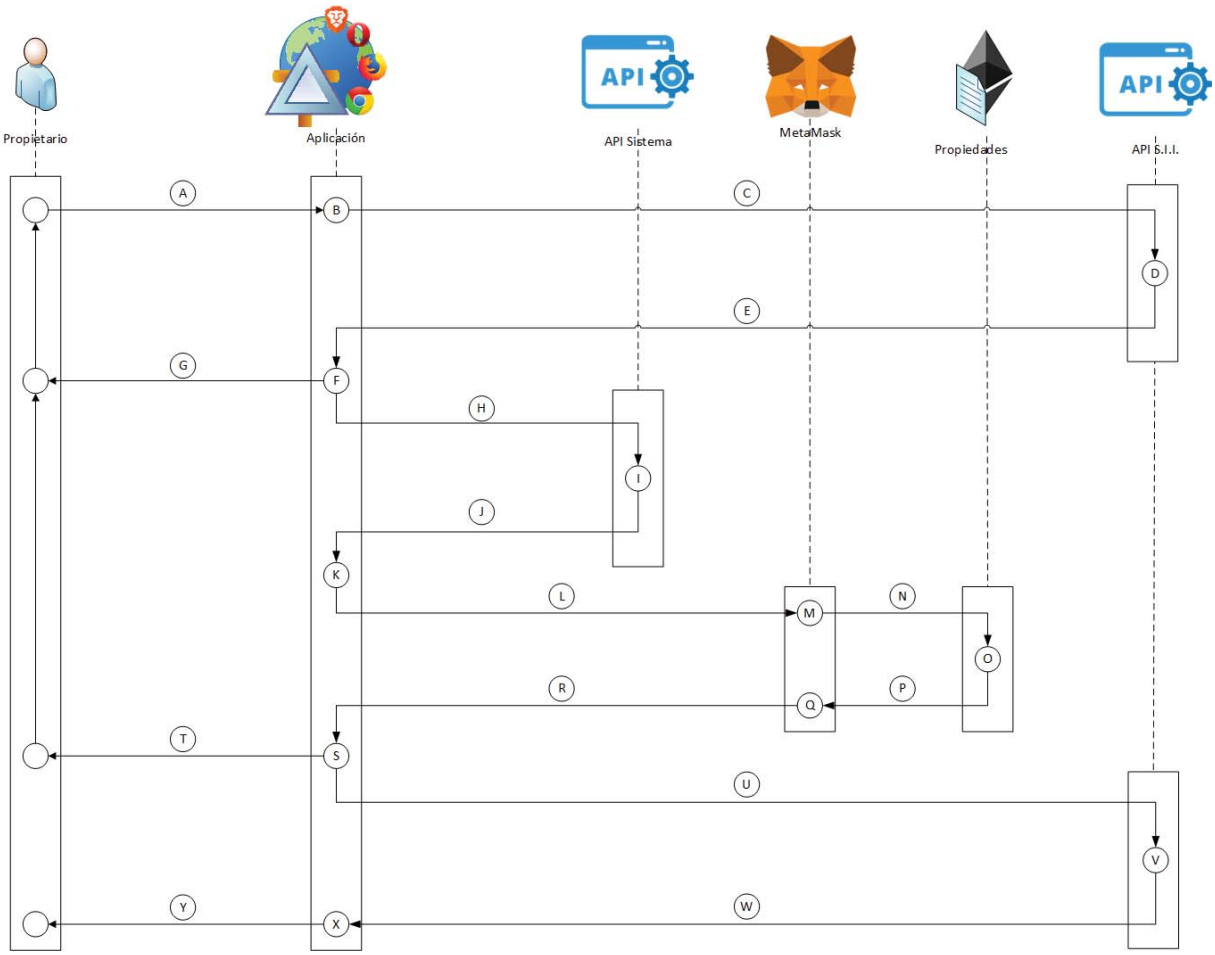

Figura B.17: Flujo de Datos Transferir Propiedad

Tabla B.17: Explicación Flujo de Datos Transferir Propiedad

Transferir Propiedad	
Acción	**Descripción**
A	El Propietario ingresa el rol de la propiedad, el RUT de la persona y el monto a pagar
B	La Aplicación recibe los datos y los procesa
C	La Aplicación realiza una petición POST a la API del S.I.I. para probar disponibilidad
D	La API del S.I.I. recibe la petición
E	La API del S.I.I. retorna un mensaje de éxito en formato JSON
F	La Aplicación recibe la respuesta de la API del S.I.I.
G	Si no hay conexión la Aplicación retornará un mensaje de error al Propietario
H	Si hay conexión con el S.I.I. la Aplicación realiza una petición POST a la API del sistema pidiendo la clave
I	La API del sistema recibe la petición y la procesa
J	La API del sistema retorna la clave encriptada en formato JSON
K	La Aplicación recibe la clave, la desencripta y prepara los datos para comunicarse con MetaMask
L	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
M	MetaMask recibe la petición y prepara la transacción
N	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
O	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
P	La red Ethereum retorna un mensaje de éxito o error según corresponda
Q	MetaMask recibe el mensaje
R	MetaMask retorna el mensaje a la Aplicación
S	La Aplicación recibe el mensaje de MetaMask y lo procesa
T	Si la respuesta es un error muestra al Propietario el mensaje correspondiente
U	Si la respuesta es exitosa la Aplicación realiza una petición POST a la API del S.I.I. para comunicar cambio de propietario
V	La API del S.I.I. recibe la petición y la procesa
W	La API del S.I.I. retorna un mensaje de éxito en formato JSON
X	La Aplicación recibe la respuesta
Y	La Aplicación muestra al Propietario el resultado de la operación (éxito o error)

