

**UNIVERSIDAD EUROPEA DE MADRID**

**ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO**

**MASTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN**

PRIMERA ENTREGA TRABAJO FIN DE MASTER

**VALIDACIÓN DE DOCUMENTOS MEDIANTE BLOCKCHAIN**

**ALBERTO RODRIGUEZ TENA**

**CURSO 2018-2019**

**TÍTULO**: VALIDACIÓN DE DOCUMENTOS MEDIANTE BLOCKCHAIN

**AUTOR**: ALBERTO RODRIGUEZ TENA

**TITULACIÓN**: MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

**DIRECTOR DEL PROYECTO**: JUAN ANTONIO PIÑUELA IZQUIERDO

**FECHA**: MAYO de 2019

# RESUMEN

Este proyecto valida documentos subidos por un usuario a la aplicación mediante el uso de la tecnología blockchain. Con el fin de obtener un documento final que este validado y que aporte fiabilidad y veracidad sobre el mismo, esto se conseguirá debido a que la tecnología blockchain dispone de características que garantiza la trazabilidad y fiabilidad de estos documentos.

A lo largo del documento se realizará un desarrollo mas detallado sobre esta tecnología y sus características, además de realizar un análisis detallado del desarrollo de la aplicación.

Este trabajo se ha elaborado mediante una aplicación web escrita en el lenguaje de programación Python utilizando el framework web de Python, Django el cual interactúa con API externa llamada Infura para comunicarse con las red blockchain. Además de otras librerías y otras tecnologías como pueden ser JS, CSS3 y HTML5. Las acciones que ocurran sobre la aplicación quedaran reflejadas en la red blockchain a la que este conectada.

# ABSTRACT

This project validates documents uploaded by a user to the application by using blockchain technology. In order to obtain a final document that is validated and that brings reliability and truthfulness on it, this will be achieved because blockchain technology has features that guarantee the traceability and reliability of these documents.

Throughout the document, a more detailed development will be made on this technology and its characteristics, in addition to conducting a detailed analysis of application development.

This work was developed using a web application written in the Python programming language using the Python web framework, Django which interacts with external APIs called Infura to communicate with blockchain network. In addition to other libraries and other technologies such as JS, CSS3 and HTML5. The actions that occur on the application will be reflected in the blockchain network to which it is connected.

# AGRADECIMIENTOS

Este trabajo pone fin a mi etapa de estudios de máster, esta ha sido una de las etapas mas importantes y enriquecedoras a lo largo de mi etapa de estudios, ha estado marcada tanto por momentos buenos como por momentos de lucha y crecimiento que hoy veo reconocidos con la entrega de este proyecto y la finalización de esta etapa. Esta etapa ha traído consigo horas de estudio esfuerzo y sacrificio, pero también me ha dejado muchas personas que han estado apoyándome a lo largo de esta dura etapa y a las cuales no me quiero olvidar de dar las gracias en este proyecto, lo primero y mas importante a mis padres por estar apoyándome y acompañándome siempre en los momentos difíciles y en cada una de las decisiones que he tomado y que sin ellos esto no hubiera sido posible dado que me han ayudado a ser quién soy ahora. También y como no mencionar a mi tutor por el esfuerzo, dedicación y constancia que ha tenido conmigo en la realización de este proyecto. No olvidarme además de mis compañeros de promoción a los cuales agradecer que hayan recorrido conmigo este año lleno experiencias positivas y crecimiento personal.

Índice

[RESUMEN 4](#_Toc12995663)

[ABSTRACT 5](#_Toc12995664)

[AGRADECIMIENTOS 6](#_Toc12995665)

[Capítulo 1. INTRODUCCIÓN 10](#_Toc12995666)

[1.1 Planteamiento del problema 10](#_Toc12995667)

[1.2 Estructura de la memoria 10](#_Toc12995668)

[Capítulo 2. OBJETIVOS 11](#_Toc12995669)

[2.1 Objetivos generales 11](#_Toc12995670)

[2.2 Planificación temporal 11](#_Toc12995671)

[Capítulo 3. TECNOLOGÍA UTILIZADA 12](#_Toc12995672)

[3.1 Python 12](#_Toc12995673)

[3.2 Django 12](#_Toc12995674)

[3.3 SQLite 13](#_Toc12995675)

[3.4 HTML5 14](#_Toc12995676)

[3.5 CSS3 14](#_Toc12995677)

[3.6 JavaScript 15](#_Toc12995678)

[3.7 BootStrap 15](#_Toc12995679)

[3.8 Web3py 16](#_Toc12995680)

[3.9 Ethereum 16](#_Toc12995681)

[3.10 Truffle 17](#_Toc12995682)

[3.11 Infura 17](#_Toc12995683)

