

Adrià Torija Ruiz

Diseño y desarrollo de una aplicación móvil para la gestión de contratos inteligentes

Ingeniería Informàtica

Trabajo de Fin de Grado

Dirigido por el

Dr. Jordi Castellà Roca y Cristòfol Daudén Esmel



UNIVERSITAT ROVIRA I VIRGILI

Tarragona, 2022

Índice

1	Introducción	2
1.1	Objetivos	3
1.2	Motivación	4
1.3	Organización de la memoria	5
2	Tecnologías y Herramientas Utilizadas	6
2.1	Conceptos básicos	6
2.1.1	Función hash	6
2.1.2	Firma digital	6
2.1.3	Criptografía asimétrica	6
2.2	Blockchain	6
2.3	Ethereum y Smarts contracts	8
2.3.1	Ethereum	8
2.3.2	SmartContracts	8
2.3.3	Dapps	8
2.4	Ganache Truffle Suite	8
2.5	Firebase	9
2.6	AndroidStudio	9
2.6.1	Secure Element	9
3	Arquitectura y diseño	10
3.1	Estructura general	10
3.1.1	Fase 1	11
3.1.2	Fase 2	14
3.1.3	Fase 3	15
3.1.4	Fase 4	15
3.1.5	Fase 5 y 6	15
3.2	Comunicaciones en la arquitectura general	16
3.3	Aplicación Android	17
3.3.1	Requisitos de funcionamiento	17
3.3.2	Funcionalidades del Usuario	18
4	Implementación	25
4.1	Aplicación Java	25
4.2	Aplicación Android	28
5	Juego de pruebas	31
5.1	Test de proceso previo	31
5.2	Test de aplicación Android	32
5.3	Recuperar la contraseña	34
5.4	Inicio de sesión	37
5.5	Select Smart contract	39
5.6	Select Option	40
5.6.1	Obtener información contrato	40
5.6.2	Modificar información	42
5.6.3	Modificar Consentimiento	43
5.6.4	Comprobar el historial del contrato	46

6 Conclusiones	48
6.1 Trabajo Futuro	48
Referencias	49

Resumen

El rápido avance de nuevas tecnologías ha cambiado la dinámica de nuestras vidas previéndonos con nuevos servicios y dispositivos. Sin embargo, estas innovaciones permiten a los proveedores de servicio generar y agregar una gran cantidad de datos e información sobre los usuarios, los cuales pueden ser procesados para mejorar dichos servicios o financiar todos estos avances. Esto puede suponer un riesgo para la privacidad de dichos usuarios, ya que entre estos datos puede haber información sensible sobre ellos (ideologías políticas, rutina diaria, orientación sexual, etc.).

Para mitigar esto, la UE aprobó el RGPD[5], según la cual, los proveedores de servicio necesitan el consentimiento explícito de los usuarios para obtener y procesar sus datos personales. Sin embargo, dicha regulación no indica como obtener este consentimiento ni provee ninguna herramienta para llevarlo a cabo. Así, han aparecido una gran cantidad de propuestas [18] [6] [8] [3] que pretenden migrar del modelo actual de consentimiento, un modelo centralizado, a un modelo distribuido mediante el uso de la tecnología blockchain y los contratos inteligentes.

El uso de la blockchain tiene sus ventajas, el problema es que no se ofrece una herramienta para que los usuarios inexpertos de estas tecnologías, puedan llegar a tener un control sobre estos contratos inteligentes generados, no pudiendo interactuar con estos.

Por eso, en este trabajo se crea una aplicación Android para que el usuario pueda gestionar sus contratos inteligentes. Todos estos contratos están desplegados en una blockchain, de esta forma la información está repartida de forma distribuida y pública, donde el usuario podrá modificar los consentimientos que haya aceptado. La modificación de estos contratos inteligentes se lleva a cabo a través del dispositivo móvil. Así, la aplicación permite al usuario una gestión cómoda e intuitiva de dichos contratos.

1 Introducción

Actualmente, estamos en un periodo de desarrollo de nuevas tecnologías, las cuales se encuentran en pleno crecimiento y van ganando un gran protagonismo. Un gran ejemplo que tenemos son las blockchain. Esta idea empezó en los años 1991. En 2008 se presentó el paper: Bitcoin: A Peer-to-Peer Electronic Cash System [15]. Este describe una moneda digital llamada bitcoin y la tecnología detrás de ella, la blockchain.

Esta tecnología permite tener la información distribuida y de forma pública. Toda la información de las transacciones queda registrada para todos los usuarios y no puede ser eliminada. Otorgando más seguridad a los usuarios cada vez que se haga una transacción, ya que esta quedara registrada. Esta tecnología ha sido utilizada por diferentes monedas digitales en los últimos años, el ejemplo más importante ha sido bitcoin. Por otro lado, Ethereum sacó un nuevo concepto al mundo en la tecnología blockchain, los contratos inteligentes.

Los contratos inteligentes son fragmentos de código que permiten facilitar el intercambio de información, dinero o cualquier elemento que pueda llegar a tener un valor. Estos contratos se forman gracias a una máquina virtual llamada Ethereum Virtual Machine (EMV). Cuando un contrato está activo en una blockchain, este actúa como un programa informático que se va ejecutando automáticamente, y gracias a que estos están distribuidos y descentralizados, su ejecución se produce sin censuras ni caídas. [1]

Un caso de contratos inteligentes es la propuesta [6]. En este caso cada vez que se requiere el consentimiento de un usuario para almacenar y/o procesar los datos de un usuario se genera un contrato inteligente. En esta propuesta es necesaria la creación de una herramienta que permita conocer los contratos inteligentes (consentimientos) que ha dado un usuario. Esta operación solamente la debe poder realizar el propietario del contrato inteligente, al igual que hacer cualquier modificación del mismo.

Para que los usuarios tengan acceso a sus contratos inteligentes desde el dispositivo móvil, estas claves tendrán que ser guardadas en el cloud para poder ser posteriormente descargadas y guardadas de forma segura en el dispositivo móvil.

Se utiliza el servicio de Google de Firebase, que permite hacer la gestión de la aplicación Android: ya sea guardar la base de datos de los usuarios para su correcta autenticación; o guardar la información propia de cada usuario, como por ejemplo la información para acceder a sus contratos inteligentes.

1.1 Objetivos

Diseñar un sistema de gestión de contratos inteligentes, robusto, modular, fácil de usar y seguro, que permita obtener los contratos inteligentes del usuario para visualizar su información y si el usuario desea modificarlos, todo esto de manera segura.

La aplicación está destinada a la gestión de contratos inteligentes los cuales tienen los consentimientos de los datos del usuario.

Para que el sistema sea robusto, cada usuario debe ser el único que pueda llegar a hacer esa modificación de información, a partir de la validez de un proceso de autenticación.

Una implementación modular permitirá añadir nuevas funcionalidades a la aplicación sin modificar su estructura prácticamente, pudiendo añadir complementos o funcionalidades. Adaptándose a los posibles cambios de estas nuevas tecnologías o mejorar las funcionalidades y su rendimiento.

Para que esta aplicación sea fácil de usar, la interfaz de usuario tiene que ser clara, permitiendo que se pueda aprender de forma autodidáctica sus funcionalidades.

Gracias al sistema de usuarios y a su autenticación se podrá utilizar la aplicación de forma controlada y segura, permitiendo recuperar la contraseña en caso de pérdida. Además, al ser una aplicación en un dispositivo móvil, hay una posible autenticación extra, la autenticación biométrica.

1.2 Motivación

Este sistema tiene como finalidad aportar a todos los usuarios cotidianos una herramienta para poder gestionar los contratos inteligentes que contienen sus consentimientos hacia los proveedores de servicios web que extraen su información., de manera descentralizada y segura.

Esta aplicación permitirá ver fácilmente que consentimientos y finalidades están siendo aceptados por el usuario, con la posibilidad de modificar estos permisos cuando el usuario lo desee.

Al utilizar la blockchain, todos los cambios realizados por el usuario o el controlador web, quedarían registrados en la blockchain de forma permanente. Además, aportaría un sistema descentralizado y seguro.

El sistema no ha de ser complejo, ya que se quiere llegar a un nivel de usuario general, siendo un sistema fácil de utilizar a la vez que de aprender, teniendo una interfaz muy visual y clara para el usuario. Este sistema permitirá autenticarse y recuperar las credenciales.

1.3 Organización de la memoria

En la memoria se tratará de explicar el proceso para la creación del sistema y estará dividida en 6 apartados.

En la Sección 2 se describen brevemente las tecnologías esenciales utilizadas en el trabajo. La sección 3 contiene el diseño del sistema junto a los requisitos necesarios. A demás se explican las funcionalidades de los usuarios. La Sección 4 se explica las funcionalidades que ofrecen cada una de las estructuras definidas en el diseño. La Sección 5 evalúa el correcto funcionamiento del sistema y sus funcionalidades, junto a sus resultados. Y finalmente en la Sección 6 relata las conclusiones obtenidas a partir de la realización del trabajo y las sensaciones obtenidas gracias a su realización.

2 Tecnologías y Herramientas Utilizadas

En este apartado se explican las diferentes tecnologías y aplicaciones utilizadas para realizar este trabajo.

2.1 Conceptos básicos

2.1.1 *Función hash*

Una función hash es una operación criptográfica unidireccional que genera identificadores únicos irrepetibles a partir de una información. Estos son una pieza clave en la tecnología blockchain, ya que estas tienen como objetivo codificar datos para formar cadenas de caracteres única. Sirven para asegurar la autenticidad de los datos, almacenar de manera segura contraseñas, y firmar documentos electrónicos.

2.1.2 *Firma digital*

Una firma digital es un método criptográfico que permite autenticar digitalmente a un usuario. Permitiendo verificar la autenticidad e integridad de los datos digitales.

2.1.3 *Criptografía asimétrica*

La criptografía asimétrica permite establecer una conexión segura entre dos partes, autenticando cada una de estas partes, permitiendo el traspaso de información entre estas.

Cada usuario tiene un par de claves, la privada solo ha de ser conocida por el propietario de la misma y su clave pública estará disponible para todos los usuarios del sistema. Si un usuario quiere enviar un mensaje a otro este deberá cifrar el mensaje a partir de la clave pública del otro usuario, y se enviará el mensaje cifrado. Este mensaje solo podrá ser descifrado por el usuario que tenga la clave privada perteneciente a esa clave pública.

2.2 Blockchain

El concepto de blockchain fue descrito por primera vez en 1991 por Stuart Haber i W. Scott Stornetta. [12]. En 2008, Satoshi Nakamoto conceptualizó la teoría de las blockchains distribuidas junto al concepto de bitcoin, esto se ve reflejado en el paper Bitcoin: A Peer-to-Peer Electronic Cash System [15]. Esta tecnología está basada en una cadena de bloques donde la base de datos está guardada públicamente y distribuida. En esta se registran de forma segura todas las transacciones que se van realizando en la red y se utilizan funciones hash para ello. La red puede estar en Internet o por ejemplo se puede crear una red local para poder ejecutar tests.

En esta cadena cada vez que se introducen nuevos datos se crea una marca de tiempo y un enlace al bloque anterior y estos datos son verificables y no se pueden manipular, eso significa que una vez hecha una transacción esta misma no se puede modificar esto sucede gracias a las características de las funciones hash.

Esta cadena de bloques tiene como finalidad registrar todas las transacciones de todos los participantes de la red de la blockchain, la cual las almacena y se comparten de forma pública. Actualmente, las blockchains pueden ser públicas, privadas, autori-

zadas o construidas por un consorcio, todo esto dependerá de quien se puede unir a la blockchain y de la forma en la que un usuario puede llegar a ser partícipe.

Estos bloques están formados de varias transacciones, hay un bloque raíz y a partir de él se unen diferentes bloques en la cadena. Estos se componen de un código alfanumérico para poder-se enlazar con el bloque anterior. Estos bloques contienen el número de transacciones y el código alfanumérico para unir cada transacción con el bloque siguiente. Estos necesitan una serie de cálculos para que llegar a ser válidos, y estos cálculos son verificados por los mineros que son los encargados de ir verificando para ir creando bloques. En la figura 2.1 podremos observar un ejemplo de como funciona.

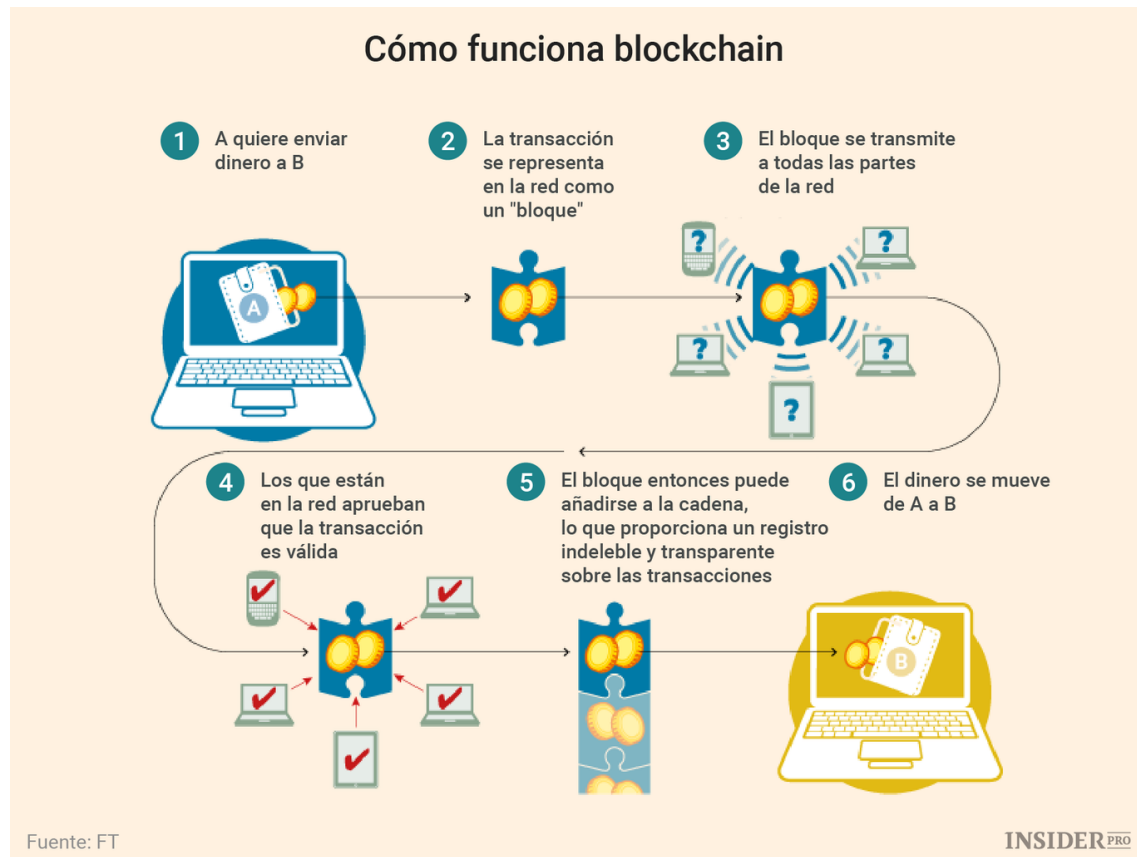


Figura 2.1: Como funciona una blockchain [16]

Estos mineros participan en el proceso de minería, este proceso se realiza para que las operaciones sean más seguras. Estos mineros pueden llegar a realizar millones de cálculos por segundo, calculando los millones de hashes permitiendo que una transacción es verificada y se completa un nuevo bloque.

En el proceso de registro de las transacciones, estas tienen que estar firmadas. Para poder hacer la firma de estos datos se utiliza la clave privada del usuario, esta clave ha sido creada gracias a la criptografía asimétrica. [2.1.3]. Esto permite la autenticación, integridad y el no repudio de estas transacciones.