B.6: Flujo de Datos Comprador

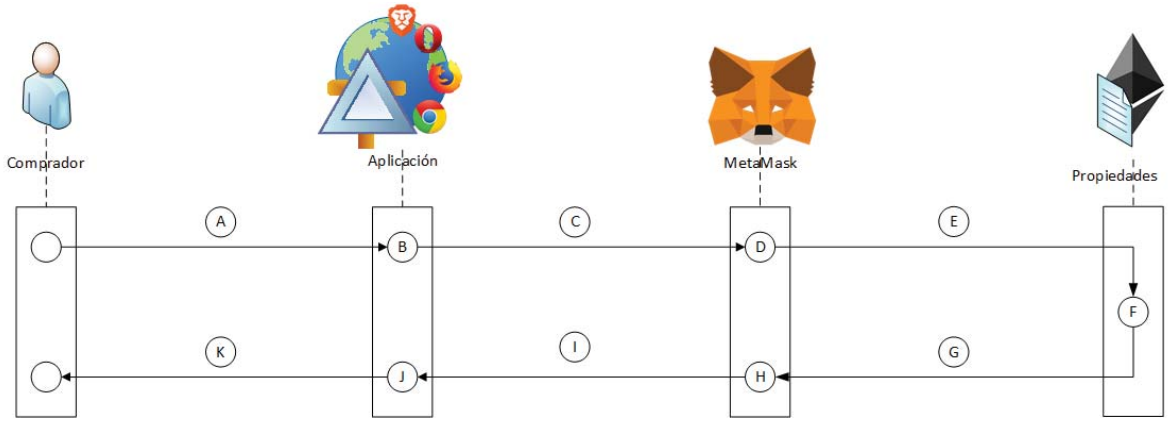

Figura B.18: Flujo de Datos Firmar Venta Propiedad

Tabla B.18: Explicación Flujo de Datos Firmar Venta Propiedad

Firmar Venta de Propiedad	
Acción	**Descripción**
A	El Comprador ingresa el número de documento de venta
B	La Aplicación recibe el número de documento de venta
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna un mensaje de éxito o error según corresponda
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	La Aplicación muestra al Comprador el resultado de la operación (éxito o error)

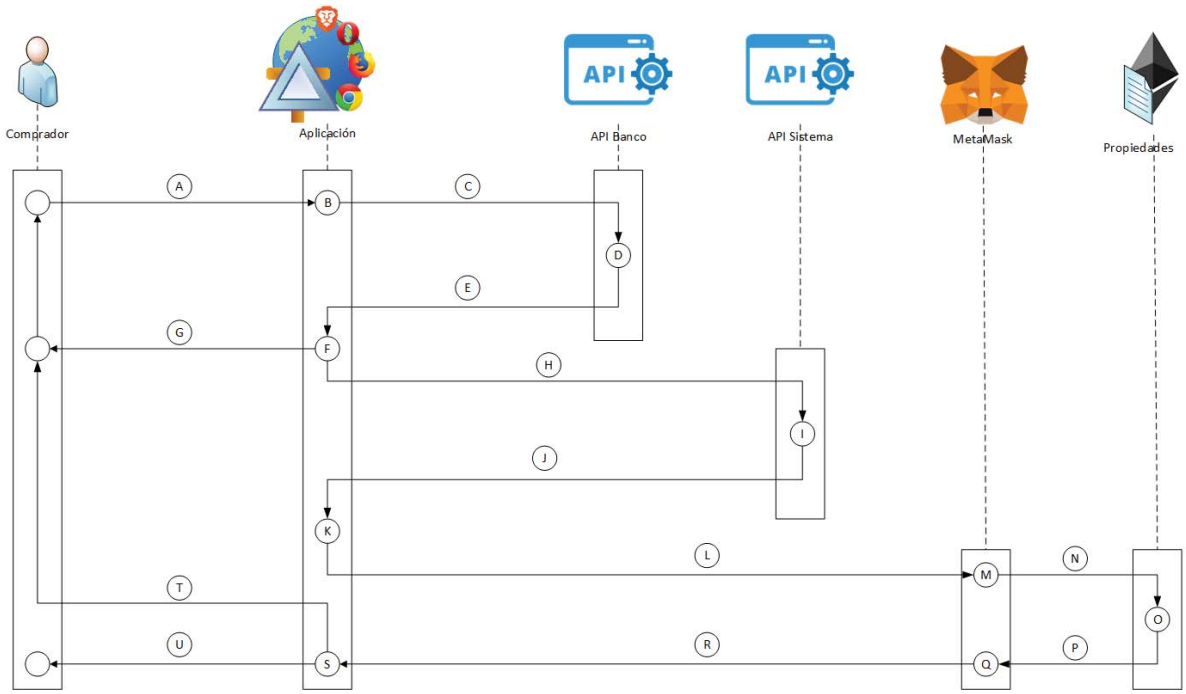

Figura B.19: Flujo de Datos Agregar Custodia Bancaria

Tabla B.19: Explicación Flujo de Datos Agregar Custodia Bancaria

Agregar Documento de Custodia Bancaria	
Acción	**Descripción**
A	El Comprador ingresa el número de documento de venta, el número de documento de custodia, el RUT del comprador, el RUT del vendedor y el monto de la operación
B	La Aplicación recibe los datos y los procesa
C	La Aplicación realiza una petición POST a la API del Banco para validar la información
D	La API del Banco recibe la petición y la procesa
E	La API del Banco retorna si la información recibida es correcta o no en formato JSON
F	La Aplicación recibe la respuesta de la API del Banco
G	Si hubo algún error o la información es incorrecta la Aplicación retornará un mensaje de error al Comprador
H	Si la información es correcta la Aplicación realiza una petición POST a la API del sistema pidiendo la clave
I	La API del sistema recibe la petición y la procesa
J	La API del sistema retorna la clave encriptada en formato JSON
K	La Aplicación recibe la clave, la desencripta y prepara los datos para comunicarse con MetaMask
L	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
M	MetaMask recibe la petición y prepara la transacción
N	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
O	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
P	La red Ethereum retorna un mensaje de éxito o error según corresponda
Q	MetaMask recibe el mensaje
R	MetaMask retorna el mensaje a la Aplicación
S	La Aplicación recibe el mensaje de MetaMask y lo procesa
T	Si la respuesta es un error la Aplicación muestra al Comprador el mensaje correspondiente
U	Si la respuesta es exitosa la Aplicación muestra al Comprador un mensaje de éxito