[Capítulo 4. BLOCKCHAIN 18](#_Toc12995684)

[4.1 Cómo funciona la blockchain 18](#_Toc12995685)

[4.2 Características de la blockchain 19](#_Toc12995686)

[4.3 Smart Contract 20](#_Toc12995687)

[Capítulo 5. DISEÑO E IMPLEMENTACIÓN 22](#_Toc12995688)

[5.1 Arquitectura general 22](#_Toc12995689)

[5.2 Back-end 23](#_Toc12995690)

[5.2.1 Validación de documentos 25](#_Toc12995691)

[5.3 Front-end 33](#_Toc12995692)

[5.3.1 Estados del documento 38](#_Toc12995693)

[Capítulo 6. UTILIDADES 40](#_Toc12995694)

[Capítulo 7. APÉNDICE 41](#_Toc12995695)

[7.1 Manual de usuario 41](#_Toc12995696)

[Capítulo 8. BIBLIOGRAFÍA 44](#_Toc12995697)

Índice de Figuras

[Ilustración 1 Logo Python 12](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995640)

[Ilustración 2 Logo de Django 13](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995641)

[Ilustración 3 Logo SQLite 13](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995642)

[Ilustración 4 Logo HTML5 14](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995643)

[Ilustración 5 Logo CSS3 14](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995644)

[Ilustración 6 Logo de JavaScript 15](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995645)

[Ilustración 7 Logo BootStrap 15](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995646)

[Ilustración 8 Logo de Ethereum 16](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995647)

[Ilustración 9 Logo de Truffle 17](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995648)

[Ilustración 10 Logo de Infura 17](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995649)

[Ilustración 11 Etapas funcionamiento de la Blockchain 19](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995650)

[Ilustración 12 Infraestructura de la red 23](#_Toc12995651)

[Ilustración 13 Smart Contract 32](#_Toc12995652)

[Ilustración 14 Pantalla de inicio de la Aplicación 33](#_Toc12995653)

[Ilustración 15 Información y acceso a la obtención de ether. 34](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995654)

[Ilustración 16 Página de Rinkeby para conseguir ether 34](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995655)

[Ilustración 17 Transacción en el explorador de la blockchain 35](#_Toc12995656)

[Ilustración 18 Representación del identificador en la aplicación 35](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995657)

[Ilustración 19 Apartado de la aplicación para subir docuementos. 36](#_Toc12995658)

[Ilustración 20 Formulario para enviar solicitud para colaborar 37](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995659)

[Ilustración 21 Correo solicitud para colaborar 37](#_Toc12995660)

[Ilustración 22 Representación de estados de los documentos 39](file:///C:\Users\artena\Documents\Mis_Archivos\mio\Segunda_entrega.docx#_Toc12995661)

[Ilustración 23 Ejemplo dirección smart contract 42](#_Toc12995662)

# INTRODUCCIÓN

## Planteamiento del problema

Uno de los grandes problemas en la actualidad es conocer si la información que recibimos o enviamos mayoritariamente mediante medios electrónicos es recibida de la manera mas adecuada tanto en tiempo como en forma. Este proyecto surge para dar respuesta a esta necesidad, la validación de documentos electrónicos. Aunque, actualmente es cierto que existen medios a través de los cuales se da respuesta a esta necesidad, mediante aplicaciones como*DocuSign* o*HelloSign*, por qué no mezclar esta necesidad con una de las tecnologías que está más en auge actualmente, blockchain y que ofrece como características fundamentales la confianza, la transparencia y la inmutabilidad.

## Estructura de la memoria

En esta sección se describe la estructura de la memoria:

* En el capítulo 1 se realizará una explicación del contexto y la motivación para llevar a cabo el trabajo.
* En el capítulo 2 se determinan los objetivos y planificación temporal del proyecto.
* En el capítulo 3 se hará una explicación de las diferentes tecnologías utilizadas.
* En el capítulo 4 se realizará una introducción y entendimiento sobre la tecnología blockchain.
* En el capítulo 5 se realizará una introducción y explicación sobre la Aplicación tanto desde el punto de vista del frontend como del backend.
* En el capítulo 6 se realizará una explicación del desarrollo de la aplicación en una empresa.

# OBJETIVOS

En este capítulo se van a tratar los objetivos por los cuales se ha decidido llevar a cabo este proyecto.

## Objetivos generales

Como se ha comentado en el anterior capítulo el objetivo principal de este proyecto es validar documentos electrónicos dejando así a disposición del usuario documentos veraces y fiable.

Esto se llevará a cabo mediante la realización de una aplicación en la que el usuario introducirá aquellas personas que quiera que validen el documento a los cuales les llegará un aviso mediante correo electrónico y estos validarán o no el documento. En posteriores apartados se explicará de manera más detallada como se produce esta validación.