Antes se ha comentado que este sistema era un sistema distribuido. Para llegar a conseguir este sistema se utilizará el término nodo. Los nodos son ordenadores que están conectados a la red y se encargan de guardar y distribuir una copia actual de cada bloque. Esto permite que si un nodo dejase de funcionar o se perdiese no cambiaría nada

en la cadena de bloques, haciendo que este sistema sea resistente a ataques informáticos y a fallos.

2.3 Ethereum y Smarts contracts

En esta sección se describirá como funciona esta tecnología y como se ha desarrollado y creado el concepto de Smarts contracts (contratos inteligentes). [19]

2.3.1 *Ethereum*

Ethereum es una plataforma de código abierto basada en la tecnología blockchain. Esta blockchain funciona como un registro del historial de transacciones, que además permite construir y desplegar aplicaciones descentralizadas o dapps gracias a esta red.

2.3.2 *SmartContracts*

Para entender el concepto de contrato inteligente (Smart contract) tendremos que primero centrarnos en que es un contrato. Un contrato es un acuerdo entre dos o más partes donde se define todo lo que puede suceder, estos normalmente tienen que estar verificados por una tercera persona.

Un contrato inteligente es un script de código que facilita un intercambio de información, dinero, o acciones con valor. Estos contratos se forman con la máquina virtual llamada Ethereum Virtual Machine(EVM), para la programación de esta máquina se utiliza un lenguaje especializado llamado Solidity, que facilita la creación de contratos inteligentes. Una vez este contrato está activo en la red, este se va ejecutando automáticamente, sin poder tener caídas ni censuras.[1]

Lo que permite este contrato es hacer un acuerdo entre dos o más partes, siendo capaz de hacerse cumplir por sí mismo de manera autónoma y automática, sin intermediarios ni mediadores. Y tiene como ventaja que al estar ejecutándose en una blockchain, de manera distribuida, se evita que esta sea controlada por diferentes autoridades y reflejándolo todo públicamente.

2.3.3 *Dapps*

Las dapps [3] son programas informáticos de código abierto que emplean la tecnología de la blockchain. Estas no necesitan un intermediario para funcionar permitiendo su descentralización. Estas dapps se crean a partir de grupos de contratos inteligentes que se comentaran en el siguiente apartado.

2.4 Ganache Truffle Suite

Ganache Truffle Suite [4] es un framework que nos permite iniciar una blockchain privada, creándose manera local, para poder ejecutar tests, comandos e inspeccionar el estado de las operaciones gracias a la interfaz gráfica y sencilla de utilizar que nos proporciona.

Gracias a este framework se podrá obtener direcciones de usuarios, y hacer transacciones para la creación y modificación de los Smart contracts, ya que esta blockchain está basada en Ethereum.

2.5 Firebase

Firebase [10] es una aplicación para el desarrollo de aplicaciones móviles. Esta aplicación proporciona diferentes utilidades para el back-end de una aplicación.

Las principales utilidades son la autenticación, donde hay un sistema de usuarios, los cuales se pueden personalizar con el uso de varios proveedores de usuarios, ya sea Google, Microsoft, un email...

Aparte también proporciona una base de datos con un lenguaje NO-SQL, Firestore [11]. Esta facilita las solicitudes de subidas y bajadas de datos, permitiendo poner normas para hacer el acceso de los datos más específicos para que solo los usuarios con permiso puedan leerlos o escribirlos.

Las estructuras básicas de la base de datos Firestore pueden ser:

- Colecciones: Pueden contener varios documentos.
- Documentos: Pueden contener varios tipos de valores e información.

2.6 AndroidStudio

AndroidStudio [9] es un IDE (entorno de desarrollo integrado) oficial que permite desarrollar aplicaciones Android. Este permite trabajar tanto como en el backend de la aplicación como con la interfaz de esta, ofreciendo diversas herramientas para el diseño de esta o por ejemplo ofreciendo la herramienta de probar la aplicación Android ya sea en un emulador o dispositivo móvil conectado.

2.6.1 *Secure Element*

El SecureElement [13] es un chip contenido en el teléfono que permite almacenar información. Este tiene un sistema operativo independiente y los sistemas operativos de dispositivos móviles no pueden leer o copiar esta información. Este elemento, pero, solo funciona en aplicaciones especiales de confianza, normalmente monederos de confianza.

Basándonos en este sistema se utilizará una utilidad que nos proporciona Android denominada KeyStore. Esta utilidad permite almacenar claves criptográficas en un contenedor para hacer más difícil extraer claves del dispositivo.

3 Arquitectura y diseño

En este apartado se explica la estructura general de la propuesta, explicando los diferentes módulos que esta contiene y como se establecen sus comunicaciones. A demás se establece la arquitectura y diseño de la aplicación Android, como sus requisitos necesarios o las funcionalidades del usuario.

3.1 Estructura general

En la figura 3.1 podemos observar que esta arquitectura general principalmente se utilizan 4 módulos. La aplicación principal está representada de color verde (Android App), esta utiliza los módulos de color amarillo (Firebase, Blockchain) los cuales son necesarios para el uso de la aplicación, ya que estos guardan la información de los contratos o los usuarios. Por otro lado, el módulo de color naranja (JavaProgram) está destinado a crear esta información que es guardada en la blockchain y en Firebase.

1. Blockchain: Permite tener una blockchain en una red local, pudiendo generar bloques con transacciones.
2. JavaProgram: Permite generar contratos inteligentes y subirlos a la blockchain. Además, genera llaves para el usuario y guarda estas llaves y las direcciones de los contratos inteligentes en la base de datos.
3. Firebase: Permite manejar la autenticación de la aplicación Android y tener la base de datos del JavaProgram y de la aplicación Android
4. Aplicación Android: Permite obtener las llaves del usuario y las direcciones de los contratos de la base de datos. Obtiene la información y hacer modificaciones utilizando la blockchain

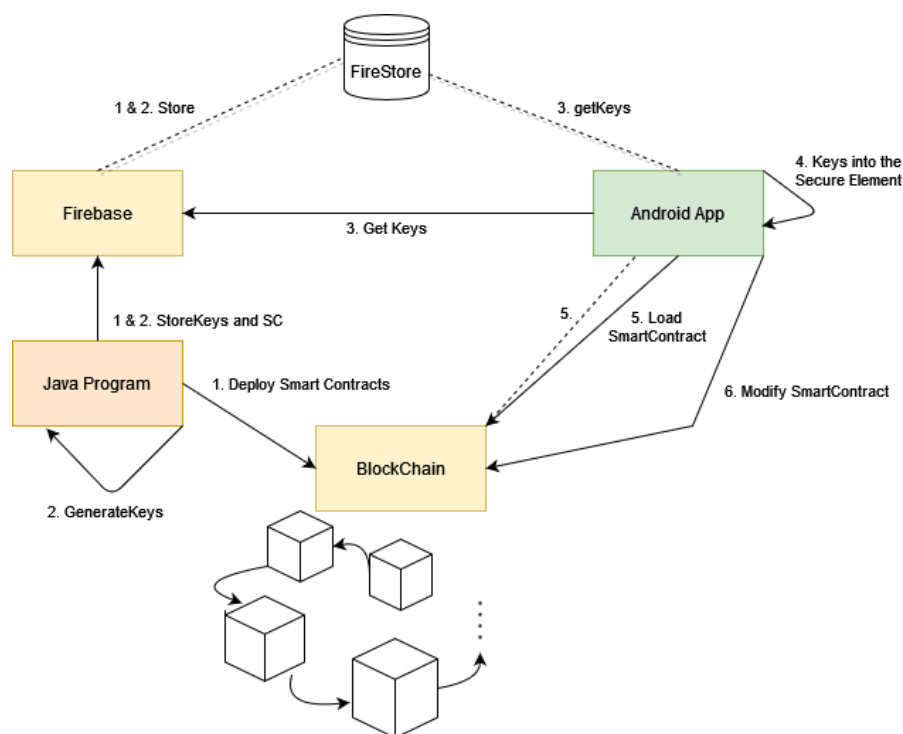
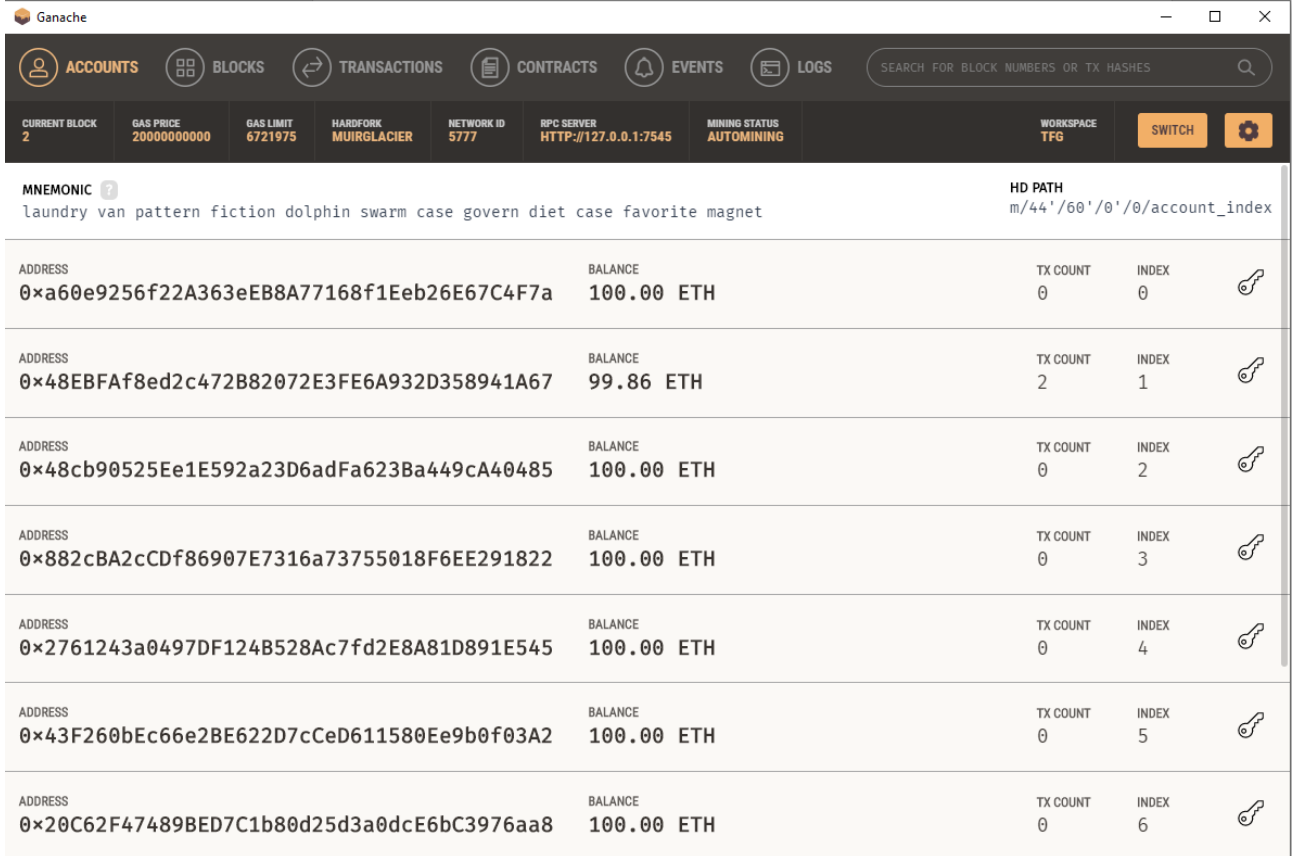


Figura 3.1: Estructura general

3.1.1 Fase 1

En esta primera fase se tiene que crear una blockchain en una red local, esto permitira hacer varias pruebas de manera privada. Para ello utilizaremos la aplicación Ganache Truffle Suite mencionada en el apartado 2.4. Aqui tenemos un ejemplo de una blockchain generada en nuestra red local [3.2](#)



The screenshot shows the Ganache application interface. At the top, there's a navigation bar with icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this is a status bar showing various network parameters like CURRENT BLOCK, GAS PRICE, GAS LIMIT, HARDFORK, NETWORK ID, RPC SERVER, MINING STATUS, and WORKSPACE. The main area displays a list of accounts with their addresses, balances, transaction counts, and indices. The mnemonic phrase is also visible at the top.

ADDRESS	BALANCE	TX COUNT	INDEX
0xa60e9256f22A363eEB8A77168f1Eeb26E67C4F7a	100.00 ETH	0	0
0x48EBFAf8ed2c472B82072E3FE6A932D358941A67	99.86 ETH	2	1
0x48cb90525Ee1E592a23D6adFa623Ba449cA40485	100.00 ETH	0	2
0x882cBA2cCDf86907E7316a73755018F6EE291822	100.00 ETH	0	3
0x2761243a0497DF124B528Ac7fd2E8A81D891E545	100.00 ETH	0	4
0x43F260bEc66e2BE622D7cCeD611580Ee9b0f03A2	100.00 ETH	0	5
0x20C62F47489BED7C1b80d25d3a0dcE6bC3976aa8	100.00 ETH	0	6

Figura 3.2: Ejemplo Ganache Blockchain

En esta blockchain se suben los contratos inteligentes los cuales utilizaremos para hacer las pruebas de la aplicación Android. Para ello, en este apartado se define la estructura de los Smart contracts para la gestión de las cookies de nuestro navegador web. Estos contratos están descritos en el paper Lightweight Blockchain-based Platform for GDPR-compliant Personal Data Management [6] y haremos una descripción de los atributos y las operaciones de estos.

Smart Contracts del proyecto Los Smart Contracts de este proyecto han sido generados gracias al lenguaje de alto nivel llamado Solidity [17], orientado en la creación de contratos inteligentes para la blockchain. Para generar el código Java de la clase Solidity se utilizara web3j [14]

A continuación se explicaran los Smart contracts utilizados, descritos en el paper Lightweight Blockchain-based Platform for GDPR-compliant Personal Data Management [6]

1) Consent Smart contract:

Este contrato tiene los siguientes atributos:

- Identities: Claves públicas del Data Subject, Controller y recipientes
- Data: Contiene las categorías de información que son recogidas.
- Consent Lifetime: Periodo en el cual los datos pueden estar recogidos y guardados
- Default Purposes: Contiene una lista de finalidades para las cuales cada Data Subject está permitiendo que procesen su información personal.
- Purposes: Link hacia todos los Purposes Smart contract
- Valid Flag: Valor que debe estar en 1 para que este contrato sea válido
- Erasure Flag: Valor que si esta en uno el DataController tiene que eliminar toda la información recogida del usuario.

Este contrato tendrá las siguientes funciones:

- grantConsent() y revokeConsent(): Modifican el valor del valid flag.
- verify(): Comprueba si el contrato es válido gracias al valid flag.
- modifyData(): Modifican el contenido del atributo Data.
- eraseData(): Pone el atributo ErasureFlag a 1.
- revokeConsentPurppose(): Quita el consentimiento de todos los Purpose Smart contract creados por este contrato.

2) Purpose Smart contract:

Este contrato tiene los siguientes atributos:

- Identities: Claves públicas del Data Subject y Data Controller con la dirección del DataProcessor.
- Data: Qué información puede ser procesada.
- Processing Purpose: Finalidad en específico para la cual se puede procesar información.
- Contract Lifetime: Periodo en el cual los datos pueden ser procesados.
- Validity Array: Array de cuatro elementos para la validez del DataSubject, DataController, Dataprocessor. Estas posiciones tienen que ser válidas para que el Purpose Smart contract válido

3) Flujo de trabajo

Estos contratos serán utilizados en un entorno real de la siguiente manera descrita también en el paper mencionado anteriormente [6].

Para simplificar la explicación se utilizarán las siguientes notaciones:

Notation	Description
DC	Data Controller
DP	Data Processor
DR	Data Recipient
DS	Data Subject
SA	Supervisory Authority
SC	Smart Contract
SP	Service Provider

Cuadro 1: Notaciones

En la figura 3.3 se puede observar las interacciones entre las entidades y los SC. Este es el proceso que se llevara a cabo durante la creación de un smartContract.