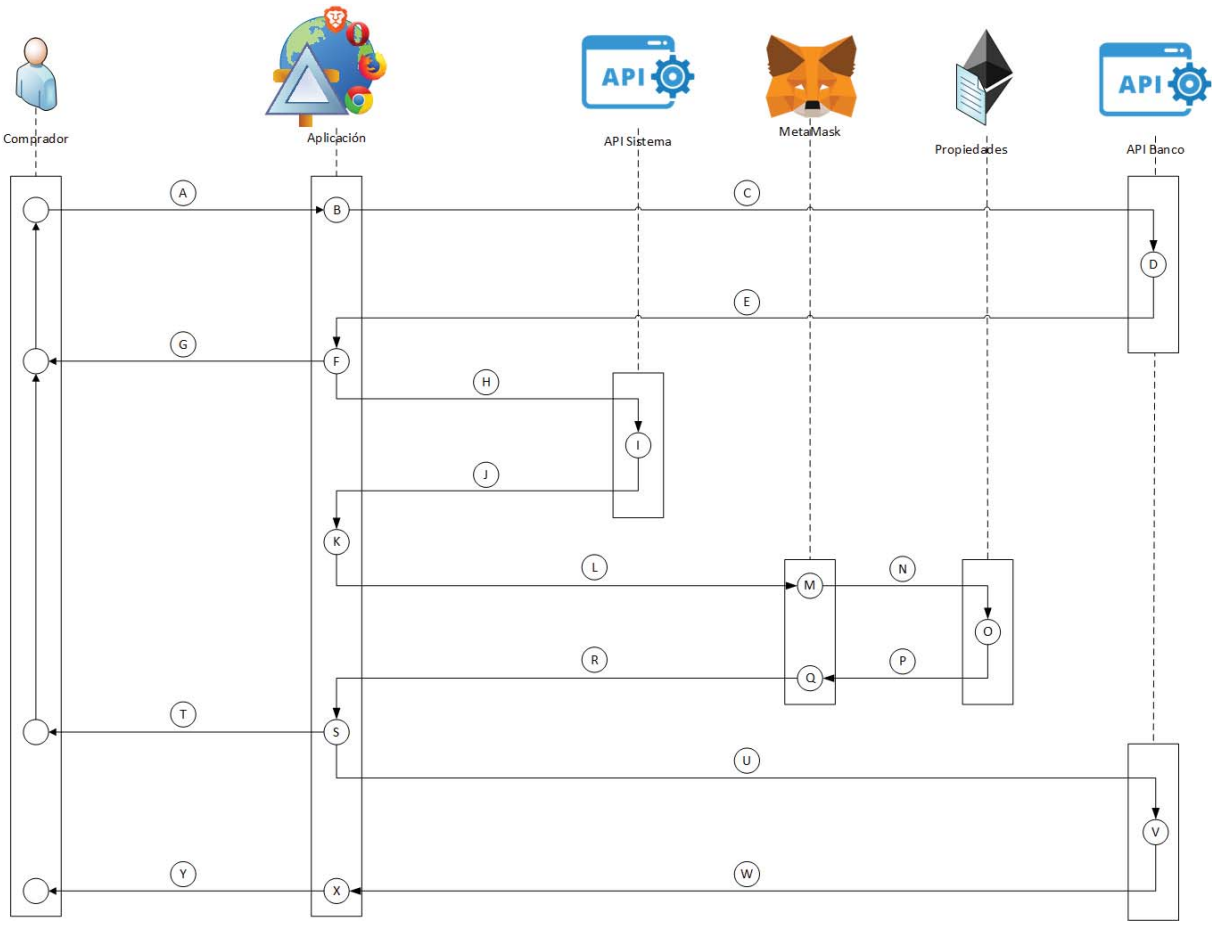

Figura B.20: Flujo de Datos Solicitar Devolución Custodia

Tabla B.20: Explicación Flujo de Datos Solicitar Devolución Custodia

Devolución de Custodia	
Acción	**Descripción**
A	El Comprador ingresa el número de documento de custodia, el RUT del comprador y el RUT del vendedor
B	La Aplicación recibe los datos y los procesa
C	La Aplicación realiza una petición POST a la API del Banco para ver si se puede realizar una devolución de custodia
D	La API del Banco recibe la petición
E	La API del Banco retorna un mensaje de éxito en formato JSON
F	La Aplicación recibe la respuesta de la API del Banco
G	Si no hay conexión o no se puede realizar la devolución la Aplicación retornará un mensaje de error al Comprador
H	Si se puede realizar la devolución en el Banco la Aplicación realiza una petición POST a la API del sistema pidiendo la clave
I	La API del sistema recibe la petición y la procesa
J	La API del sistema retorna la clave encriptada en formato JSON
K	La Aplicación recibe la clave, la desencripta y prepara los datos para comunicarse con MetaMask
L	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
M	MetaMask recibe la petición y prepara la transacción
N	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
O	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
P	La red Ethereum retorna un mensaje de éxito o error según corresponda
Q	MetaMask recibe el mensaje
R	MetaMask retorna el mensaje a la Aplicación
S	La Aplicación recibe el mensaje de MetaMask y lo procesa
T	Si la respuesta es un error muestra al Comprador el mensaje correspondiente
U	Si la respuesta es exitosa la Aplicación realiza una petición POST a la API del banco para pedir la devolución de la custodia
V	La API del Banco recibe la petición y la procesa
W	La API del Banco retorna un mensaje de éxito en formato JSON
X	La Aplicación recibe la respuesta
Y	La Aplicación muestra al Comprador el resultado de la operación