## Planificación temporal

La planificación seguida para la realización de este proyecto es la siguiente:

* Reunión con el tutor para ver cuál va a ser el tema a tratar, así como los objetivos y que finalidad se busca con este proyecto.
* Una vez acordado lo anterior, búsqueda de la información necesaria para la familiarización con el entorno.
* Realización de las primeras pruebas y elaboración de una aplicación de prueba.
* Una vez realizada la aplicación de prueba, reunión con el tutor para definir posibles mejoras para el proyecto definitivo.
* Realización de la parte de análisis de la aplicación, realizando diversas pruebas con las diferentes casuísticas contempladas en la aplicación.
* Realización de la aplicación definitivas con las modificaciones finales acordadas.

# TECNOLOGÍA UTILIZADA

En este capítulo vamos a describir las tecnologías que han sido utilizadas para la realización de este proyecto y cuyo concepto es importante para tener una mejor compresión del proyecto.

## Python

Python es un lenguaje de programación de propósito general, orientado a objetos, programación imperativa y en menor medida programación funcional. Debido a las características citadas anteriormente se puede decir que Python es un lenguaje de uso sencillo y de fácil aprendizaje y, por tanto, fue la opción mas idónea de entre los lenguajes de programación para llevar a cabo este proyecto. Fue creado por Guido Van Rossum, con el objetivo de cubrir la necesidad de un lenguaje orientado a objetos de sencillo uso que se podrá utilizar para tratar diversas tareas dentro de la programación, que habitualmente se hacían en Unix usando C. Python es un lenguaje de scripting, independiente de plataforma y como ya se ha comentado orientado a objetos, esto le permite a este lenguaje estar preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso páginas web. Es un lenguaje interpretado esto quiere decir que no se necesita compilar el código fuente para poder ejecutarlo, esto ofrece grandes ventajas como la rapidez de desarrollo y como inconveniente menor velocidad. Python dispone de dos versiones que son usadas en la actualidad la 2.x y la 3.x esta última incluye una serie de cambios que hacen necesario tener que reescribir el código de versiones anteriores. La versión utilizada en este proyecto es 3.6.6.

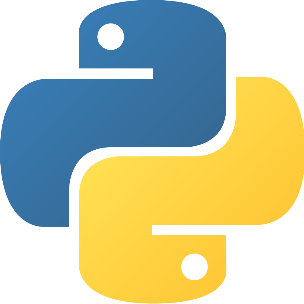


Ilustración 1 Logo Python

## Django

Django es un framework (conjunto de componentes que te ayudan a desarrollar sitios web más fácil y rápidamente) para aplicaciones web gratuito y open source escrito en Python. Fue desarrollado en principio para gestionar varias páginas orientadas a noticias de World Company de Lawrence, Kansas y fue liberada al público bajo una licencia de BSD (Distribución de Software Berkeley). El principal objetivo que tiene Django es facilitar la creación de sitios web complejos, además, pone bastante hincapié en reusar la conectividad y extensibilidad de componentes de desarrollo rápido y el principio DRY (No te repitas). Python es utilizado en todas las partes de framework, incluso en configuraciones, archivos y en los modelos de datos. Django requiere de una versión de Python 2.5 o superiores en este proyecto se ha utilizado la versión de Python 2.7.6. No se necesitarán por tanto otras bibliotecas de Python para obtener una funcionalidad básica. Para la realización de este proyecto se ha utilizado la versión de Django 1.9.5.

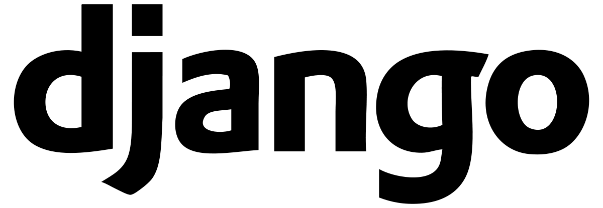


Ilustración Logo de Django

## SQLite

SQLite es una biblioteca escrita en C que implementa un motor de base de datos SQL de dominio público, de alta confiabilidad, incorporado y con todas las funciones. Es compatible con ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad). Es el motor de base de datos mas utilizado en el mundo. A diferencia de los sistemas de gestión de bases de datos cliente-servidor, el motor de SQLite no es un proceso independiente con el que el programa principal se comunica. En lugar de eso, la biblioteca de SQLite se enlaza

con el programa pasando a ser parte integral del mismo. En su versión 3, que es la que se ha utilizado para este proyecto SQLite permite bases de datos de hasta 2 Terabytes de tamaño y también permite la inclusión de campos de tipo BLOB (Objetos binarios grandes).