1. El Data subject se suscribe al DC (acepta las cookies), el DC crea un nuevo-Consent SC, especificando los atributos comentados anteriormente en el apartado 3.1.1. Si el DS está de acuerdo con este contrato, aceptará su permiso. A partir de este momento, el DC recoge la información personal del DS que sera guardada en el DR. Para cada Consent SC, el DS usara una nueva dirección pública y privada de claves para identificar-se a sí mismo y firmar las transacciones. Esto se hace para evitar el ataque de enlace utilizando el SC con otros DC.
2. Una vez el Consent SC sea válido, el DP puede solicitar al DC procesar la información recolectada. Si el DC accede se crea un nuevo Purpose SC especificando que data será procesada, para que propósito, quien procesara la información y la duración del permiso.
3. Una vez el Consent SC sea válido, el DP puede solicitar al DC procesar la información recolectada. Si el DC accede se crea un nuevo Purpose SC especificando que data será procesada, para que propósito, quien procesara la información y la duración del permiso.
4. Una vez el Consent SC sea válido, el DP puede solicitar al DC procesar la información recolectada. Si el DC accede se crea un nuevo Purpose SC especificando que data será procesada, para que propósito, quien procesara la información y la duración del permiso.
5. Si el DS acepta la finalidad del Purpose SC dará consentimiento al Purpose SC. (Esto solo será solicitado si no es una de las finalidades predefinidas.)
6. Si el DS acepta el SC necesitará el consentimiento del DP para que sea válido, asegurando el proceso que hacía la solicitud.
7. Cuando este sea válido, el DP puede solicitar la información personal del DS para el DR, donde el DR verifica si esta es válida, dando acceso a la información en caso afirmativo.
8. El DS tiene el derecho de modificar que información puede ser recolectada o procesada. También podrá eliminar toda la información o denegar el consentimiento previamente dado.

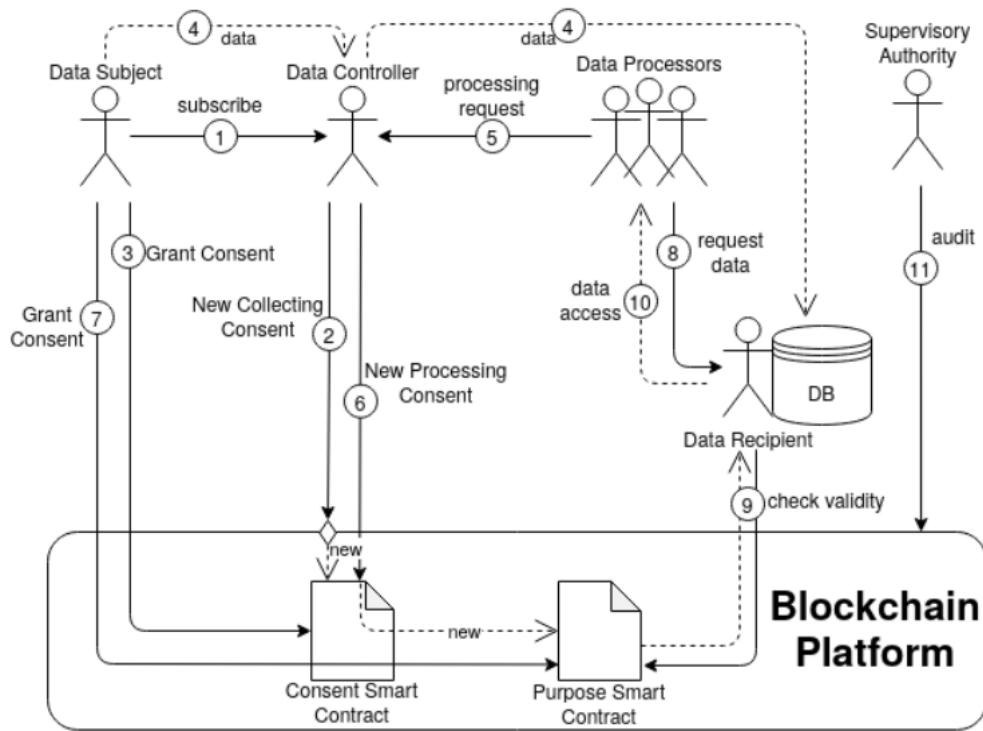


Figura 3.3: Diagrama de Flujo Smart contracts. Fuente:[6]

3.1.2 Fase 2

Gracias al código [7] podemos hacer un deploy de los Smart contracts en la block-Chain local. Hay diferentes modificaciones en el código para poder generar las llaves para el usuario y guardar estas y las direcciones de los Smart contract generados en una base de datos en el cloud.

Para esta funcionalidad utilizaremos la plataforma descrita en el apartado 2.5, Firebase. En concreto, se utiliza Firestore para trabajar en el cloud a partir de la aplicación java. El proveedor nos proporciona unas reglas que permiten establecer los permisos de escritura y lectura de los datos, como se puede observar en la figura 3.4. A demás las claves se almacenan de forma segura en Firebase de acuerdo con la propuesta publicada en [2]

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /Users/{userId}{
      allow update: if request.auth != null && request.auth.uid == userId;
      allow read, create: if request.auth != null;
    }