B.7: Flujo de Datos Propietario y Comprador

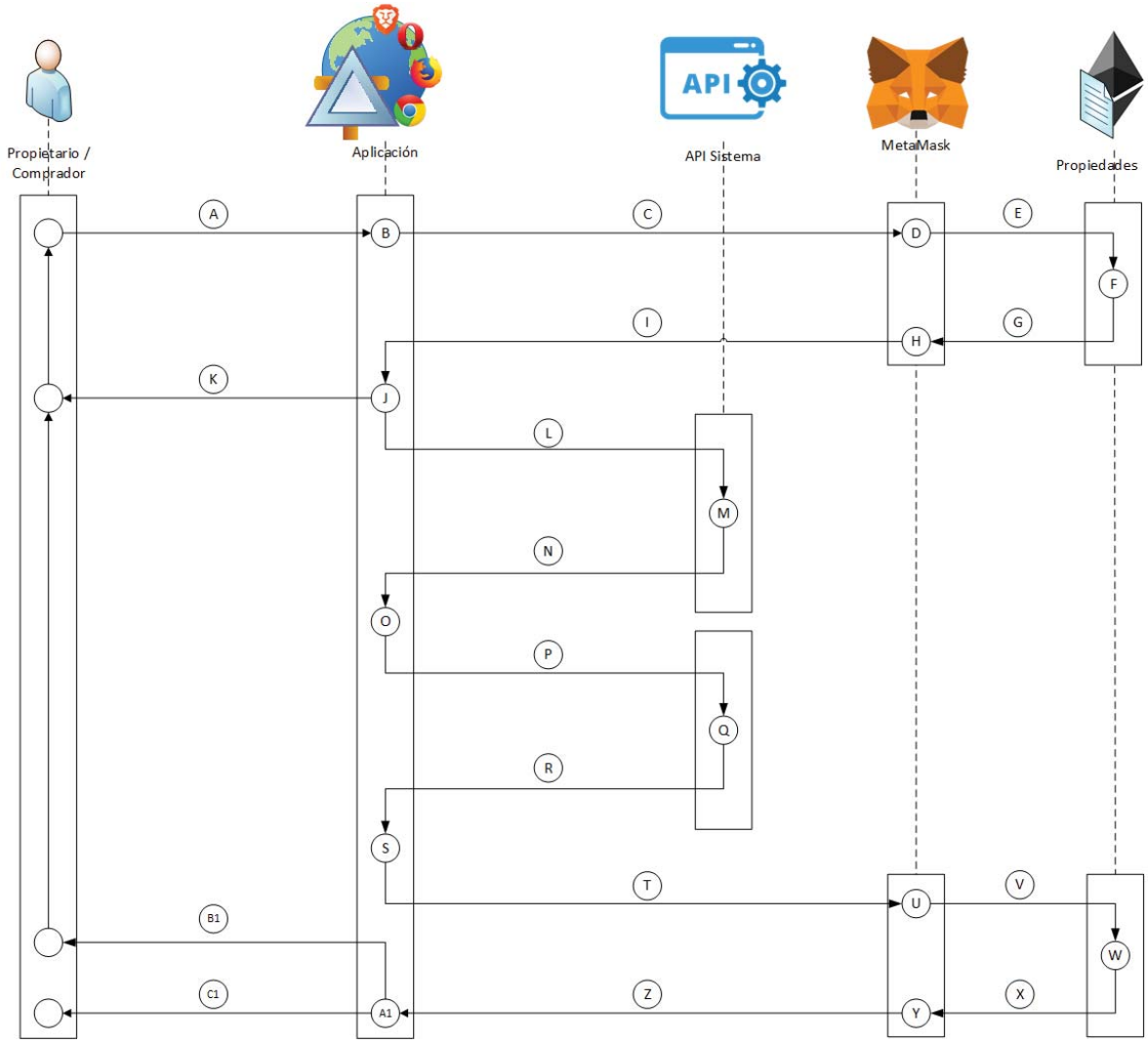

Figura B.21: Flujo de Datos Dar Permiso Ver Venta

Tabla B.21: Explicación Flujo de Datos Dar Permiso Ver Venta

Dar Permiso para Ver Venta	
Acción	**Descripción**
A	El Propietario o el Comprador ingresa el número de documento de venta y el RUT de la persona a dar permiso
B	La Aplicación recibe los datos y los procesa
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código para validar identidad
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna si el usuario es vendedor o comprador, si no es ninguno retorna error
H	MetaMask recibe el mensaje
I	MetaMask retorna el mensaje a la Aplicación
J	La Aplicación recibe el mensaje de MetaMask y lo procesa
K	Si el usuario no es ni comprador ni vendedor la Aplicación retorna un mensaje correspondiente
L	Si el usuario es Vendedor (Propietario) la Aplicación realizará una petición POST a la API del sistema para generar acceso de vendedor
	Si el usuario es Comprador la Aplicación realizará una petición POST a la API del sistema para generar acceso de comprador
M	La API del sistema recibe la petición y genera el acceso al RUT
N	La API del sistema retorna el acceso encriptado en formato JSON
O	La Aplicación recibe el acceso y lo desencripta
P	La Aplicación realiza una petición POST a la API del sistema para pedir la clave
Q	La API del sistema recibe la petición y la procesa
R	La API del sistema retorna la clave encriptada en formato JSON
S	La Aplicación recibe la clave, la desencripta y prepara los datos para comunicarse con MetaMask
T	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
U	MetaMask recibe la petición y prepara la transacción
V	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
W	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
X	La red Ethereum retorna un mensaje de éxito o error según corresponda
Y	MetaMask recibe el mensaje
Z	MetaMask retorna el mensaje a la Aplicación
A1	La Aplicación recibe el mensaje de MetaMask y lo procesa
B1	Si la operación no se pudo realizar la Aplicación muestra al Usuario un mensaje de error
C1	Si la operación fue exitosa la Aplicación muestra al usuario un mensaje de éxito

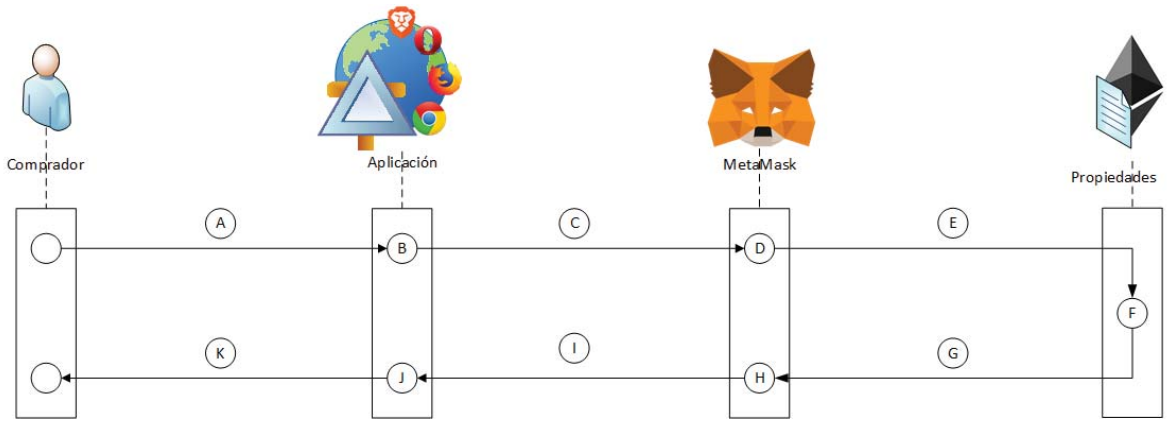

Figura B.22: Flujo de Datos Ver Estado Venta

Tabla B.22: Explicación Flujo de Datos Ver Estado Venta

Ver Estado de la Venta	
Acción	**Descripción**
A	El Usuario ingresa el número de documento de venta
B	La Aplicación recibe el número de documento de venta
C	La Aplicación pide a MetaMask que realice una transacción de ejecución de código con los datos recibidos
D	MetaMask recibe la petición y prepara la transacción
E	Metamask realiza la transacción a la red Ethereum para ejecutar una función en el contrato inteligente
F	La red Ethereum procesa la transación y ejecuta el código del contrato inteligente
G	La red Ethereum retorna la información pedida o error según corresponda
H	MetaMask recibe la información
I	MetaMask retorna la información a la Aplicación
J	La Aplicación recibe la información de MetaMask y la procesa
K	La Aplicación muestra al Usuario el estado actual de la venta, si hubo un error se muestra un mensaje de error