Ilustración Logo SQLite

## HTML5

HTML5 es el lenguaje de marcado estándar para crear páginas Web. HTML significa Hyper Text Markup Language, es la última versión de HTML.

El término representa dos conceptos diferentes, se trata de una nueva versión de HTML, con nuevos elementos, atributos y comportamientos. Contiene un conjunto mas amplio de tecnologías que permite a los sitios Web y a las aplicaciones ser mas diversas y de gran alcance. Diseñado para ser utilizable por todos los desarrolladores de OpenWeb, esta página referencia numerosos recursos sobre las tecnologías de HTML5, clasificados en varios grupos según su función.



Ilustración Logo HTML5

## CSS3

CSS3 es la última versión de CSS, es un lenguaje que define el estilo o la apariencia de las páginas web, escritas en HTML o documentos XML.

Fue creada para separar el contenido de la forma y esto permite a los diseñadores tener un control mayor de la apariencia de la página.



Ilustración Logo CSS3

## JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. JavaScript es un lenguaje que se utiliza principalmente en su forma del lado cliente, implementado como parte de un navegador web lo que permite mejoras en la interfaz de usuario y páginas web, existe además una forma de JavaScript del lado del servidor. También es significativo su uso en aplicaciones externas a la web. Fue diseñado con una sintaxis muy similar a C, aunque tiene también convenciones y nombres de Java.



Ilustración Logo de JavaScript

## BootStrap

BootStrap es un framework de código abierto para diseño de sitios y aplicaciones web, contiene elementos de diseños basados en HTML y CSS, así como extensiones de JavaScript opcionales. Es conocido como el proyecto mas popular de GitHub, fue desarrollado por Mark Otto y Jacbod Thornton de Twitter como un marco de trabajo que fomentara las herramientas internas.

Está mantenido por Twitter. Es compatible con la mayoría de los navegadores web.



Ilustración Logo BootStrap

## Web3py

Web3.py es una biblioteca de Python para interactuar con Ethereum. Su API se deriva de la API de Javascript de Web3.js y debe ser familiar para cualquiera que haya usado. El punto de entrada común para interactuar con la biblioteca Web3 es el objeto Web3. El objeto web3 proporciona API para hacer posible la interacción con la cadena de bloques ethereum, normalmente conectándose a un servidor JSON-RPC.

## Ethereum

Es una plataforma escrita en software libre, podría ser comparado con el Bitcoin debido a que tiene su propia blockchain pero no es como la mayoría de las criptodivisas existentes. No es solamente una red para reflejar transacciones de valor monetario, sino que sería una evolución de la web denominada Web 3.0 que nos permitirá crear contratos inteligentes utilizando Solidity como lenguaje de programación.

Ethereum usa como moneda interna el Ether esta moneda servirá para ejecutar los contratos inteligentes (Smart contract) dentro de la red. Estos contratos se utilizarán para ejecutar de forma segura gran variedad de servicios. Esta característica de los Smart contract será uno de los puntos clave por los cuales se ha decidido utilizar una testnet de Ethereum para realizar el proyecto.

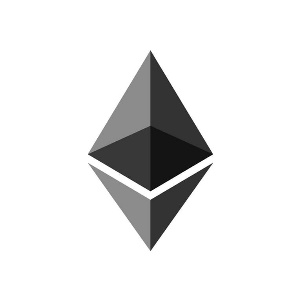


Ilustración Logo de Ethereum

## Truffle

Es el framework más popular para el desarrollo en Ethereum hoy en día, Truffle ofrece Compilación, enlace y despliegue de Smart contracts desde el propio framework, Depuración y testing automatizado de contratos, framework con scripts de despliegue y migraciones en redes públicas y privadas, acceso a cientos de paquetes externos y gestión con EthPM & NPM Consola interactiva para comunicación directa con los contratos, interacción con contratos mediante scripts externos.



Ilustración Logo de Truffle

## Infura

Proporciona un acceso seguro, confiable y escalable a las redes Ethereum. La cual nos permite acceder sin problemas a Ethereum a través de los nodos de carga equilibrada de Infura y la arquitectura inteligente de la misma manera que lo haría a través de sus propios nodos. Está formado por servicios y API en torno a JSON RPC sobre HTTPS y WebSockets que puede usar con sus bibliotecas y marcos favoritos, en las cuatro redes Ethereum.



Ilustración Logo de Infura

# BLOCKCHAIN

La realización de esta aplicación ha tenido como punto de partida el uso y las ventajas de la utilización de la tecnología blockchain. Ya en el capítulo anterior se ha dado una breve introducción de esta tecnología, aunque por el peso de esta para el desarrollo de la aplicación se ha decidido dedicarle un capitulo donde poder desarrollar y dar un entendimiento de manera más extensa.