    match /Users/{userId}/{document=**}{
      allow read, write: if request.auth != null && request.auth.uid ==userId;
    }
  }
}
```

Figura 3.4: Reglas de Firebase

Se utiliza la siguiente estructura de datos, que se puede observar en la figura 3.5.

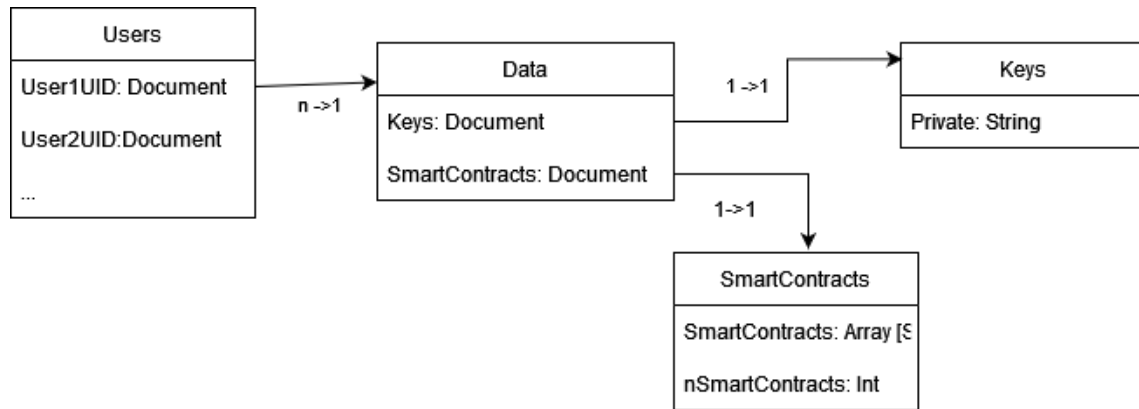


Figura 3.5: Estructura de datos

Esta estructura sigue el siguiente orden

1. Usuarios: Colección de usuarios, depende de cuantos usuarios haya registrados en el sistema y cada uno será un documento identificado por el UID de cada usuario.
2. Data: Este documento tiene una colección que contiene los documentos de información del usuario.
 - Keys: Contiene la clave privada del usuario.
 - SC: Contiene la lista de contratos del usuario y cuantos contratos tiene.

Cada vez que el usuario cree varios Smart contracts se generara una clave pública y privada, utilizando la clave privada del usuario obtenida al generar la blockchain, esta es guardada en el cloud. Esto lo que permite es conseguir una criptografía asimétrica con la blockChain, pudiendo identificar al usuario con esta.

3.1.3 Fase 3

Una vez generado un entorno de pruebas para nuestra aplicación generando un usuario en Firebase y creando las llaves y los Smart contracts, el usuario se descarga e instala la aplicación. A continuación, el usuario hará el inicio de sesión con las credenciales guardadas en Firebase. Si estas credenciales fuesen correctas, la aplicación Android solicitará la clave privada del usuario a través de Firebase. El cual comprueba la identificación del usuario como se ha comentado en el apartado 3.1.2

3.1.4 Fase 4

Al obtener estas llaves, pública y privada, se guardan de forma segura en nuestra aplicación Android. Existe un elemento que facilita esta seguridad de las llaves en Android, este se llama SecureElement, descrito en el apartado de tecnologías.

3.1.5 Fase 5 y 6

Una vez obtenidas las direcciones de los Smart contract de un usuario, podremos obtener la información de estos contratos a partir de la blockchain. A partir de las funciones

integradas en estos contratos; operaciones de modificación, como por ejemplo modificar la información, o el consentimiento. También se utilizan funciones de lectura, para poder ver qué información tiene el contrato y si este es válido.

3.2 Comunicaciones en la arquitectura general

En este apartado se explican las diferentes comunicaciones que se producen entre los módulos descritos en el apartado. 3.1.

En la figura 3.6 podemos observar las conexiones que están realizadas en este proyecto.

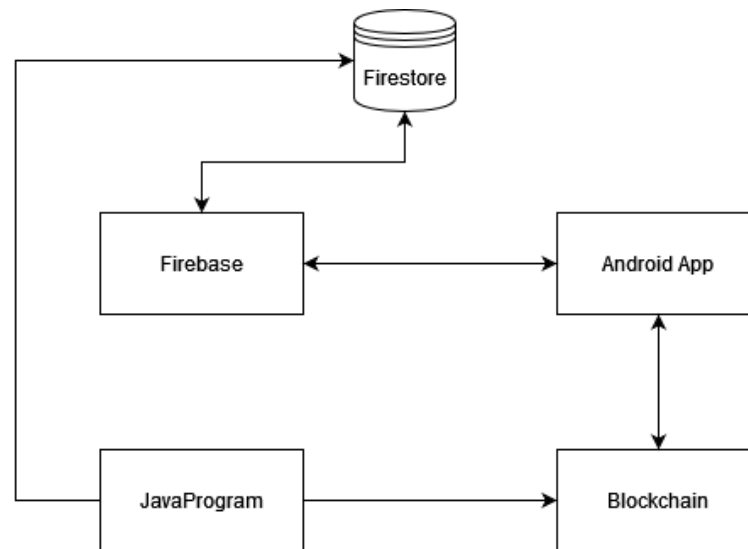


Figura 3.6: Comunicación General

A continuación se explicarán en detalle estas conexiones.

- Conexiones hacia la blockchain: Estas conexiones son realizadas a través de una librería Web3j [14]. Esta nos permite trabajar con una blockchain de Ethereum de manera segura. Para realizar estas conexiones al ser una blockchain local se han hecho las conexiones a través de HTTP iniciando esta conexión con la propia librería Web3j.
- Conexión JavaProgram y la base de datos: Esta conexión se realiza a través de las librerías de Firebase en Java. Estas nos permiten realizar una conexión a la base de datos (Firestore) gracias a unas credenciales que son generadas a partir de la API de Firebase.
- Conexión Android y la base de datos: Esta conexión se realiza gracias al SDK de Firebase que proporciona diversas librerías para ayudar a esta conectividad. A partir de unas reglas de seguridad que administra el usuario administrador de Firebase se consigue crear un proceso de autenticación que permite controlar el acceso a la base de datos de Firebase (Firestore).

3.3 Aplicación Android

Una vez establecido la arquitectura y diseño general del proyecto para que este funcione, estableceremos la arquitectura y diseño de nuestra aplicación Android.

3.3.1 *Requisitos de funcionamiento*

El sistema debe cumplir los siguientes requisitos para realizar correctamente su funcionalidad:

- La interfaz de usuario debe ser clara, sencilla y fácil de entender y utilizar, informándose al usuario de los resultados obtenidos de sus acciones.
- La aplicación guarda los datos sensibles para la autenticación del usuario en el dispositivo móvil.
- Solo el usuario propietario del dispositivo es capaz de realizar las modificaciones en cada uno de sus respectivos Smart contracts.
- Solo se permite acceso a un usuario previamente registrado y correctamente autenticado
- La administración de los contratos por parte de la aplicación será modular, pudiendo administrar-los y modificar-los fácilmente.
- Las comunicaciones en el sistema tienen que ser auténticas y seguras.

3.3.2 Funcionalidades del Usuario

Las diversas acciones que puede realizar un usuario en el sistema mediante la aplicación Android se pueden observar en la siguiente figura: 3.7

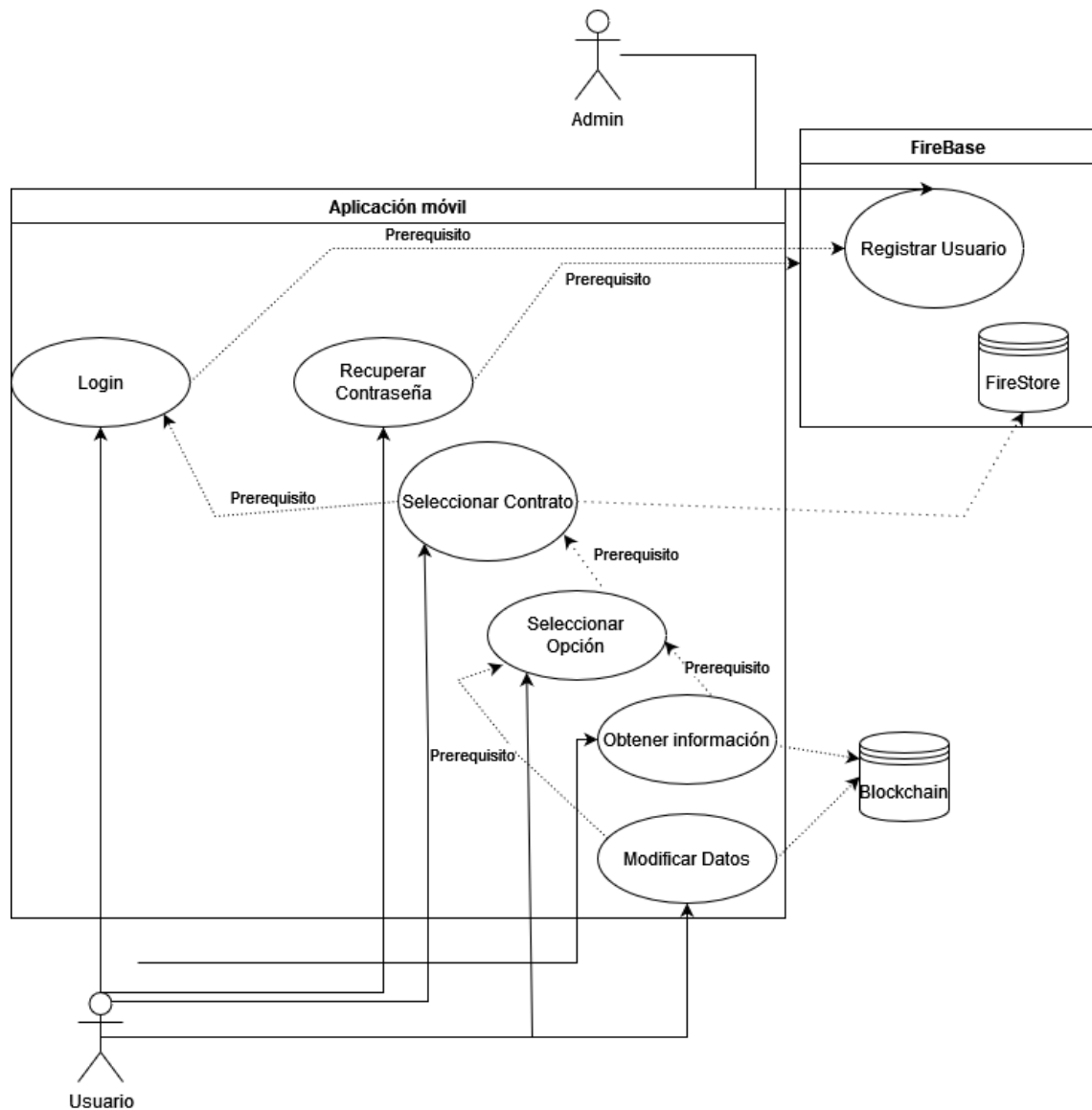


Figura 3.7: Diagrama de Casos de Usuario

La aplicación Android contiene las siguientes acciones:

1. Login: El usuario introduce su correo electrónico y contraseña para acceder a su cuenta.
2. Recuperar Contraseña: El usuario solicita una recuperación de contraseña a través del servicio de Firebase. Para ello, el usuario debe introducir el correo electrónico de la cuenta que se desea recuperar y se enviara un correo electrónico con las instrucciones para restablecer esta contraseña.
3. Seleccionar Contrato: Si el usuario tiene algún contrato en la blockchain, tiene esta dirección en su base de datos y podrá seleccionar cualquiera de las direcciones que

posea. Esto permite el uso de los casos de uso de obtener información y modificar datos.

4. Obtener información: Si el usuario ha seleccionado algún contrato, este accede a su información.
5. Modificar Datos: Si el usuario ha seleccionado algún contrato, podrá modificar su información o dar o quitar el consentimiento al Smart contract.

A continuación se explicaran cada uno de los casos de uso en detalle.

Login:

Esta funcionalidad mostrará al usuario una interfaz con diversos campos para poder introducir el correo electrónico y contraseña del usuario. Una vez rellenado estos campos, el usuario apretará un botón para hacer el intento de iniciar sesión. La aplicación entonces recibe toda la información escrita por el usuario y comprueba que los campos estén bien escritos, en caso erróneo se mostrara un error al usuario. Si esta información es correcta, se envía a la base de datos de usuarios de Firebase, la cual enviara una respuesta ala aplicación y esta hará una actualización de la página, con un inicio de sesión y cambio de actividad si esta respuesta es positiva o con un informe de error para el usuario.

Todo este proceso se puede observar en la figura 3.8

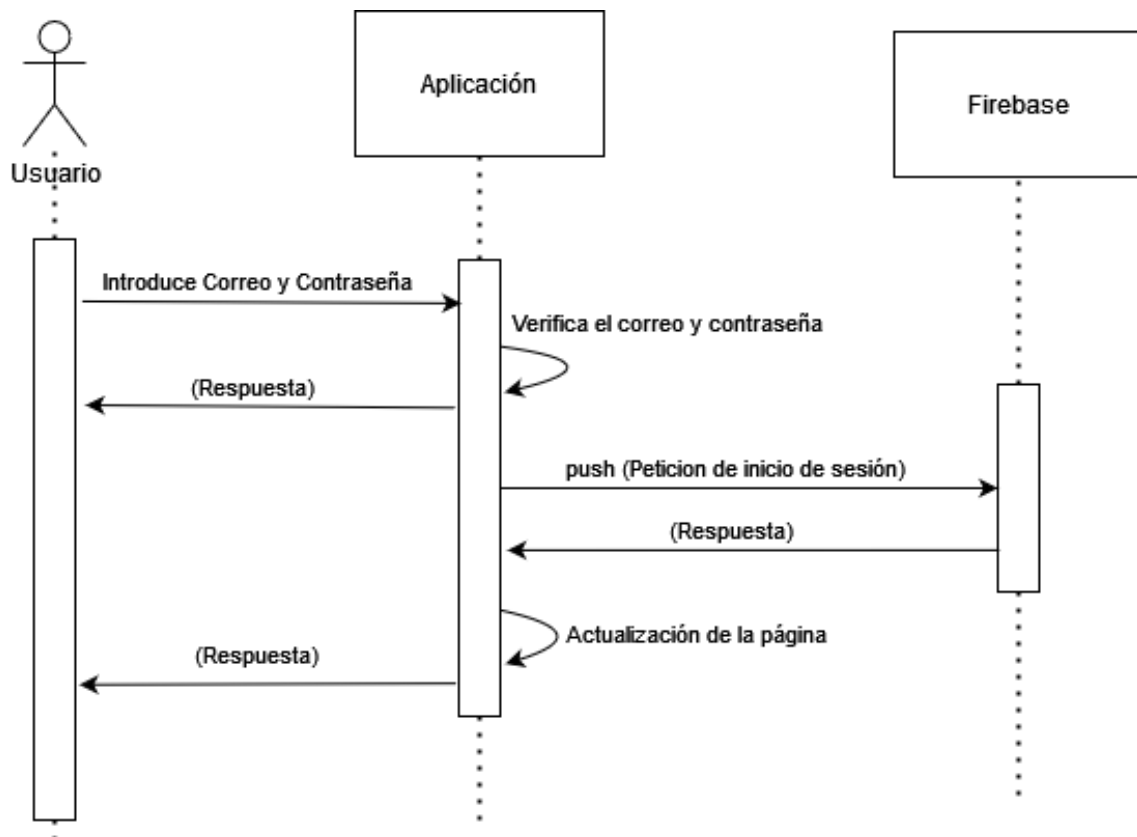


Figura 3.8: Caso de Usuario Login

Recuperar Contraseña

Esta funcionalidad necesita que el usuario indique que quiere recuperar esta contraseña, lo que provocara que la página se actualice para mostrar al usuario una interfaz con un campo para poder introducir el correo electrónico. El usuario deberá apretar un botón para solicitar esta recuperación. La página comprueba el campo de correo electrónico. El usuario recibe una respuesta si el campo está escrito incorrectamente o si se está solicitando correctamente. Entonces la aplicación envía una petición de restablecer la contraseña al servicio Firebase el cual dará una respuesta a la aplicación y esta será mostrada al usuario.

Todo este proceso se puede observar en la figura 3.9

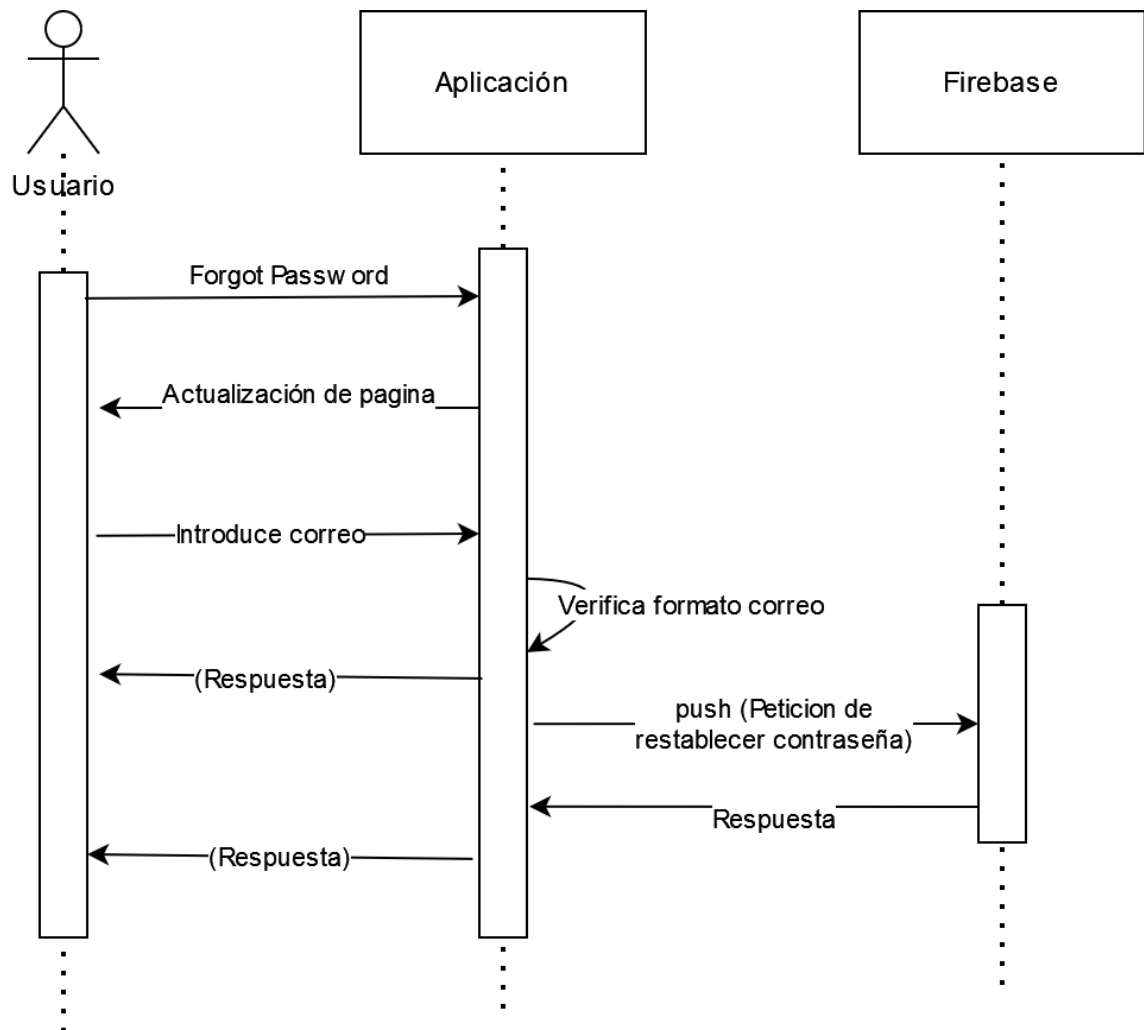


Figura 3.9: Caso de Usuario Forgot Password

Seleccionar SmartContract

Esta funcionalidad aparece una vez el usuario inicia sesión. Primero de todo la aplicación hace un trabajo previo. Este consiste en:

1. Conseguir la clave privada de blockchain guardada en Firebase.
2. Crear y guardar de forma segura las claves generadas a partir de la clave privada obtenida.
3. Solicitar las direcciones de los contratos guardados del usuario actual.
4. Recibir estos datos a través de Firebase

Una vez obtenidos estos datos, la aplicación muestra una lista de todos los contratos que tenga el usuario actual. Interactuando con estos haciendo clic y la aplicación actualizará la página, dependiendo de la acción del usuario.

Este proceso se puede observar en la figura 3.10

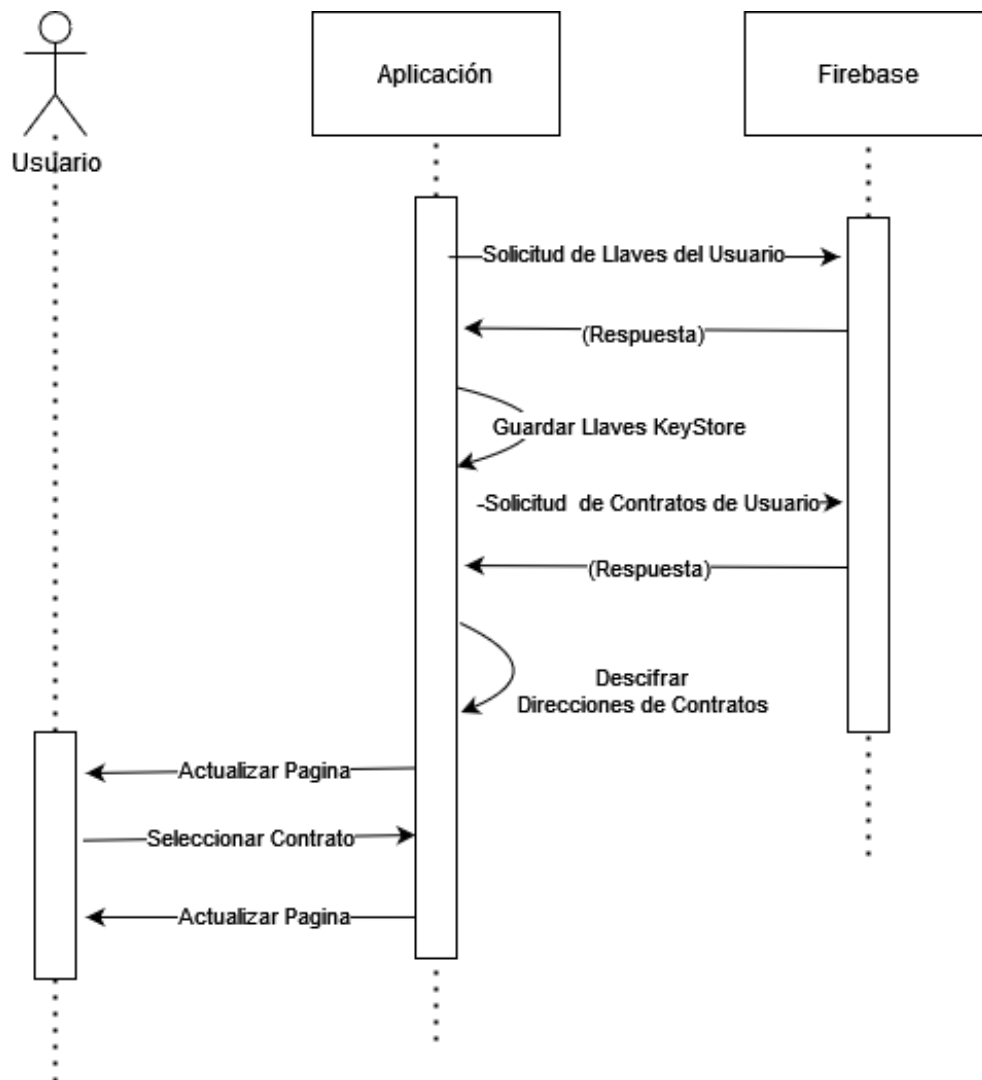


Figura 3.10: Caso de Usuario Seleccionar Smart contract

Seleccionar Opción

Esta funcionalidad aparece una vez el usuario ha elegido un Smart contract. Primero de todo la aplicación genera las posibles opciones que puede hacer, generalizando estas funciones podrán ser los casos de uso, de obtener información o modificar información.

Estas opciones serán mostradas al usuario, el cual podrá seleccionar una de estas. Entonces se enviará la correspondiente información al caso seleccionado y la aplicación actualizará la página del usuario siguiendo la acción del usuario.

Este proceso se puede observar en la figura 3.11

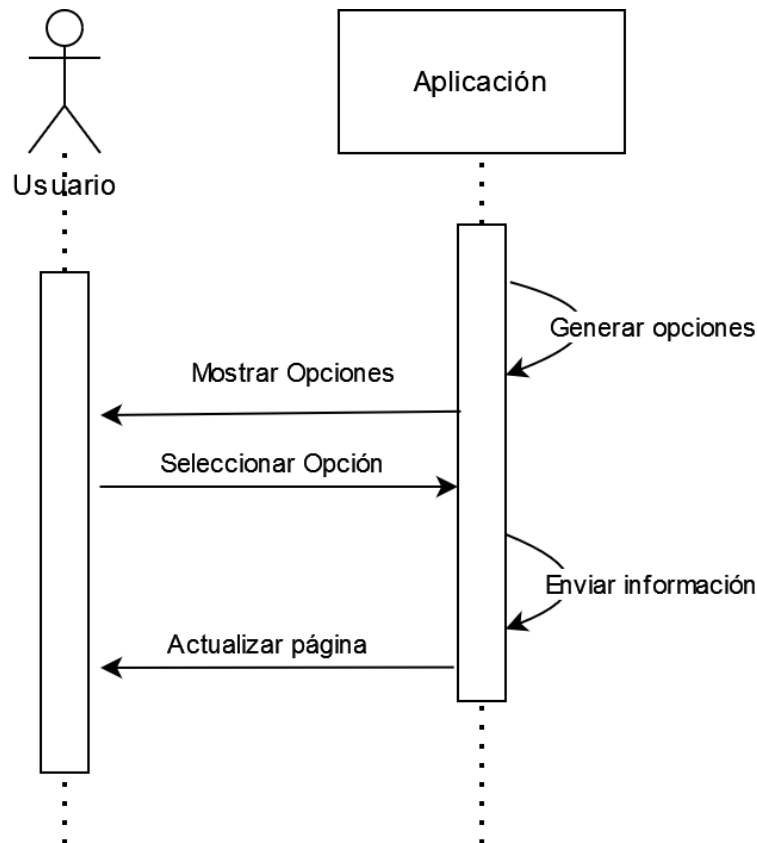


Figura 3.11: Caso de Usuario Seleccionar Opción

Obtener Información

Esta funcionalidad aparece una vez el usuario ha elegido la opción información. Primero de todo la aplicación hará un trabajo previo. Este consiste en:

1. Obtener la información del caso anterior.
2. Hacer una solicitud a la blockchain del Smart contract solicitado
3. A partir de la respuesta obtener la información del contrato
4. Mostrar la información del contrato al usuario

Este proceso es el mismo para obtener información de una transacción de la blockchain y seguirá el siguiente comportamiento [3.12](#)

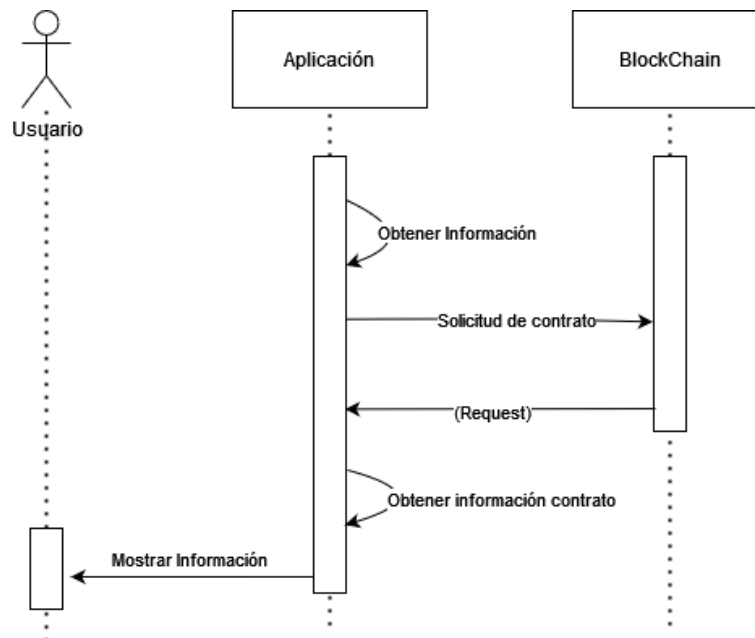


Figura 3.12: Caso de Usuario Obtener Información

Modificar Información

Esta funcionalidad aparece una vez el usuario ha elegido la opción de modificación de información. Primero de todo la aplicación hace un trabajo previo. Este consisten en:

1. Obtener la información del caso anterior.
2. Hacer una solicitud a la blockchain del Smart contract solicitado
3. A partir de la respuesta obtener la información del contrato
4. Mostrar la información del contrato al usuario