C: Código de las Funciones del Sistema

C.1: Registro de Usuario

```
/*
Descripción:
    - Agrega un usuario al sistema con todos los valores correspondientes.
Parámetros de Entrada:
    - string     _clave
    - string     _nombres
    - string     _apell_pat
    - string     _apell_mat
    - uint       _rut
    - string     _digito
    - string     _celular
Parámetros de Salida:
*/
function agregarUsuario(string _clave, string _nombres, string _apell_pat, string _apell_mat, uint _rut, string _digito, string _celular) public
soloClave(_clave)
soloUsuarioNoRegistrado
soloRutNoRegistrado(_rut) {
    Usuario memory user;
    user.direccion = msg.sender;
    user.nombres = _nombres;
    user.apellido_paterno = _apell_pat;
    user.apellido_materno = _apell_mat;
    user.rut = _rut;
    user.digitoVerificador = _digito;
    user.celular = _celular;

    cantidadUsuarios += 1;

    usuarios[_rut] = user;
    addRut[msg.sender] = _rut;
    userRegistrado[msg.sender] = true;
    rutRegistrado[_rut] = true;
}
```

Figura C.1: Registro de Usuario

C.2: Funciones Creador

```
/*
Descripción:
    - Ingresa el RUT del tasador en la variable "_rut" y le otorga los permisos correspondientes.
Parámetros de Entrada:
    - uint      _rut
Parámetros de Salida:
*/
function agregarTasador(uint _rut) public
soloUsuarioRegistrado
soloRutRegistrado(_rut)
soloCreador {
    tasadores[usuarios[_rut].direccion] = true;
}
```

Figura C.2: Agregar Tasador

```
/*
Descripción:
    - Cambia la clave antigua asignadole el valor de "_clave_nueva".
Parámetros de Entrada:
    - string    _clave
    - string    _claveNueva
Parámetros de Salida:
*/
function cambiarClaveSistema(string _clave, string _claveNueva) public
soloUsuarioRegistrado
soloCreador
soloClave(_clave) {
    claveSistema = _claveNueva;
}
```

Figura C.3: Cambiar Clave Sistema

```
/*
Descripción:
    - Retorna el valor del balance del sistema.
Parámetros de Entrada:
Parámetros de Salida:
    - unit      balance
*/
function verBalance() public view
soloUsuarioRegistrado
soloCreador
returns(uint) { //balance
    return balance;
}
```

Figura C.4: Ver Balance

```
/*
Descripción:
    - Transfiere todo el balance del sistema al creador del contrato.
Parámetros de Entrada:
Parámetros de Salida:
*/
function retirarBalance() public
soloUsuarioRegistrado
soloCreador
soloBalancePositivo {
    msg.sender.transfer(balance);
    balance = 0;
}
```

Figura C.5: Retirar Balance

C.3: Funciones Tasador

```
/*
Descripción:
    - Cambia el valor de la propiedad con rol "_rolProp" a "_valorPesos".
Parámetros de Entrada:
    - string    _rolProp
    - uint      _valorPesos
Parámetros de Salida:
*/
function cambiarValorPropiedad(string _rolProp, uint _valorPesos) public
soloUsuarioRegistrado
soloTasadores
soloRolPropiedadValido(_rolProp) {
    propiedades[_rolProp].valor = _valorPesos;
}
```

Figura C.6: Cambiar Valor Propiedad

C.4: Funciones Usuario General

```
/*
Descripción:
    - Agrega una propiedad con todos los valores correspondientes.
Parámetros de Entrada:
    - string    _clave
    - string    _rolProp
    - string    _tipo
    - uint      _valorPesos
    - string    _direccion
    - string    _comuna
    - string    _region
Parámetros de Salida:
*/
function agregarPropiedad(string _clave, string _rolProp, string _tipo, uint _valorPesos, string _direccion, string _comuna, string _region) public
soloClave(_clave)
soloUsuarioRegistrado
soloPropiedadNoExistente(_rolProp) { //  idPropiedad
    Propiedad memory prop;
    prop.propietario = msg.sender;
    prop.valor = _valorPesos;
    prop.direccion = _direccion;
    prop.tipo = _tipo;
    prop.comuna = _comuna;
    prop.region = _region;

    propiedades[_rolProp] = prop;

    esPropiedad[_rolProp] = true;
    cantidadPropiedades += 1;

    pertenencias[_rolProp] = msg.sender;
    cantidadPropUsuario[msg.sender] += 1;
}
```

Figura C.7: Agregar Propiedad

```
/*
Descripción:
    - Función que retorna la información de la propiedad con rol "_rolProp".
Parámetros de Entrada:
    - string    _rolProp
Parámetros de Salida:
    - uint      valor
    - string    direccion
    - string    tipo
    - string    comuna
    - string    region
    - bool      enVenta
    - uint      ventaActual
*/
function verPropiedad(string _rolProp) public view // Obtiene la informacion de la propiedad
soloUsuarioRegistrado
soloRolPropiedadValido(_rolProp)
returns(uint, string, string, string, string, bool, uint) { // valor - direccion - tipo - comuna - region - enVenta - ventaActual
    return (
        propiedades[_rolProp].valor,
        propiedades[_rolProp].direccion,
        propiedades[_rolProp].tipo,
        propiedades[_rolProp].comuna,
        propiedades[_rolProp].region,
        enVenta[_rolProp],
        propiedades[_rolProp].ventaActual
    );
}
```

Figura C.8: Ver Propiedad

```
/*
Descripción:
    Función que retorna la información del propietario de la propiedad con rol "_rolProp". El valor de "celular" se mostrará sólo cuando sea necesario.
Parámetros de Entrada:
    - string    _rolProp
Parámetros de Salida:
    - string    nombres
    - string    apellido_paterno
    - string    apellido_materno
    - uint      rut
    - string    celular
*/
function verPropietario(string _rolProp) public view // Obtiene la informacion del propietario de la propiedad
soloUsuarioRegistrado
soloRolPropiedadValido(_rolProp)
returns(string, string, string, uint, string) { // nombres - apellido_paterno - apellido_materno - rut - celular
    return (
        usuarios[addRut[propiedades[_rolProp].propietario]].nombres,
        usuarios[addRut[propiedades[_rolProp].propietario]].apellido_paterno,
        usuarios[addRut[propiedades[_rolProp].propietario]].apellido_materno,
        usuarios[addRut[propiedades[_rolProp].propietario]].rut,
        getNumeroPropietario(_rolProp)
    );
}
```