Aunque la blockchain nació como una base tecnológica de Bitcoin, su uso se ha ido ampliando convirtiéndose en base para el desarrollo de soluciones sobre problemas que afectan a muchos campos. En este proyecto se ha partido del uso de la blockchain como infraestructura distribuida para la validación de documentos.

Una blockchain o cadena de bloques consiste en una sucesión de bloques compuestos por transacciones, siendo una transacción una operación sobre la red. Estos bloques están conectados entre sí, lo que le proporciona la capacidad de ser inalterables. Todos los puntos de conexión (Nodos) tendrán una copia de esta cadena de bloques que son las que tendrán almacenadas todas las transacciones que se han realizado en la red donde se encuentran.

## Cómo funciona la blockchain

Explicar el funcionamiento de la blockchain facilitará el entendimiento del funcionamiento de la aplicación. Este funcionamiento se basa en cuatro puntos claves.

1. Envío de una transacción: El usuario realiza una transacción y esta se distribuirá entre los nodos de la red. En este punto cabe destacar que cada una de estas transacciones suele tener un coste, el objetivo de este es evitar que se utilice de manera inadecuada o se haga un uso malicioso sobre la red. En el caso de este proyecto se ha elegido una testnet[[1]](#footnote-1) de Ethereum llamada Rinkeby en la que se paga las transacciones con ether. Al ser una testnet es gratis obtener ether para hacer las transacciones.
2. Recepción de la transacción: La transacción que realizada por el usuario es recibida por un tipo de nodo especial denominado nodo validador[[2]](#footnote-2) que validarán esta transacción.
3. Validación[[3]](#footnote-3): Los nodos validadores generarán un bloque con las transacciones pendientes. Dicho bloque debe cumplir una serie de reglas criptográficas muy estrictas que serán después comprobadas por la red. Este bloque será aceptado posteriormente por la red mediante un mecanismo de consenso.
4. Envío de nuevo bloque: El nuevo bloque con todas las transacciones validadas es añadido a la blockchain y distribuido. Una vez está el contenido en la blockchain este será inmutable.

En el esquema representado en la ilustración 11 se muestra un resumen de las etapas de funcionamiento de la blockchain.

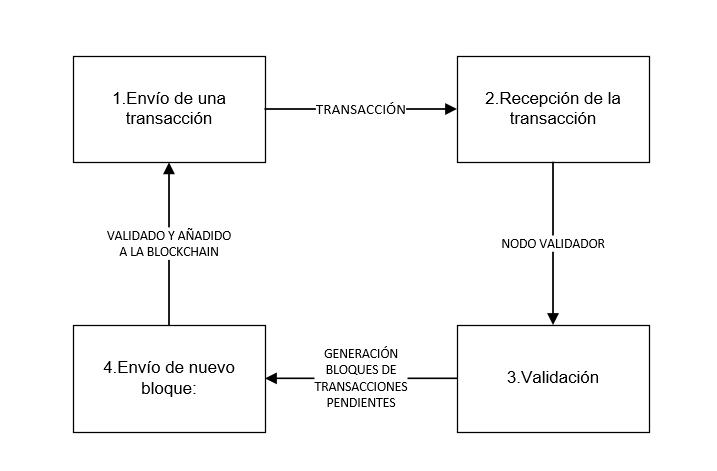


Ilustración Etapas funcionamiento de la Blockchain

## Características de la blockchain

Sus principales características son:

* Es un sistema transparente en el que todas las transacciones son registradas y se podrían consultar públicamente, pudiendo hacer un seguimiento del funcionamiento de la Plataforma.
* Es una red de confianza, la confianza reside en el consenso entre los participantes, sin que exista la figura de un tercero.
* Es un sistema inmutable, una vez introducida, la información, no se puede modificar por parte de ningún actor implicado ni ningún administrador.
* Es un ledger distribuido, Blockchain es una base de datos descentralizada, protegida criptográficamente, en la que cada nodo tiene una copia de toda la información almacenada.
* Es un paradigma de colaboración, se reduce la fricción entre los distintos participantes en base a unas reglas bien definidas y de fácil adopción, sin necesidad de un gran nivel de acuerdo.

## Smart Contract

Hasta ahora un contrato era un documento verbal o escrito, que en la mayoría de casos precisaba de la figura de un intermediario, lo que aumentaba tanto el coste, como el tiempo necesario para poder formalizar dicho contrato. En la actualidad este perfil de contrato está evolucionando al denominado Smart contract. Esto ha derivado a la utilización de un testnet de Ethereum para realizar el proyecto que nos permita crear y utilizar los Smart contract aprovechando así las ventajas que estos ofrecen y que a lo largo del desarrollo de este punto serán explicadas.