Hay dos posibles casos de modificaciones y variaran la interfaz que se muestra el usuario. Estos son:

- Información del contrato: Se muestra al usuario un campo el cual podrá editar y cambiar esta información y un botón para mandar esa solicitud.
- Consentimiento del contrato: Se muestra al usuario un campo el cual podrá elegir un botón, uno para garantizar el consentimiento y otro para denegarlo, enviando la solicitud correspondiente.

Una vez enviada la solicitud la aplicación responde con una solicitud de comprobación biométrica. Si recibe una respuesta positiva la aplicación solicitara la modificación a la blockchain y se hará una actualización de la página.

Este proceso se puede observar en la figura 3.13

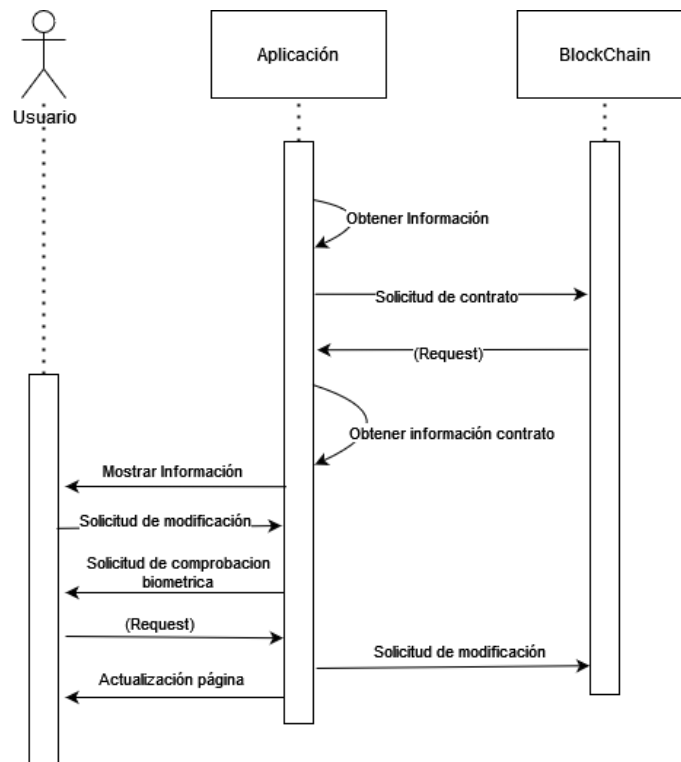


Figura 3.13: Caso de Usuario para modificar Información

4 Implementación

En este apartado se estructuran las diferentes clases, métodos y datos que utiliza cada estructura del sistema y como estas se comunican. Para las dos aplicaciones Android y Java necesitaremos una Ethereum blockchain, para ello utilizaremos la aplicación Ganache Truffle Suite, la cual permite hacer una Ethereum blockchain en una red local para hacer pruebas.

4.1 Aplicación Java

Esta aplicación sirve para generar el entorno de trabajo para nuestra aplicación Android. Este entorno sera creado para cumplir las propuestas mencionadas en el apartado 1 del trabajo. ([18] [7]) Se crearan los contratos inteligentes mencionados en el apartado de tecnologías. Estos se subiran a una blockchain local, haciendo una simulación de estos acuerdos. Para ello, se crearan las llaves de los usuarios, para que puedan acceder a sus contratos inteligentes posteriormente.

Para la siguiente aplicación se ha extraído el código del github BC GDPR-Compliant PDManagementSystem [7].

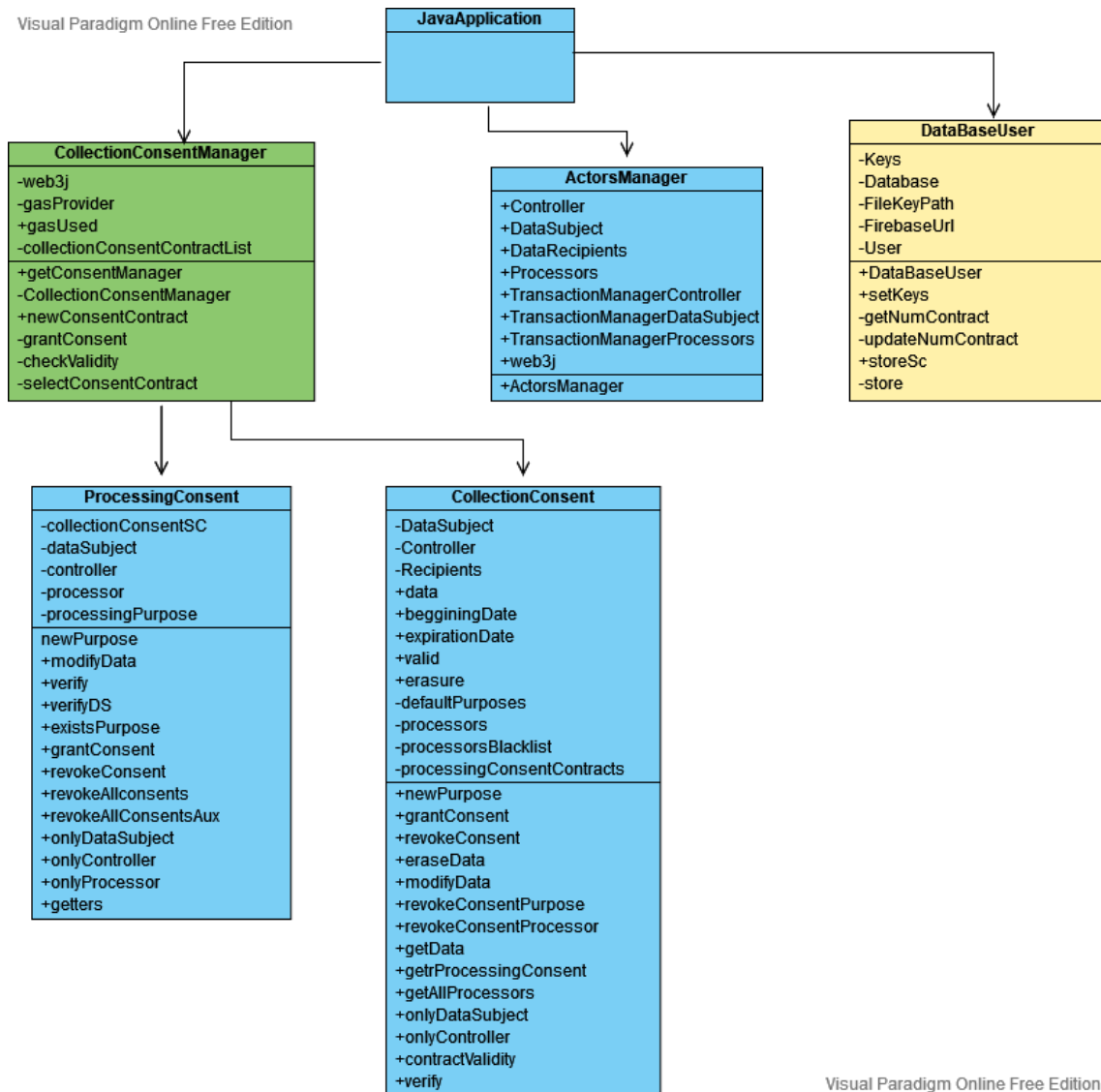


Figura 4.1: Diagrama de Clases Java Application

Como se puede observar en la figura 4.1 esta aplicación utiliza varias clases para lograr su objetivo, generar Smart contracts y guardar su información junto a las llaves en el cloud. Como se puede observar en la figura hay una clase en color amarillo, esto indica que esta clase se ha creado para añadir las funcionalidades del cloud a la aplicación (DataBaseUser). Por otra parte, tenemos la clase de color verde, la cual pertenece a una clase base pero ha sido modificada para guardar algunos de sus contenidos en el cloud. Las clases azules son las clases base de los Smart contract.

Antes de ejecutarse el programa principal se requiere que se abra el programa Ganache Truffle Suite. Con el entorno ya preparado y configurado, esto significa que nuestra aplicación java tiene los campos necesarios para establecer una conexión con la blockchain a partir de un puerto de la red local.

Empezando por la clase Main destacaremos las primeras funcionalidades.

1. Iniciar la conexión con la blockchain de Ganache.
2. Recoger los usuarios de Ganache que utilizaremos para hacer pruebas.

3. Establecer una conexión con la base de datos de Firestore.
4. Mostrar al usuario un menú que permita la generación de Smart contracts.

La librería web3j inicia la conexión y solicitar todas las cuentas Ethereum que Ganache nos proporciona.

La clase DataBaseUser establece conexión con nuestra base de datos. Esta clase creará las claves para ethereum gracias a una clase proporcionada por una librería llamada Credentials, proporcionada por web3j. Esta genera las llaves a partir de la clave privada del usuario. Estas clave privada del usuario será guardada en el cloud del Firebase para poderla usar durante la ejecución del programa. Por otro lado, las claves generadas a partir de la clase Credentials, serán las que se usaran para poder crear los contratos inteligentes.

Una vez el usuario solicite la creación de un Smart contract, el programa principal utiliza la clase CollectionConsentManager, previamente inicializada con los parámetros del web3j. Esta será la encargada de generar los Smart contract.

A continuación la clase CollectionConsentManager llama a la función newConsentContract, la cual establece los atributos y utilizara la función deploy en el contrato para subir este a la blockchain.

Una vez subido se guarda la dirección del contrato en la base de datos de Firebase gracias a la clase DataBaseUser.

4.2 Aplicación Android

El desarrollo de la aplicación Android se ha llevado a cabo utilizando la herramienta Android Studio.

Todos los componentes de la aplicación están representados en la figura 4.2

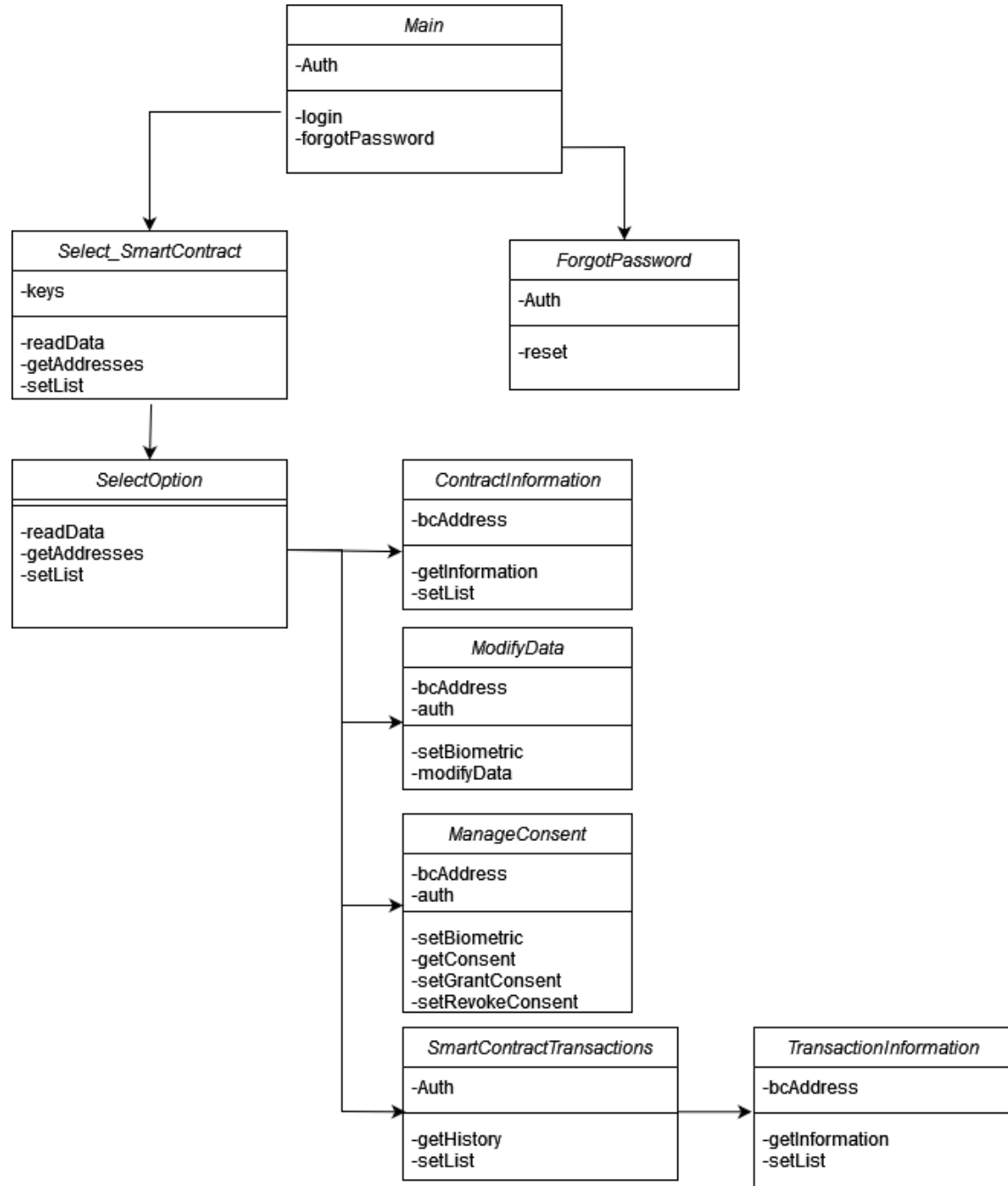


Figura 4.2: Diagrama de Clases Android Application

Todas las clases que podemos observar son actividades de la aplicación, estas son estados que representan la aplicación, que variaran según el comportamiento del usuario.

Para el uso principal de esta aplicación se necesita ser un usuario del sistema y haberse identificado en el proceso de login. Hay dos actividades que para realizar sus

funciones se necesitara una autenticación biométrica, estas serán las funcionalidades que cambien algún aspecto del Smart contract en la blockchain.

La primera clase que podemos encontrar es la Main activity. Esta tendrá dos objetivos:

- Mostrar al usuario una interfaz de inicio de sesión.
- Mostrar al usuario una interfaz para recuperar la contraseña.

La funcionalidad de añadir sesión permite obtener los datos que el usuario ha introducido y validar los campos, ya sea verificando que estén rellenos o verificando que el correo electrónico tenga el formato correspondiente. Si los campos no son correctos, se emite un mensaje de error para el usuario.

Una vez que los campos estén rellenos correctamente, se enviara una solicitud a la base de datos de usuarios de Firebase, la cual aceptara el inicio de sesión o mostrara un mensaje de error para el usuario.

La aplicación continuara con la clase SelectSmartContracts, la cual tiene como objetivo obtener las direcciones de los Smart contracts del usuarios. Estos son mostrados en una lista en la nueva actividad del usuario, mientras la aplicación carga la llave necesaria para conectarse a la blockchain.

Una vez el usuario seleccione uno de los contratos, esta información es enviada a la actividad llamada SelectOption. Esta actividad mostrará el contrato elegido y un menú con las siguientes posibles opciones:

1. Obtener información del contrato.
2. Modificar los datos.
3. Gestionar el consentimiento del contrato.
4. Mirar el historial de transacciones del contrato.
5. Seleccionar otro contrato.

Para todas estas opciones se tiene que haber generado previamente los contratos gracias a la aplicación Java, previamente descrita, y tener la blockchain de Ganache abierta.

La actividad SelectOption es la encargada de enviar la dirección de la blockchain junto a la información elegida por el usuario para poder realizar las funciones solicitadas.

A continuación se detallan cada una de estas actividades. Cada una de ellas obtiene la información enviada por SelectOption para poderse conectar a la blockchain y la información necesaria para solicitar cada una de las funciones. Todas las actividades se conectarán a la blockchain una vez iniciadas.

Obtener información del contrato

Esta actividad permite al usuario ver la información del contrato seleccionado. Con la conexión a la blockchain se obtendrá el contrato a través de una solicitud, utilizando la implementación del Smart contract Collection Consent. Una vez cargado el Smart contract en local, se pedirán varios campos de información y estos serán mostrados al usuario.

Modificar la información del contrato

Esta actividad permite al usuario ver la información guardada en el Smart Contract y modificarla, si el usuario lo desea.

Para esta actividad se ve un campo con la información anterior, y un campo donde el usuario introduce la nueva información. A través de un botón el usuario solicita la modificación del Smart Contract, lo que provoca que la aplicación pida una autenticación extra, en este caso una autenticación biométrica, ya sea huella dactilar o un reconocimiento facial, esto depende del dispositivo móvil que se esté usando. Si la autenticación biométrica tiene éxito gracias al contrato, se hace una solicitud a la blockchain y se efectúa el cambio, actualizando la actividad mostrando los nuevos datos escritos por el usuario.

Gestionar el consentimiento del contrato

Esta actividad permite al usuario ver el consentimiento actual del contrato y poder modificarlo.

Para esta actividad se verá un campo con la validez actual del contrato, y dos botones que permitirán al usuario dar o quitar el consentimiento a ese Smart contract.

Una vez el usuario solicite una modificación, la aplicación pide una autenticación biométrica, depende del dispositivo móvil utilizado. Si esta tiene éxito gracias al contrato, se hace una solicitud a la blockchain y se efectúa el cambio de consentimiento en el contrato. Actualizando la página para renovar el estado de validez del contrato.

Obtener el historial del contrato

Esta actividad tiene como objetivo conseguir todas las transacciones efectuadas con el contrato.

Gracias a la blockchain se pueden conseguir todas las transacciones que hayan tenido relación con la dirección del contrato. Una vez obtenidas estas son mostradas al usuario, el cual podrá interactuar con ellas. Esto permite al usuario ver la información de cada una de ellas, como por ejemplo qué acción es la que se ha solicitado.

5 Juego de pruebas

5.1 Test de proceso previo

Primero de todo se utiliza Ganache para poder tener una blockchain local, en la figura se puede ver la configuración.

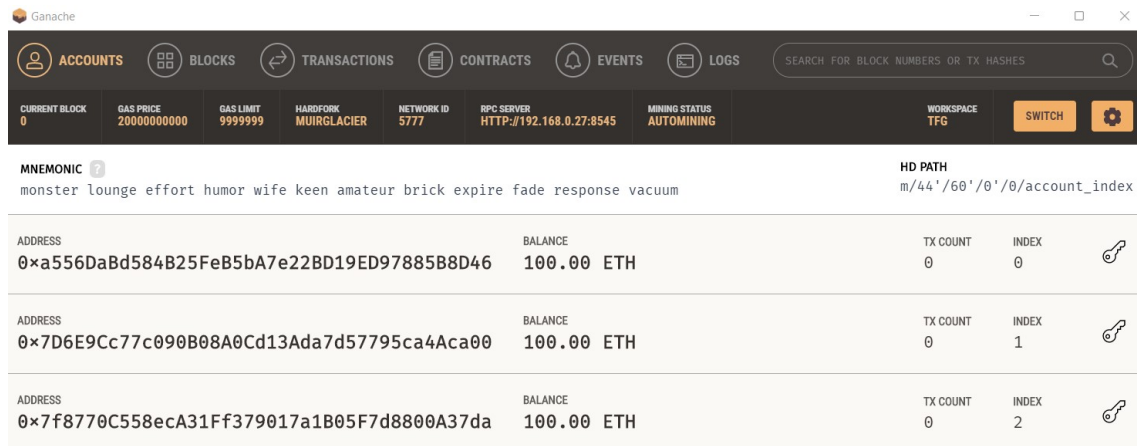


Figura 5.1: Configuración Ganache

Utilizaremos la siguiente cuenta como usuario principal de nuestra aplicación.

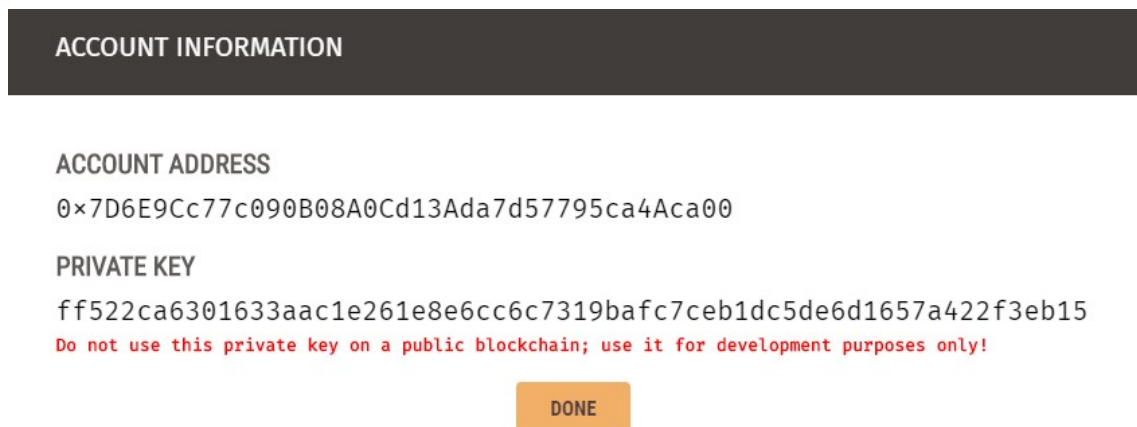


Figura 5.2: Usuario utilizado para las pruebas

Abriendo la aplicación java podremos generar un contrato inteligente (Figura: 5.3), este es subido a la blockchain (Figura: 5.4) y también se guarda la dirección de este contrato en el cloud de FireBase (Figura: 5.5). Además se guarda en el cloud la llave privada del usuario de la blockchain.(Figura: 5.6)