Figura C.9: Ver Propietario

```
/*
Descripción:
    - Función que retorna el contrato de compra venta. Requiere permisos.
Parámetros de Entrada:
    - string    _clave
    - uint      _idVenta
    - string    _claveAccesoVendedor
    - string    _claveAccesoComprador
Parámetros de Salida:
    - string    descripcion
*/
function verDescripcionVenta(string _clave, uint _idVenta, string _claveAccesoVendedor, string _claveAccesoComprador) public view
soloUsuarioRegistrado
soloIdVentaValido(_idVenta)
soloClave(_clave)
returns(string) { // descripcion
    if( ventas[_idVenta].vendedor == msg.sender ) {
        return ventas[_idVenta].descripcion;
    } else {
        if( ventas[_idVenta].permisoFirmar == msg.sender || ventas[_idVenta].comprador == msg.sender ) {
            return ventas[_idVenta].descripcion;
        } else {
            require( sha256(ventas[_idVenta].accesoVendedor[msg.sender]) == sha256(_claveAccesoVendedor) && sha256(ventas[_idVenta].accesoComprador[msg.sender]) == sha256(_claveAccesoComprador) );
            return ventas[_idVenta].descripcion;
        }
    }
}
```

Figura C.10: Ver Descripción Venta

```
/*
Descripción:
    - Función que retorna el precio del trámite.
Parámetros de Entrada:
Parámetros de Salida:
    - uint      precioTramite
*/
function verPrecioTramite() public view
soloUsuarioRegistrado
returns(uint) { //precioTramite
    return precioTramite;
}
```

Figura C.11: Ver Precio Trámite

```
/*
Descripción:
    - Actualiza el precio del trámite asignándole el valor "_valorWei".
Parámetros de Entrada:
    - string    _clave
    - unit      _valorWei
Parámetros de Salida:
*/
function actualizarPrecioTramite(string _clave, uint _valorWei) public
soloUsuarioRegistrado
soloClave(_clave) {
    precioTramite = _valorWei;
}
```

Figura C.12: Actualizar Precio Trámite

C.5: Funciones Propietario

```
/*
Descripción:
    - Se pondrá en venta la propiedad con rol "_rolProp".
Parámetros de Entrada:
    - string    _rolProp
Parámetros de Salida:
*/
function ponerEnVentaPropiedad(string _rolProp) public
soloUsuarioRegistrado
soloPropietario(_rolProp)
soloRolPropiedadValido(_rolProp)
soloPropiedadNoEnVenta(_rolProp) {
    enVenta[_rolProp] = true;
    uint idVta;
    idVta = agregarVenta(_rolProp,msg.sender);
    esVenta[idVta] = true;
    ventas[idVta].estado = 0; // Venta creada
    propiedades[_rolProp].ventaActual = idVta;
}
```

Figura C.13: Poner en Venta Propiedad

```
/*
Descripción:
    - Se cancelará la venta de la propiedad con rol "_rolProp".
Parámetros de Entrada:
    - string    _rolProp
Parámetros de Salida:
*/
function cancelarVentaPropiedad(string _rolProp) public
soloUsuarioRegistrado
soloPropietario(_rolProp)
soloRolPropiedadValido(_rolProp)
soloPropiedadEnVenta(_rolProp) {
    uint _idVenta = propiedades[_rolProp].ventaActual;
    if( ventas[_idVenta].estado < 2 ) {
        delete ventas[_idVenta];
        delete esVenta[_idVenta];
        cantidadVentas -= 1;
        enVenta[_rolProp] = false;
        delete propiedades[_rolProp].ventaActual;
    }
}
```

Figura C.14: Cancelar Venta Propiedad

```
/*
Descripción:
    - Se agregará la descripción "_descripcion" de la venta con id "_idVenta".
Parámetros de Entrada:
    - uint      _idVenta
    - string    _descripcion
Parámetros de Salida:
*/
function agregarDescripcionVenta(uint _idVenta, string _descripcion) public
soloUsuarioRegistrado
soloPropietarioVenta(_idVenta)
soloVentaCreada(_idVenta) {
    ventas[_idVenta].descripcion = _descripcion;
    ventas[_idVenta].estado = 1; //Descripción agregada
}
```

Figura C.15: Agregar Descripción Venta

```
/*
Descripción:
    - Se dará permiso a una persona con RUT "_rut" para firmar la venta con id "_idVenta".
Parámetros de Entrada:
    - uint      _idVenta
    - uint      _rut
Parámetros de Salida:
*/
function agregarPermisoParaFirmarVenta(uint _idVenta, uint _rut) public
soloUsuarioRegistrado
soloPropietarioVenta(_idVenta)
soloVentaDescripcionAgregada(_idVenta)
soloRutRegistrado(_rut) {
    ventas[_idVenta].permisoFirmar = usuarios[_rut].direccion;
    ventas[_idVenta].estado = 2; // Listo para firmar
}
```

Figura C.16: Agregar Permiso Para Firmar Venta

```
/*
Descripción:
    - Se finalizará la venta con id "_idVenta" de la propiedad con rol "_rolProp". El bien cambiará de dueño y se liberará el pago de la custodia con id "_idCustodia".
Parámetros de Entrada:
    - string    _clave
    - string    _rolProp
    - uint      _idVenta
    - uint      _idCustodia
Parámetros de Salida:
*/
function finalizarVentaPropiedad(string _clave, string _rolProp, uint _idVenta, uint _idCustodia) public
soloClave(_clave)
soloUsuarioRegistrado
soloPropietarioVenta(_idVenta)
soloVentaDocumentoCustodiaIngresado(_idVenta)
soloPropiedadVenta(_rolProp, _idVenta)
soloVentaCustodia(_idVenta, _idCustodia) {
    string memory _rolPropiedad = ventas[_idVenta].rolPropiedad;
    propiedades[_rolPropiedad].propietario = ventas[_idVenta].comprador;
    propiedades[_rolPropiedad].ultVenta = _idVenta;
    delete propiedades[_rolPropiedad].ventaActual;
    pertenencias[_rolPropiedad] = ventas[_idVenta].comprador;
    cantidadPropUsuario[ventas[_idVenta].vendedor] -= 1;
    cantidadPropUsuario[ventas[_idVenta].comprador] += 1;
    enVenta[_rolPropiedad] = false;
    ventas[_idVenta].estado = 5; // Finalizado con exito
}
```