Los smart contracts son nuevos tipos de contratos, que van más allá de los contratos digitales o digitalizados, y son posibles gracias a la tecnología blockchain.

Un Smart Contract se caracterizan por ser contratos verificables y ejecutables por sí mismos, sin la necesidad de un tercero de confianza. Se pueden almacenar sobre Blockchain Smart Contracts, contratos autoejecutables y escritos en lenguaje de programación, ajenos al control de nadie y, por tanto, en los que cualquiera puede confiar. Las partes definirían el objeto del contrato, las acciones que se pueden realizar sobre él y las cláusulas de aplicación. Al guardarse en la Blockchain, este sería inmutable y se ejecutaría solo al cumplirse las condiciones acordadas. Aumenta la seguridad y la eficiencia, así como reduce costes en la ejecución y el mantenimiento.

Además, debido a la naturaleza de blockchain, un Smart contract se almacena en la cadena de bloques (Red Blockchain) y se distribuye entre todos los nodos de la red, sin que pueda ser alterado por ninguno de ellos.

# DISEÑO E IMPLEMENTACIÓN

En este capítulo se mostrará la arquitectura del proyecto, así como su forma de implementarlo y la explicación de las partes mas relevantes de la aplicación, el front-end y el back-end.

## Arquitectura general

Esta aplicación permitirá al usuario subir documentos de los cuales necesita validación por el resto de las partes implicadas y así confirmar que tengan el aprobado de todas las partes. En un primer momento se tendrá que subir el documento y este se mandará al smart contract desarrollado para este proyecto el cual se despliega dentro de la blockchain. Esta invitación al resto de las partes, se hará a través de correo electrónico. Estos invitados podrán realizar acciones sobre el documento al que han sido invitados.

Para llevar a cabo este proyecto se realizará una aplicación en un framework web de Python llamado Django que servirá para poder validar documentos de forma online. Cada vez que una de las partes valide el documento se enviará una transacción a la blockchain a través de la API de Infura. Una vez esta transacción queda confirmada en la blockchain esta validación quedará registrada en ella.

Esta blockchain será una red pública basada en Ethereum ya que es una plataforma open source, descentralizada que permite la creación de acuerdos de smart contract entre pares, basada en el modelo blockchain. Estas redes además nos ofrecen una alta disponibilidad y un fácil acceso. Debido a que son las redes blockchain mas globalizadas y maduras.

Para comunicarnos con este tipo de redes desde la aplicación se utilizará la librería web3py para conectaremos vía HTTP al servicio que nos ofrece la API de Infura la cual nos permite poder interactuar con la blockchain tanto a la hora de hacer transferencias como de obtener información de ella.

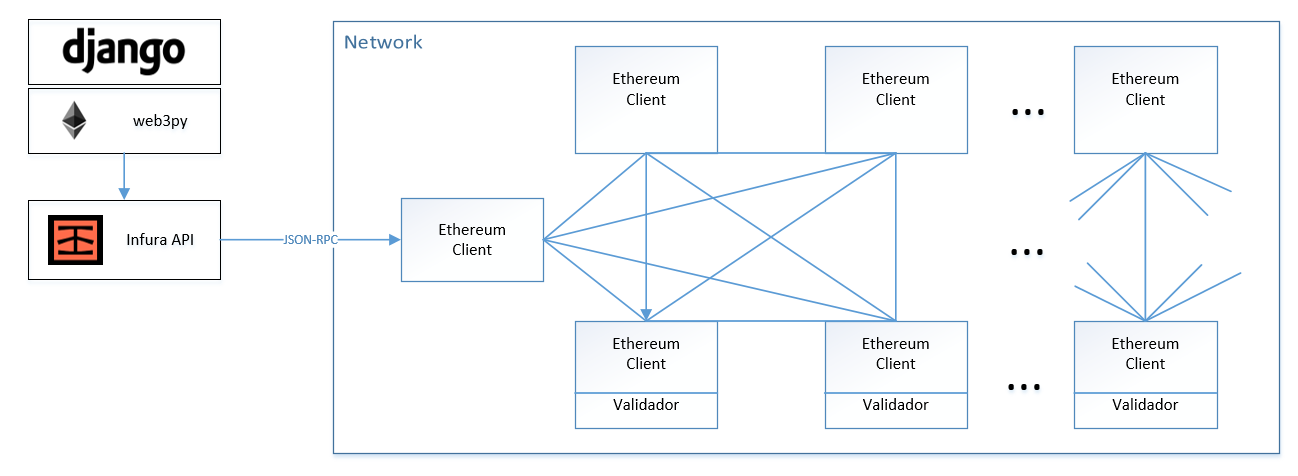


Ilustración Infraestructura de la red

Como se observa en la Ilustración 12 la aplicación utiliza el framework web django que utiliza la librería web3py para comunicarnos con la red blockchain a través de la API de infura. La librería web3py proporciona herramientas para poder obtener los balances de los usuarios, es decir, saber la cantidad de ether que tienen los usuarios, enviar transacciones a la blockchain, saber el estado de las transacciones que se han enviado, obtener información de los Smart contract y la posibilidad de poder crear wallet entre otras funcionalidades.