```

1. Deploy a new SC
2. Exit
1
Consent Contract Created
Transaction Hash: 0xe6187bdc8d8ccad9b097eb4075ed020503a94ed3ff7d6c1671fb552cb96e688d
Contract address: 0x3b14a3640e59c43a4fd4f7e50e95617a9faa4ee1
Write result: com.google.cloud.firestore.WriteResult@35c786e7
1. Deploy a new SC
2. Exit

```

Figura 5.3: Generación contrato aplicación Java

TX 0xe6187bdc8d8ccad9b097eb4075ed020503a94ed3ff7d6c1671fb552cb96e688d				
<div>← BACK</div>				
SENDER ADDRESS	CREATED CONTRACT ADDRESS			
0x7D6E9Cc77c090B08A0Cd13Ada7d57795ca4Aca00	0x3B14a3640E59c43A4FD4f7E50E95617A9FAa4EE1			
VALUE	GAS USED	GAS PRICE	GAS LIMIT	MINED IN BLOCK
0.00 ETH	3239226	22000000000	4300000	1

Figura 5.4: Creacion de contrato en Ganache

```

▼ SmartContracts
0 "0x3b14a3640e59c43a4fd4f7e50e95617a9faa4ee1"
nSmartContracts: 1

```

Figura 5.5: Comprobación dirección en el cloud