Figura C.17: Finalizar Venta de Propiedad

```
/*
Descripción:
    - Se transferirá el dominio de la propiedad con rol "_rolProp" a la persona con RUT "_rut".
Parámetros de Entrada:
    - string    _clave
    - string    _rolProp
    - uint      _rut
Parámetros de Salida:
*/
function transferenciaPropiedad(string _clave, string _rolProp, uint _rut) public payable
soloClave(_clave)
soloUsuarioRegistrado
soloRolPropiedadValido(_rolProp)
soloPropietario(_rolProp)
soloPropiedadNoEnVenta(_rolProp)
soloRutRegistrado(_rut)
soloPagoTramiteValido {
    if( msg.value > precioTramite ) { // Si se deposita más de lo que vale el trámite se devuelve el excedente
        msg.sender.transfer( msg.value - precioTramite );
    }
    balance += precioTramite;
    propiedades[_rolProp].propietario = usuarios[_rut].direccion;
    pertenencias[_rolProp] = usuarios[_rut].direccion;
    cantidadPropUsuario[msg.sender] -= 1;
    cantidadPropUsuario[usuarios[_rut].direccion] += 1;
}
```

Figura C.18: Transferencia Propiedad

C.6: Funciones Comprador

```
/*
Descripción:
    - Se firmará la venta con id "_idVenta" con el hash de la clave pública del usuario que llama a la función.
Parámetros de Entrada:
    - uint      _idVenta
Parámetros de Salida:
*/
function firmarCompraPropiedad(uint _idVenta) public
soloUsuarioRegistrado
soloIdVentaValido(_idVenta)
soloVentaListaParaFirmar(_idVenta)
soloPermitidoFirmarVenta(_idVenta) {
    ventas[_idVenta].comprador = msg.sender;
    ventas[_idVenta].estado = 3; // Venta firmada
}
```

Figura C.19: Firmar Compra Propiedad

```
/*
Descripción:
    - Se agrega un documento de custodia con id "_idCustodia" junto con información relevante para la misma.
Parámetros de Entrada:
    - string    _clave
    - uint      _idCustodia
    - uint      _idVenta
    - uint      _rutVend
    - uint      _monto
Parámetros de Salida:
*/
function agregarDocumentoCustodia(string _clave, uint _idCustodia, uint _idVenta, uint _rutVend, uint _monto) public
soloClave(_clave)
soloUsuarioRegistrado
soloCustodiaNoExistente(_idCustodia)
soloIdVentaValido(_idVenta)
soloCompradorVenta(_idVenta)
soloVentaFirmada(_idVenta)
soloCustodiaMontoCorrecto(_monto, _idVenta) {
    agregarCustodia(_idCustodia, _rutVend, _monto);
    esCustodia[_idCustodia] = true;

    custodias[_idCustodia].idVenta = _idVenta;
    ventas[_idVenta].idCustodia = _idCustodia;
    ventas[_idVenta].estado = 4; // Documento de custodia ingresado
}
```

Figura C.20: Agregar Documento Custodia

```
/*
Descripción:
    - Se finaliza con error la venta correspondiente a la custodia con id "_idCustodia", la propiedad correspondiente no estará a la venta luego de esta acción.
Parámetros de Entrada:
    - string    _clave
    - uint      _idCustodia
Parámetros de Salida:
*/
function devolverPagoCustodia(string _clave, uint _idCustodia) public
soloClave(_clave)
soloUsuarioRegistrado
soloIdCustodiaValido(_idCustodia)
soloCompradorCustodia(_idCustodia)
soloCustodiaConVentaNoFinalizada(_idCustodia) {
    ventas[custodias[_idCustodia].idVenta].estado = 6; // No concretado
    string memory _rolPropiedad = ventas[custodias[_idCustodia].idVenta].rolPropiedad;
    propiedades[_rolPropiedad].ultVenta = custodias[_idCustodia].idVenta;
    delete propiedades[_rolPropiedad].ventaActual;
    enVenta[_rolPropiedad] = false;
}
```

Figura C.21: Devolver Pago Custodia

C.7: Funciones Propietario y Comprador

```
/*
Descripción:
    -   Se dará permiso a una persona de RUT "_rut" para poder ver la información de la venta con id "_idVenta".
        Se guardará el acceso para dicha persona "_claveAcceso" dependiendo de quien llame a la función (vendedor o comprador).
Parámetros de Entrada:
    - string    _clave
    - uint      _idVenta
    - uint      _rut
    - string    _claveAcceso
Parámetros de Salida:
*/
function darPermisoTemporalVenta(string _clave, uint _idVenta, uint _rut, string _claveAcceso) public
soloUsuarioRegistrado
soloRutRegistrado(_rut)
soloIdVentaValido(_idVenta)
soloVendedorCompradorVenta(_idVenta)
soloClave(_clave) {
    if( msg.sender == ventas[_idVenta].vendedor ) {
        ventas[_idVenta].accesoVendedor[usuarios[_rut].direccion] = _claveAcceso;
    } else {
        ventas[_idVenta].accesoComprador[usuarios[_rut].direccion] = _claveAcceso;
    }
}
```

Figura C.22: Dar Permiso Temporal Venta

```
/*
Descripción:
    - Se retornará el estado y el siguiente paso a seguir de una venta con id "_idVenta".
Parámetros de Entrada:
    - uint      _idVenta
Parámetros de Salida:
    - string    estado
    - string    siguientePaso
*/
function verEstadoVenta(uint _idVenta) public view
soloUsuarioRegistrado
soloIdVentaValido(_idVenta)
soloVendedorCompradorVenta(_idVenta)
returns(string, string) { //estado - siguientePaso
    return( estadoDescripcion[ventas[_idVenta].estado], estadoSgtePaso[ventas[_idVenta].estado] );
}
```

Figura C.23: Ver Estado Venta

```
/*
Descripción:
    - Se retornará el tipo de usuario (vendedor o comprador) correspondiente a la custodia con id "_idCustodia". El valor depende de quien llame a la función.
Parámetros de Entrada:
    - uint      _idCustodia
Parámetros de Salida:
    - string    tipoUsuario
*/
function soyEnCustodia(uint _idCustodia) public view
soloUsuarioRegistrado
soloIdCustodiaValido(_idCustodia)
soloVendedorCompradorCustodia(_idCustodia)
returns(string) { //tipoUsuario
    if( custodias[_idCustodia].vendedor == msg.sender ){
        return "vendedor";
    } else {
        return "comprador";
    }
}
```

Figura C.24: Soy en Custodia

C.8: Funciones de Consulta

```
/*
Descripción:
    - Se retornará "true" o "false" dependiendo si el usuario que llama a la función es creador o no.
Parámetros de Entrada:
Parámetros de Salida:
    - bool      soyCreador
*/
function soyCreador() public view
soloUsuarioRegistrado
returns(bool) { //soyCreador
    return msg.sender == creador;
}
```