## Back-end

En este apartado se explicarán las características de la parte del Back-end de la aplicación. La elaboración del back-end tiene lugar en el servidor Django desde el cual se sirven los contenidos al front-end. Este servidor tiene disponibles varios recursos. A continuación, se explicarán de manera detalla cada uno de ellos y su función dentro del back-end.

* /: este recurso nos redirigirá en el caso de que el usuario este identificado al recurso /home que será explicado en puntos sucesivos y en el caso de no estar identificados hará una redirección al recurso /login para que llevemos a cabo el proceso de registro.
* /login: el acceso a este recurso devolverá la página de acceso, es decir la página inicial de la aplicación y en caso que se quiera llevar a cabo el proceso de autentificación se le enviará en el cuerpo de la petición HTTP el usuario y la contraseña con las que se haya identificado.
* /logout: una petición a este recurso hará que la sesión del usuario se cierre dentro de la aplicación.
* /register: cuando un usuario se quiere registrar en la aplicación se mandará en el cuerpo de la petición HTTP los datos que el usuario ha puesto en el formulario de registro. Una vez recibe todos los datos necesarios procederá a crear el wallet en la red blockchain además de la cuenta de la aplicación. Y esta devolverá un mensaje informando si el usuario se ha podido crear correctamente o no.
* /home: este recurso devolverá la página de inicio de la aplicación. En esta página se mostrarán los documentos que el usuario ha registrado en la aplicación, así como los documentos a los que el usuario sido invitado a colaborar.
* /home/doc\_details/[hash]: del acceso a este recurso se obtendrá la información del documento del cual se ha pasado su hash en la url. Cuando el usuario propietario de este documento quiera invitar tanto a un usuario de la aplicación como a uno ajeno le enviará en el cuerpo de la petición HTTP el email del usuario que se quiere invitar a este recurso y este se encargará de asociar este usuario como colaborador de dicho documento además de enviar el email al usuario invitado.
* /upload: el acceso a este recurso devolverá la página de subir un documento a la aplicación. Cuando se tenga decidido el documento que se quiere subir a la aplicación se tendrá que enviar en el cuerpo de la petición HTTP dicho documento y este recurso se encargar de registrarlo y devolver un mensaje donde se confirme si el registro se ha llevado a cabo de manera exitosa o no.
* /profile: este recurso devolverá la información perteneciente al usuario. En el caso de querer editarla se le enviará en el cuerpo de la petición HTTP los contenidos a editar.
* /balance: este recurso devolverá la cantidad de ether que tiene el usuario identificado. Se utilizará para mostrar el balance del usuario en el front-end de manera asíncrona.

### Validación de documentos

En este apartado se explicará como se realiza el flujo de validación de los documentos subidos a la aplicación.

#### Proceso e implementación

Este proceso de validación se inicia desde que un usuario elige cual es el documento que quiere o necesita que sea validado por el resto de parte implicadas en él. Una vez se ha seleccionado el documento el siguiente paso a seguir será subirlo a la aplicación. Cuando un documento se sube a la aplicación se harán dos cosas con él, en primer lugar, guardarlo en la aplicación para que en un futuro pueda ser validado por el resto de las partes y en segundo calcular el hash de dicho documento.

Este hash se calcula con la función hash “SHA256” ¿Y para qué se va a utilizar este hash?, este hash va a ser la información que se mandará a la blockchain y que se comporta como el identificador del documento dentro de la blockchain debido a que un hash es único y unidireccional y esto proporciona que la información que esta en la blockchain pública no este comprometida y pueda ser accesible por cualquiera.

Una vez se ha registrado el documento en la aplicación y confirmado la transacción en la blockchain cambiará el estado del documento a confirmado. Una vez el documento este confirmado se podrá añadir colaboradores a este. Para ello habrá que enviar una petición HTTP al recurso previamente explicado (/home/doc\_details/[hash]) el cual comprobará que no es el propio usuario el que se está añadiendo como colaborar, así como que el colaborador que se está añadiendo no se haya añadido con anterioridad.

Si el usuario invitado no pertenece a la aplicación se le creará, por un lado, una cuenta con dicho email y por otro una wallet que será su identificador dentro de la red blockchain.

Cuando se cumplen todos estos pasos se realizará el acuerdo de colaboración entre el usuario invitado y el documento. Esta relación también se va hacer a nivel de blockchain diciendo que al documento con dicho hash se le ha añadido un colaborador con la dirección del wallet que tenga el usuario invitado.