```
Private: "ff522ca6301633aac1e261e8e6cc6c7319bafc7ceb1dc5de6d1657a422f3"
```

Figura 5.6: Comprobación llave privada en el cloud

5.2 Test de aplicación Android

En la siguiente figura 5.1 se puede observar que la aplicación recibe el nombre provisional de SCManager, y instalada en un dispositivo Android lista para su uso.

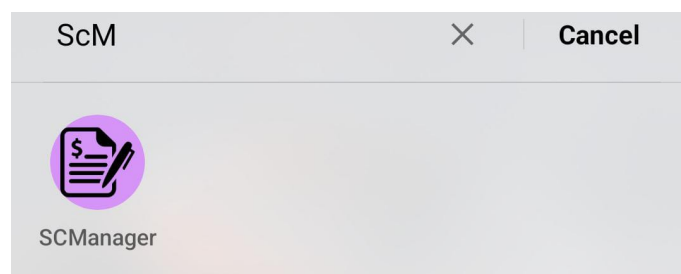


Figura 5.1: Icono de la aplicación en un dispositivo Android

Al iniciar la aplicación podremos ver la interfaz de inicio de sesión que también permite recuperar la contraseña con el texto Forgot Password, como se puede observar en la figura 5.2

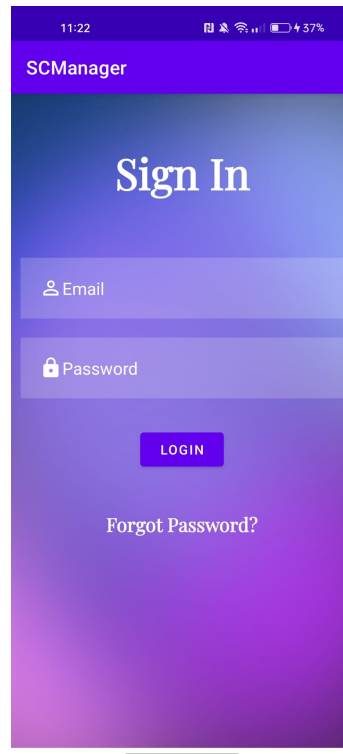


Figura 5.2: Inicio de la aplicación

5.3 Recuperar la contraseña

Si el usuario no recordase la contraseña i hace click en el texto "Forgot password", se cambiara a la siguiente actividad para recuperar la contraseña como se puede ver en la figura 5.3

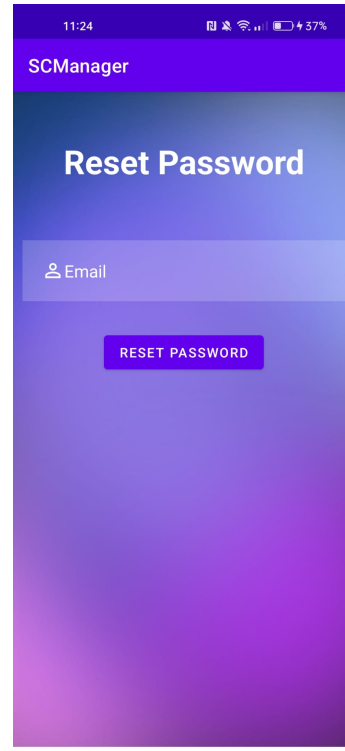


Figura 5.3: Inicio de la aplicación

Hay tres posibles casos de error:

1. El correo electrónico no tiene el formato correcto. (Figura 5.3a)
2. El campo del correo electrónico está vacío. (Figura 5.3b)
3. El usuario no está registrado en la aplicación. (Figura 5.3c)

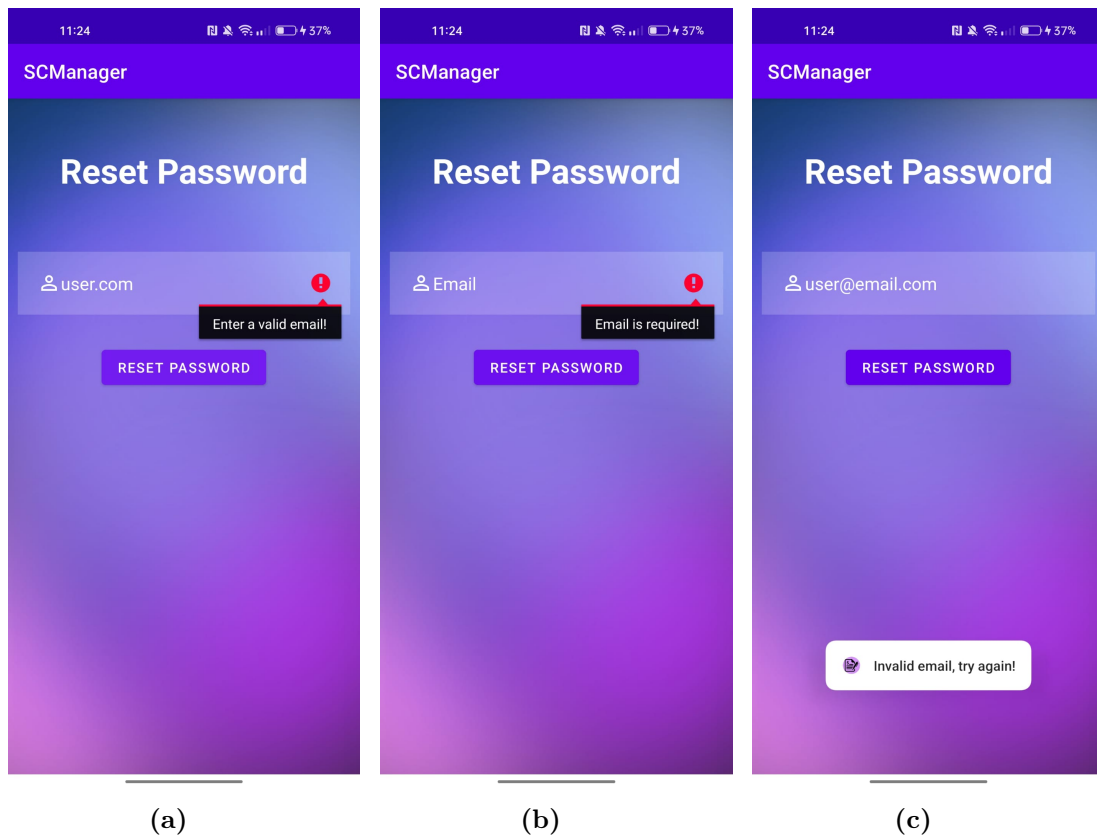


Figura 5.3: Ejemplos de errores en el proceso de Recuperar Contraseña

Si el usuario está registrado se notificara al usuario que se ha enviado un correo electrónico como se puede observar en la figura 5.3d

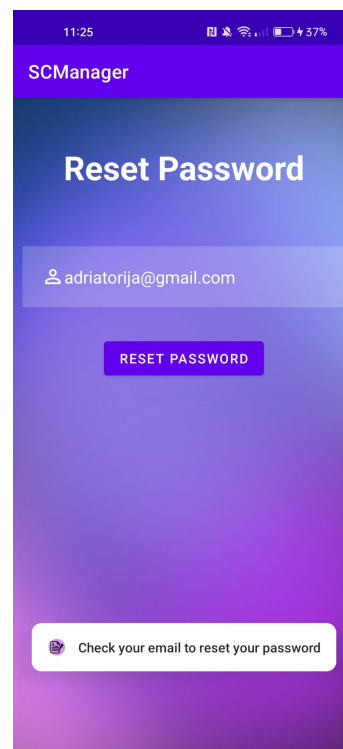


Figura 5.3d: Función de Recuperar Contraseña Válida

Si se comprueba el correo electrónico se observa que hemos recibido el siguiente mensaje y podemos obtener el cambio de contraseña gracias al enlace recibido como se puede observar en las figuras 5.4a y 5.4b

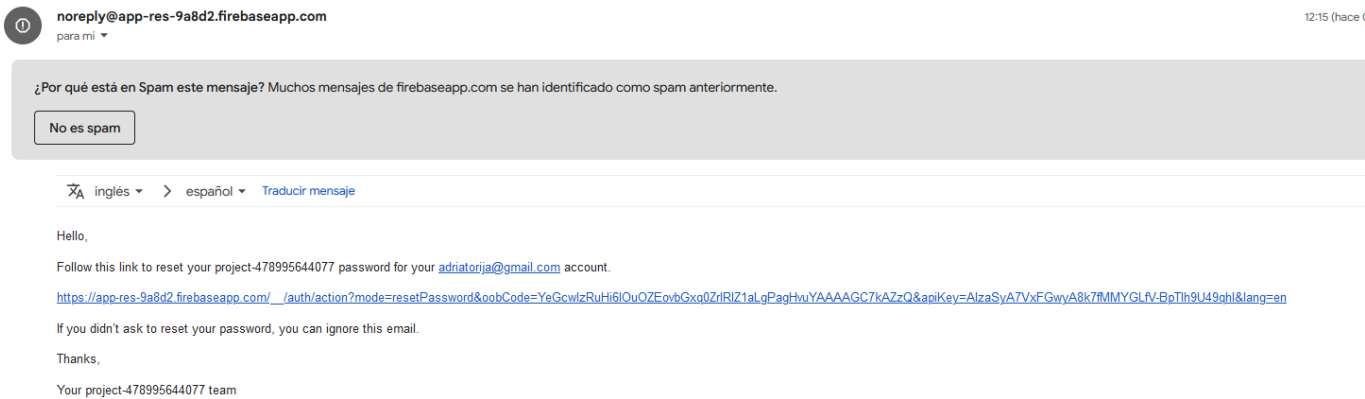


Figura 5.4a: Correo electrónico de recuperar contraseña

Reset your password

for **adriatorija@gmail.com**

New password



SAVE

Figura 5.4b: Enlace de recuperar contraseña

5.4 Inicio de sesión

En el apartado de inicio de sesión encontraremos 4 posibles errores a la hora de iniciar sesión. Estos pueden ser:

1. El campo del correo electrónico está vacío. (Figura 5.5a)
2. El correo electrónico no tiene el formato correcto. (Figura 5.5b)
3. El campo de contraseña está vacío (Figura 5.6a)
4. El usuario no está registrado en la aplicación. (Figura 5.6b)

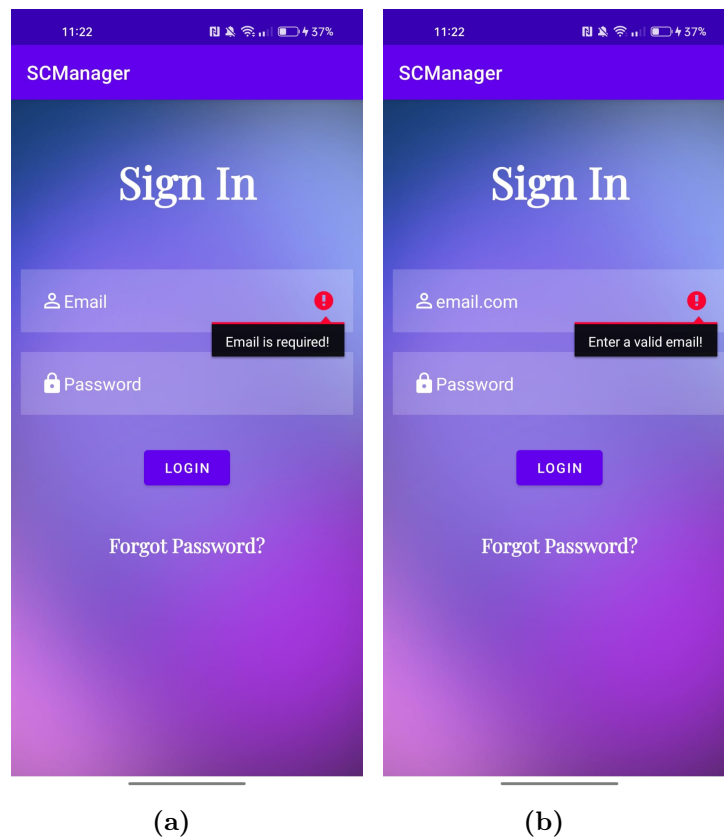


Figura 5.5: Ejemplos de errores en el proceso de Recuperar Contraseña

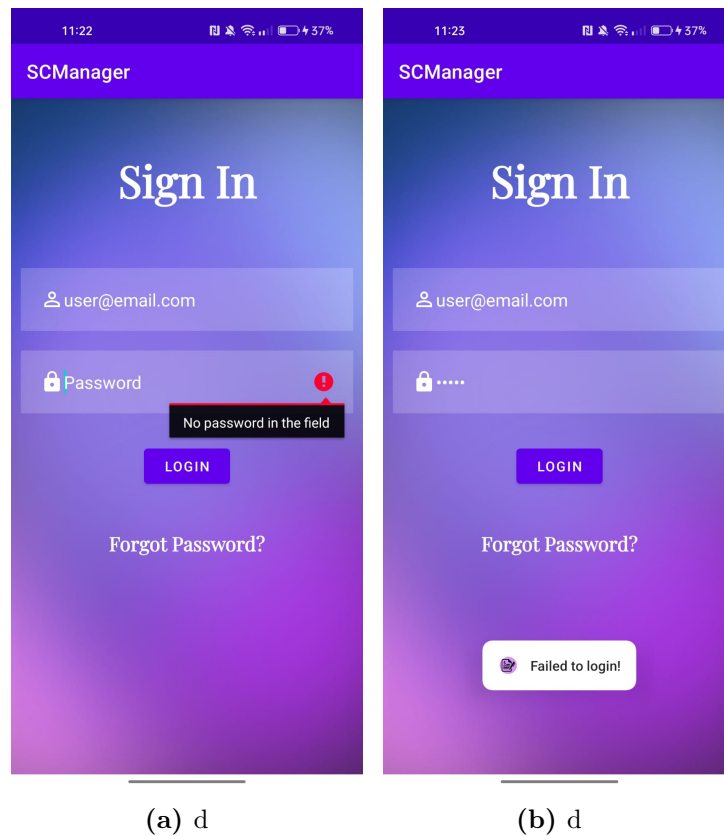


Figura 5.6: Ejemplos de errores en el proceso de Recuperar Contraseña

Si el usuario es correcto se empieza a cargar su información, primero las claves y después los contratos. A continuación se obtendrían los contratos inteligentes y guardaría las direcciones de estos en una lista en el backend. Esto sera representado a partir de una barra de progreso como se puede observar en la figura 5.7

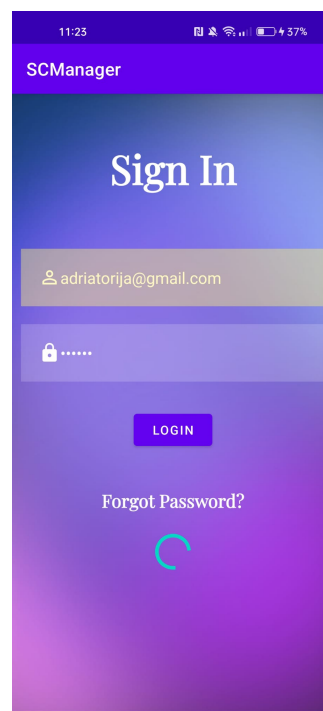


Figura 5.7: Función de Inicio de sesión válido

5.5 Select Smart contract

Una vez cargado el contenido se cambia de actividad (Select SmartContract) para poder mostrarle al usuario una lista de contratos ordenados por números. Cada uno tiene su dirección en el backend de la aplicación y cuando el usuario escoja uno esta información es enviada a la siguiente actividad. Aquí se puede ver la representación de un usuario con dos Smart Contracts. (Figura 5.8)

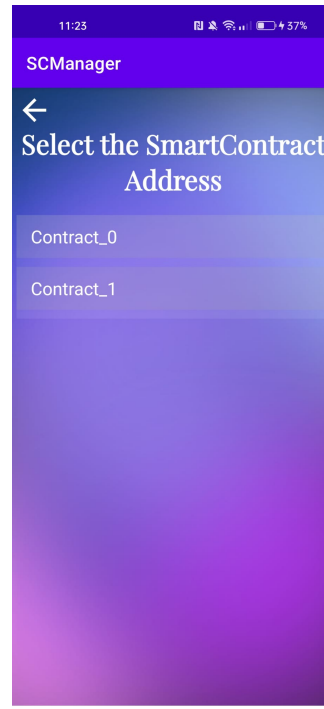


Figura 5.8: Función de Seleccionar Smart contract

5.6 Select Option

Si el usuario selecciona un SmartContract se cambia a la actividad SelectOption, esta ofrece varias posibilidades para poder interactuar con el contrato seleccionado como se puede ver en la figura 5.10.

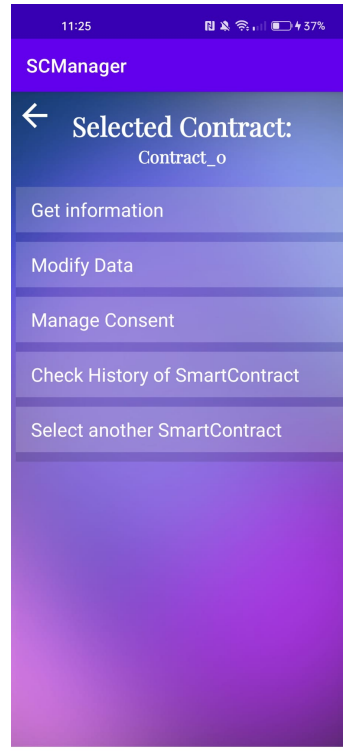


Figura 5.9: Función de Seleccionar Opción

Ahora entraremos en detalle en cada una de las implementaciones de estas opciones:

5.6.1 *Obtener información contrato*

Si el usuario selecciona la opción de obtener información del contrato se debera de mostrar por pantalla. Esta información consistirá en:

- Dirección del contrato
- Validez del contrato
- Información del contrato

El resultado se puede observar en la figura 5.10.

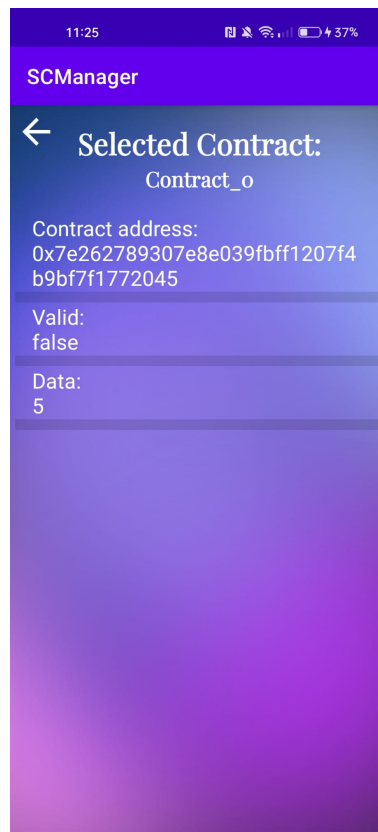


Figura 5.10: Función de Obtener información

5.6.2 Modificar información

Si el usuario selecciona la opción de modificar información, se tiene que mostrar por pantalla la actual información y los campo para poder escribir la nueva información conjunto con un botón para hacer la solicitud. El resultado se puede ver en la figura 5.10

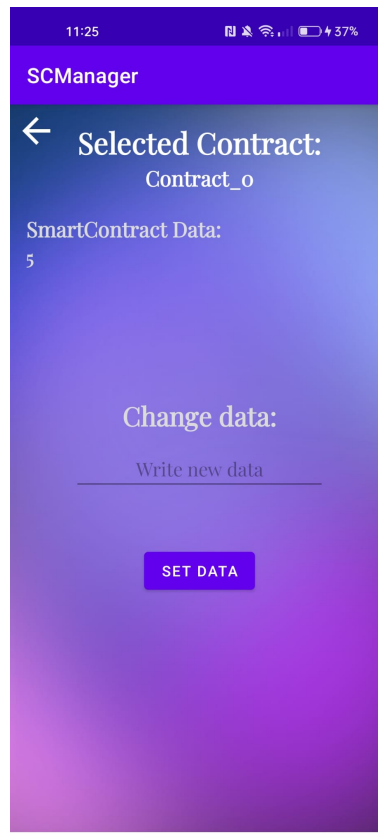


Figura 5.10: Función de Modificar Información

Para hacer la prueba de esta función se modificar la información actual por unos valores nuevos. Una vez iniciada la solicitud (Figura 1) se mostrara para poder ser autenticado con la huella (Figura 2) si la autenticación es correcta se actualizara la pagina y se podrá ver la nueva información. (Figura 3)

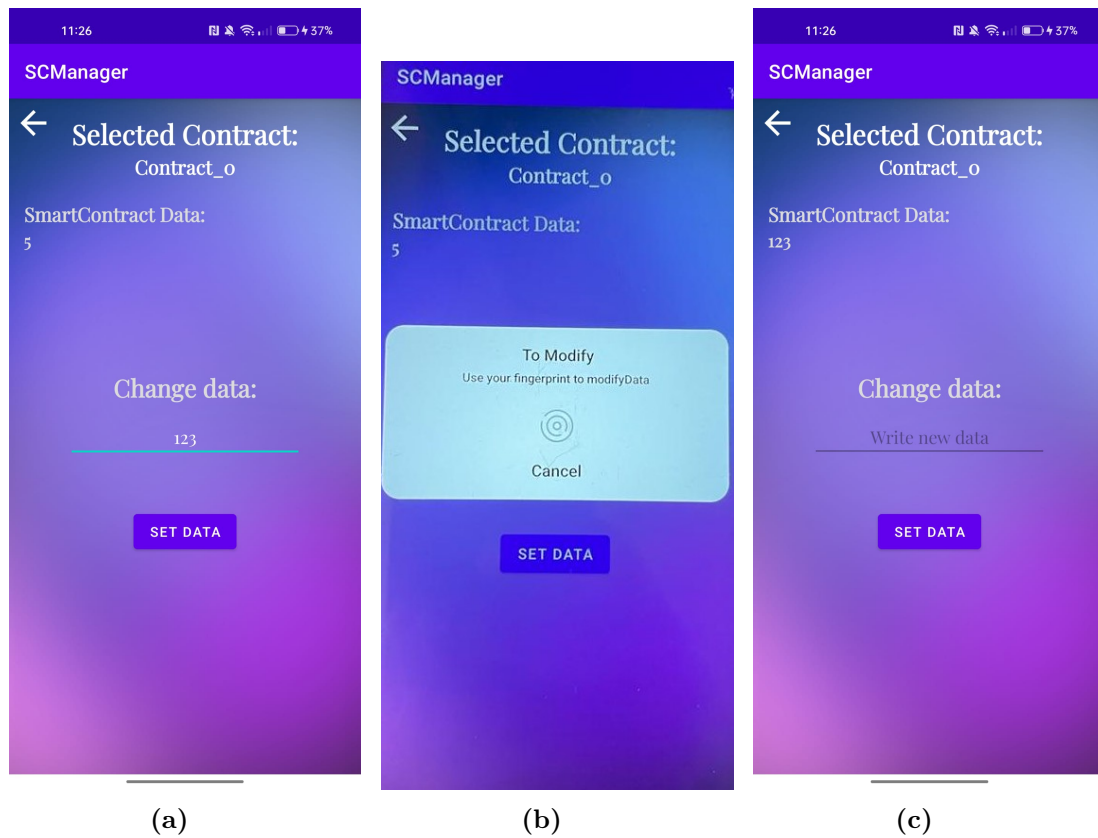


Figura 5.11: Ejemplo de modificar información

5.6.3 Modificar Consentimiento

Si el usuario selecciona la opción de modificar consentimiento, se tiene que mostrar por pantalla el actual consentimiento del contrato, si es valido o no. En la interfaz deben aparecer dos botones para poder modificar este campo como se puede ver en la figura 5.11.

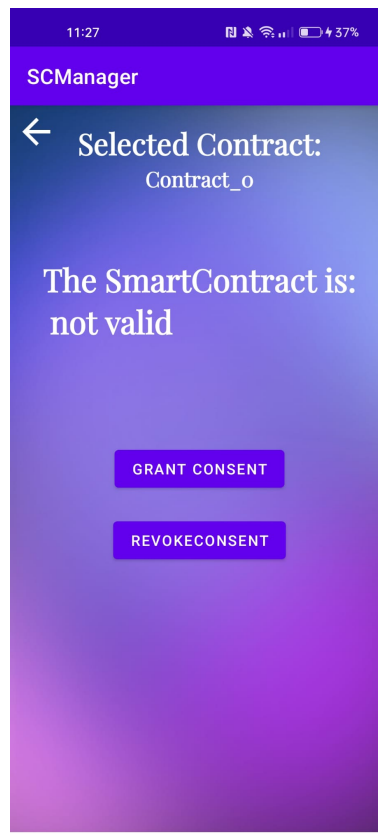


Figura 5.11: Función de Modificar Consentimiento

Para hacer la prueba de esta modificación se prueba de cambiar el consentimiento actual por un consentimiento valido o invalido, dependiendo el estado del actual. Primero veremos el estado actual del contrato (Figura 1) Se muestra para poder ser autenticado con la huella (Figura 2) y si esta autenticación es correcta se actualiza la página viendo el nuevo consentimiento del contrato. (Figura 3)

Test Grant consent:

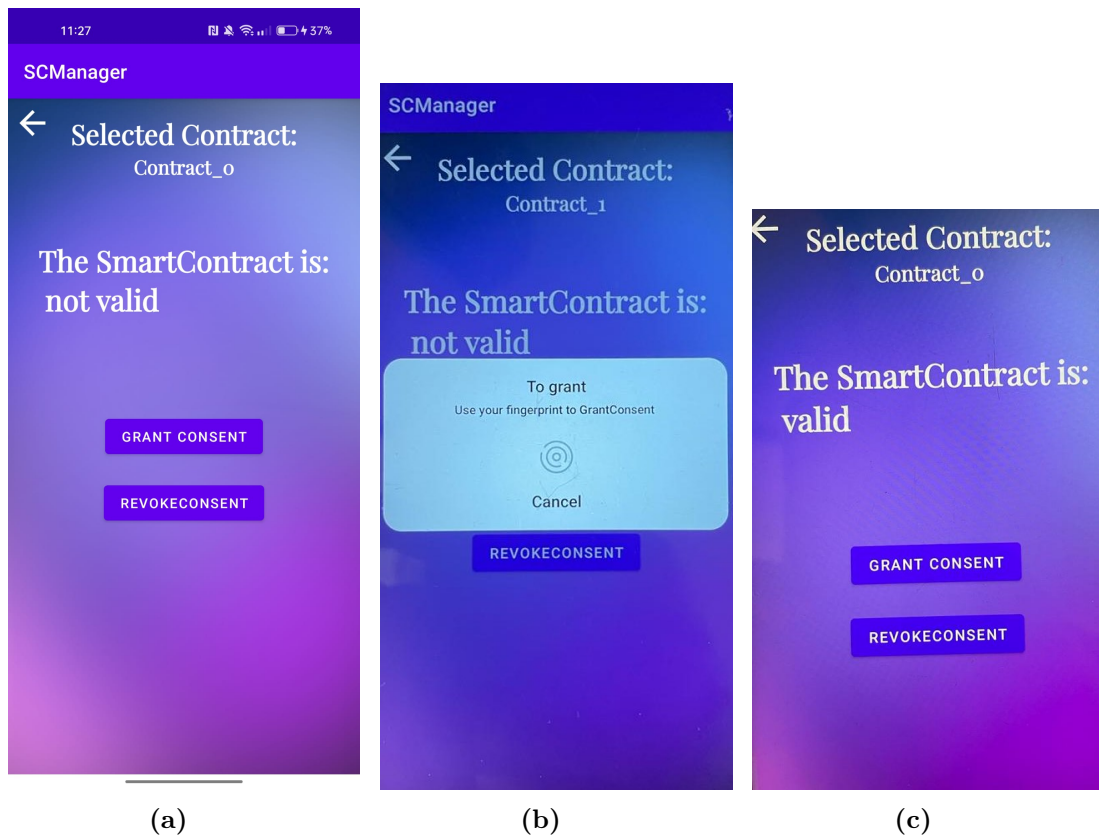


Figura 5.14: Ejemplo de modificar información

Test Revoke consent:

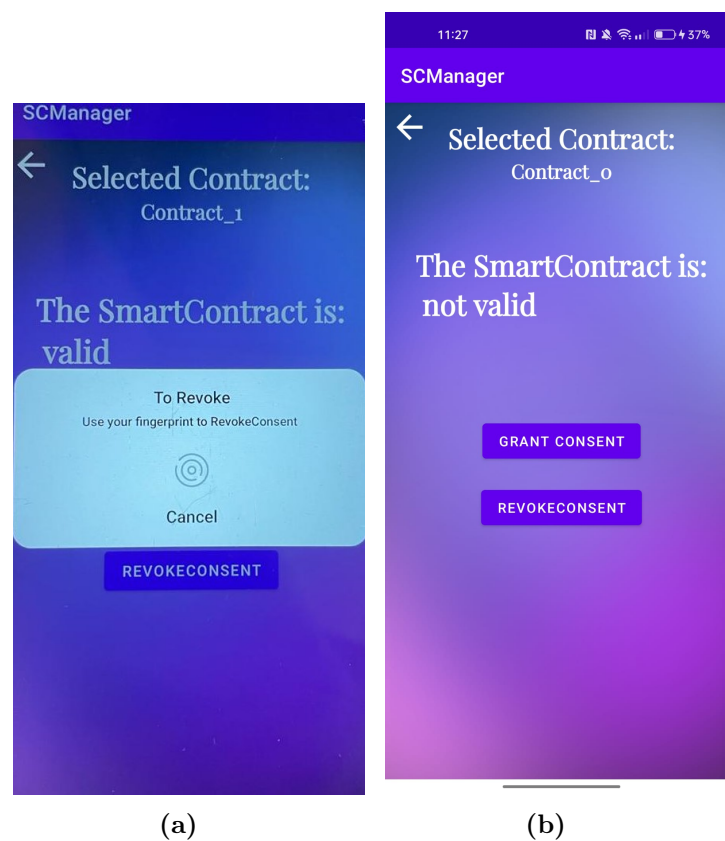