Figura C.25: Soy Creador

```
/*
Descripción:
    - Se retornará "true" o "false" dependiendo si el usuario que llama a la función es tasador o no.
Parámetros de Entrada:
Parámetros de Salida:
    - bool      soyTasador
*/
function soyTasador() public view
soloUsuarioRegistrado
returns(bool) { //soyTasador
    return tasadores[msg.sender];
}
```

Figura C.26: Soy Tasador

```
/*
Descripción:
    - Se retornará "true" o "false" dependiendo si el usuario que llama a la función es propietario o no.
Parámetros de Entrada:
Parámetros de Salida:
    - bool      soyPropietario
*/
function soyPropietario() public view
soloUsuarioRegistrado
returns(bool) { //soyPropietario
    return cantidadPropUsuario[msg.sender] > 0;
}
```

Figura C.27: Soy Propietario

```
/*
Descripción:
    - Se retornará "true" o "false" dependiendo si la persona que llama a la función es usuario o no.
Parámetros de Entrada:
Parámetros de Salida:
    - bool      soyUsuario
*/
function soyUsuario() public view
returns(bool) { //soyUsuario
    return userRegistrado[msg.sender];
}
```

Figura C.28: Soy Usuario

```
/*
Descripción:
    - Se retornará el nombre completo del usuario que llama a la función.
Parámetros de Entrada:
Parámetros de Salida:
    - string    nombres
*/
function getNombreUsuario() public view
soloUsuarioRegistrado
returns(string) { //nombres
    return usuarios[addRut[msg.sender]].nombres;
}
```

Figura C.29: Get Nombre Usuario

```
/*
Descripción:
    - Se retornará el apellido paterno del usuario que llama a la función.
Parámetros de Entrada:
Parámetros de Salida:
    - string    apellidoPaterno
*/
function getApellidoPaternoUsuario() public view
soloUsuarioRegistrado
returns(string) { //apellidoPaterno
    return usuarios[addRut[msg.sender]].apellido_paterno;
}
```

Figura C.30: Get Apellido Paterno Usuario

```
/*
Descripción:
    - Se retornará el apellido materno del usuario que llama a la función.
Parámetros de Entrada:
Parámetros de Salida:
    - string    apellidoMaterno
*/
function getApellidoMaternoUsuario() public view
soloUsuarioRegistrado
returns(string) { //apellidoMaterno
    return usuarios[addRut[msg.sender]].apellido_materno;
}
```

Figura C.31: Get Apellido Materno Usuario

```
/*
Descripción:
    - Se retornará el RUT sin dígito verificador del usuario que llama a la función.
Parámetros de Entrada:
Parámetros de Salida:
    - uint      rut
*/
function getRutUsuario() public view
soloUsuarioRegistrado
returns(uint) { //rut
    return usuarios[addRut[msg.sender]].rut;
}
```

Figura C.32: Get Rut Usuario

```
/*
Descripción:
    - Se retornará el dígito verificador del RUT del usuario que llama a la función.
Parámetros de Entrada:
Parámetros de Salida:
    - string    digitoVerificador
*/
function getDigitoUsuario() public view
soloUsuarioRegistrado
returns(string) { //digitoVerificador
    return usuarios[addRut[msg.sender]].digitoVerificador;
}
```

Figura C.33: Get Digito Usuario

```
/*
Descripción:
    - Se retornará si un usuario es "vendedor", "comprador" o "externo" en una venta con id "_idVenta"
Parámetros de Entrada:
    - uint      _idVenta
Parámetros de Salida:
    - string    soyEnVenta
*/
function soyEnVenta(uint _idVenta) public view
soloUsuarioRegistrado
soloIdVentaValido(_idVenta)
returns(string) { //soyEnVenta
    if( ventas[_idVenta].vendedor == msg.sender ) {
        return "vendedor";
    } else {
        if( ventas[_idVenta].permisoFirmar == msg.sender || ventas[_idVenta].comprador == msg.sender ) {
            return "comprador";
        } else {
            return "externo";
        }
    }
}
```

Figura C.34: Soy en Venta

```
/*
Descripción:
    - Se retornará "true" o "false" si el RUT indicado en "_rut" esta registrado o no en la plataforma.
Parámetros de Entrada:
    - uint      _rut
Parámetros de Salida:
    - bool      rutRegistrado
*/
function rutEstaRegistrado(uint _rut) public view
soloUsuarioRegistrado
returns(bool) { //rutRegistrado
    return rutRegistrado[_rut];
}
```

Figura C.35: Rut esta Registrado

C.9: Funciones Internas

```
/*
Descripción:
    - Crea una nueva venta conectándola con la propiedad con rol "_rolProp" y con el vendedor con address "_vendedor".
Parámetros de Entrada:
    - string    _rolProp
    - address   _vendedor
Parámetros de Salida:
    - uint      idVenta
*/
function agregarVenta(string _rolProp, address _vendedor) internal
returns(uint) { //idVenta
    Venta memory vta;
    vta.rolPropiedad = _rolProp;
    vta.vendedor = _vendedor;
    uint idVenta = maxIdVta += 1;
    cantidadVentas += 1;
    ventas[idVenta] = vta;
    return(idVenta);
}
```

Figura C.36: Agregar Venta

```
/*
Descripción:
    - Crea una nueva custodia con id "_idCustodia", añadiendo al vendedor a partir de "_rutVend", al comprador de forma interna y asignando un valor "_monto".
Parámetros de Entrada:
    - uint      _idCustodia
    - uint      _rutVend
    - uint      _monto
Parámetros de Salida:
*/
function agregarCustodia(uint _idCustodia, uint _rutVend, uint _monto) internal {
    Custodia memory cust;
    cust.comprador = msg.sender;
    cust.vendedor = usuarios[_rutVend].direccion;
    cust.monto = _monto;

    custodias[_idCustodia] = cust;
}
```

Figura C.37: Agregar Custodia

```
/*
Descripción:
    - Si la propiedad con rol "_rolProp" está a la venta y la venta correspondiente aún no tiene un comprador se retornará el "celular" del propietario, en caso contrario se devolverá "NO DISPONIBLE".
Parámetros de Entrada:
    - string    _rolProp
Parámetros de Salida:
    - string    celular
*/
function getNumeroPropietario(string _rolProp) internal view returns(string) { //celular
    if( enVenta[_rolProp] ) {
        if( ventas[propiedades[_rolProp].ventaActual].estado == 0 ) {
            return usuarios[addRut[propiedades[_rolProp].propietario]].celular;
        } else {
            return "NO DISPONIBLE";
        }
    } else {
        return "NO DISPONIBLE";
    }
}
```

Figura C.38: Get Número Propietario