Figura 5.14: Ejemplo de modificar información

5.6.4 Comprobar el historial del contrato

Si el usuario selecciona la opción de comprobar el historial del contrato, se tiene que mostrar por pantalla un listado de todas las transacciones producidas, estas están ordenadas numéricamente como se puede observar en la figura 5.14

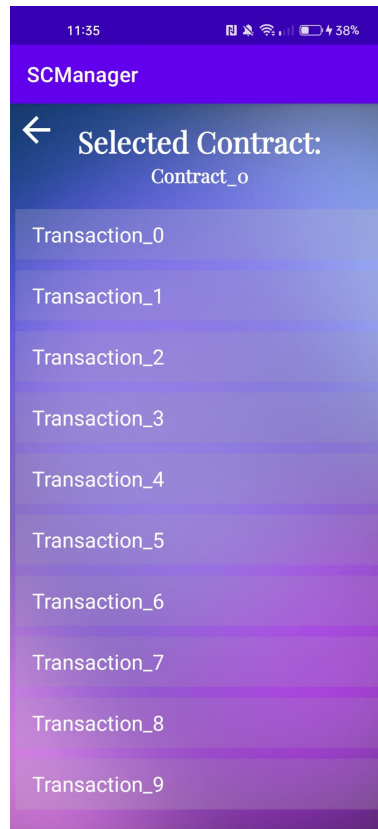


Figura 5.14: Función de Comprobar Historial del Contrato

Si el usuario selecciona una transacción se tiene que mostrar por pantalla la información de esta. Esta información consiste en:

- Hash de la transacción.
- Dirección del contrato.
- Bloque en el que se encuentra la transacción.
- Quien ha hecho la transacción.
- Que acción ha sido realizada.

El resultado se podrá observar en la figura 5.14.

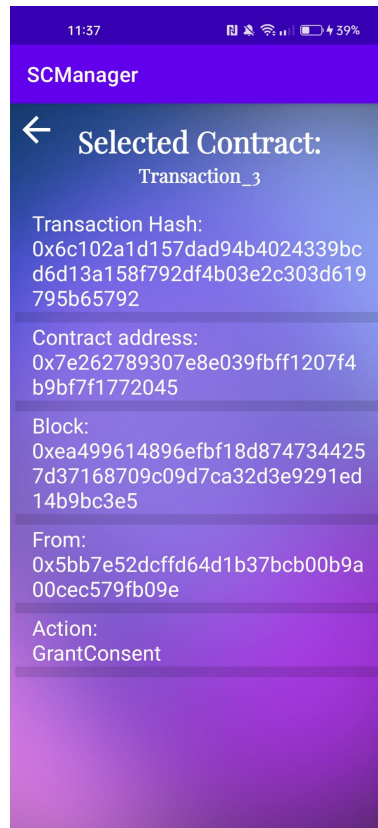


Figura 5.14: Función de obtener Información Transacción

6 Conclusiones

En la propuesta [6] se presenta un esquema que permite conocer y gestionar los consentimientos para la recolección de sus datos. En la propuesta es necesario que se incluya una aplicación segura y fácil de utilizar por parte de los usuarios. En este proyecto se ha dado solución a una de las principales carencias de esta propuesta: el control por parte de los usuarios de estos contratos inteligentes generados.

Se ha creado una aplicación que nos permite autenticar a un usuario, y obtener todos los contratos que se hayan generado en la blockchain. A partir de estos el usuario tiene las funcionalidades de leer la información de estos, modificar esa información, ya sea la propia información del contrato o el consentimiento de este. Además se permite al usuario ver todas las transacciones realizadas en el contrato que el usuario desee.

Para testear esta aplicación, se ha desarrollado una infraestructura que consiste en una blockchain local; y una aplicación para realizar el deploy de los contratos inteligentes en la blockchain y guardar sus direcciones en el cloud. Además, usamos dicho cloud como medio de almacenaje de las claves usadas por los usuarios para poder hacer la interacción con los contratos inteligentes en la blockchain. Pudiendo volver a ver la información y hacer modificaciones desde cualquier dispositivo móvil, después de la creación de estos contratos.

Este trabajo ha tenido algunos puntos complicados, el principal problema ha sido que cada fase de este proyecto era dependiente de la anterior, por lo tanto, se necesitaba hacer las fases en orden y correctamente para no llevar los errores a las siguientes fases. La mayor dificultad en este proceso ha sido el hecho de obtener los contratos de la blockchain en la aplicación Android, la fase más importante para el correcto funcionamiento de la aplicación Android.

6.1 Trabajo Futuro

La principal tarea que se debería realizar es buscar una alternativa a Firebase. En este trabajo se ha buscado la idea de la descentralización, con la utilización de la blockchain, pero utilizando Firebase hace que haya un administrador y distribuidor. Esto se ha hecho para tener un mayor control del sistema pero en una versión futura también se debería descentralizar la gestión de usuarios.

Esta aplicación está diseñada solo para dispositivos Android, se debería crear una implementación para dispositivos iOS para poder llegar a todos los usuarios de dispositivos móviles.

Referencias

- [1] BIT2ME: *Smart Contracts: ¿Qué son, cómo funcionan y qué aportan?*. – URL <https://academy.bit2me.com/que-son-los-smart-contracts/>. – Online
- [2] C. DAUDÉN-ESMEL, J. Castellà-Roca ; VIEJO, A.: "*Sistema para la gestión automática de las políticas de privacidad y uso de las cookies*", *VII Reunión Española sobre Criptología y Seguridad de la Información*. (2022)
- [3] CAI, Wei ; WANG, Zehua ; ERNST, Jason B. ; HONG, Zhen ; FENG, Chen ; LEUNG, Victor C. M.: Decentralized Applications: The Blockchain-Empowered Software System. In: *IEEE Access* 6 (2018), S. 53019–53033
- [4] CONSENSYS: *Ganache Truffle Suite*. – URL <https://trufflesuite.com/ganache/>. – Online
- [5] COUNCIL OF EUROPEAN UNION: *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)*. 2016. – <https://www.boe.es/doue/2016/119/L00001-00088.pdf>
- [6] DAUDÉN-ESMEL, Cristofol ; CASTELLÀ-ROCA, Jordi ; VIEJO, Alexandre ; DOMINGO-FERRER, Josep: Lightweight Blockchain-based Platform for GDPR-Compliant Personal Data Management, 01 2021, S. 68–73
- [7] ESMEL, Cristofol D.: *BC GDPR Compliant PDManagement System*. – URL https://github.com/toful/BC_GDPR-Compliant_PDManagement_System. – Online
- [8] FABER, Benedict ; MICHELET, Georg ; WEIDMANN, Niklas ; MUKKAMALA, Raghava R. ; VATRAPU, Ravi: BPDIMS:A Blockchain-based Personal Data and Identity Management System, 01 2019
- [9] GOOGLE: *AndroidStudio*. – URL <https://developer.android.com/studio>. – Online
- [10] GOOGLE: *Firebase*. – URL <https://firebase.google.com/>. – Online
- [11] GOOGLE: *Firestore*. – URL <https://firebase.google.com/docs/firestore>. – Online
- [12] JAVATPOINT: *History of Blockchain*. – URL <https://www.javatpoint.com/history-of-blockchain>. – Online
- [13] KASPERSKY: *Secure Element*. – URL <https://encyclopedia.kaspersky.com/glossary/secure-element/>. – Online
- [14] LABS, Web3: *Web3j*. – URL <https://docs.web3j.io/4.8.7/>. – Online

- [15] NAKAMOTO, Satoshi: *Bitcoin: A Peer-to-Peer Electronic Cash System*. – URL <https://bitcoin.org/bitcoin.pdf>. – Online
- [16] PASTOR, Javier: *Qué es blockchain: la explicación definitiva para la tecnología más de moda*. 2018. – URL <https://www.xataka.com/especiales/que-es-blockchain-la-explicacion-definitiva-para-la-tecnologia-mas-de-moda>. – Online
- [17] PROJECTS, Solidity E.: *Solidity*. – URL <https://docs.soliditylang.org>. – Online
- [18] TRUONG, Nguyen B. ; SUN, Kai ; LEE, Gyu M. ; GUO, Yike: GDPR-Compliant Personal Data Management: A Blockchain-Based Solution. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), S. 1746–1761
- [19] WOOD, Gavin u.a.: Ethereum: A secure decentralised generalised transaction ledger. In: *Ethereum project yellow paper* 151 (2014), Nr. 2014, S. 1–32