Una vez se le añade un colaborador a un documento el estado del documento pasara a ser “Pendiente de validación” y se quedará así hasta que el usuario invitado valide o haga una observación sobre el documento. En el caso de haber varios usuarios invitados este estado solo cambiará cuando todos los usuarios hayan validado el documento o uno de ellos haya hecho una observación. Cuando un usuario invitado realiza cualquiera de estas acciones sobre el documento estas quedarán registradas en la blockchain. Diciendo que el usuario con cierta wallet ha realizado una acción sobre el documento con el hash correspondiente.

Las acciones que pueden ser realizadas por el usuario serán:

* Validarlo: estar conforme con la información que contiene el documento. Esto generará una transacción que se enviará a la blockchain diciendo que este usuario ha validado dicho documento.
* Añadir observación: cuando el usuario invitado no está conforme con la información que se muestra o tiene algunas sugerencias que pueden hacer que el documento mejore. Estos comentarios se registrarán a nivel de blockchain diciendo que este usuario ha generado un comentario sobre dicho documento.

Cada una de estas transacciones en la blockchain generará un evento. El cual se lanza desde el smart contract creado para el proyecto con el cual se podrá ver en el etherscan de Rinkeby. Quedando como prueba irrefutable de que dicho usuario generó cierta acción en un cierto momento debido a que este evento proporciona la información suficiente para poder hacer este mapeo.

#### Smart Contract de Validación

En este apartado se va a realizar una explicación detallada acerca de los smart contract dado que a estos van a ir dirigidas todas las transacciones que se realizan desde la aplicación a la blockchain. Los smart contract en redes Ethereum se programan con el lenguaje de programación Solidity en el caso específico de este proyecto se ha utilizado la versión 0.5.5.

Dentro de la aplicación se ha decidido denominar a este smart contract como “Validator” esto se debe a que se utilizará para la validación de los documentos en la red blockchain. Este nombre queda reflejado en el código tal y como se puede observar en la Ilustración 13 en la línea que identifica el “contract”. Además, en esta Ilustración se pueden encontrar los eventos que se han definido dentro de él. En este caso de esta aplicación hay tres:

* DocumentAdded, este evento se emite cuando un nuevo documento se ha registrado en el smart contract, este nos proporciona la información de cual es el nuevo hash del documento que se acaba de añadir, así como el usuario que lo ha registrado.
* NewInvitation, es el evento que se emite cuando el propietario de un documento añade a otro usuario a participar en él. Este evento nos proporciona cual es el hash del documento al cual se ha añadido este usuario invitado y el identificador del usuario invitado.
* ActionAdded, este evento se emitirá cada vez que un usuario de los invitados a un documento realice una acción sobre él. En este evento se puede ver la información del hash del documento sobre el cual se ha realizado la acción, el identificador del usuario que ha realizado la acción y el comentario que haya puesto. En el caso de que un usuario valide el documento este comentario será “Validated” en caso contrario el mensaje será el que haya introducido el usuario en front-end.

Después de los eventos se definen las estructuras de datos que hay en el smart contract, cuatro en el caso de esta aplicación, en la cuales se almacenará la información necesaria para el correcto funcionamiento del smart contract. Cabe mencionar en primer lugar, que todas las estructuras están definidas como “*mapping* “esto quiere decir que son un diccionario[[4]](#footnote-4), por lo que no pueden existir dos elementos con una misma clave. Los diccionarios son estructuras de datos muy extendidos en la mayoría de los lenguajes de programación. A continuación, se va a realizar una explicación de cada una de estas estructuras de datos.

* hashDocument: en esta estructura de datos se van guardar un conjunto de pares clave y valor en la cual la clave será el hash del documento que se ha subido a la aplicación y el valor será una variable booleana que indicará únicamente su existencia dentro de esta estructura de datos.
* hashOwner: tal y como indica su nombre en esta estructura se van a guardar un conjunto de pares de clave y valor donde la clave será el identificador del documento, es decir, el hash y el valor será la dirección del wallet del usuario que ha subido este documento, es decir, el propietario de este documento.
* documentValidators: en esta estructura se van almacenar como clave el hash del documento y como valor un array. Un array, es un tipo de datos estructurado que permite almacenar un conjunto de datos homogéneos, es decir, todos ellos del mismo tipo y relacionados. Cada uno de los elementos que componen este array son de tipo simple y más concretamente caracteres. Estos caracteres son las direcciones de las wallets de los colaboradores de dicho documento. De esta manera se crea la relación entre documento y colaboradores.
* hashValidatorUser: esta es una estructura un poco diferente a las anteriores debido a que esta formada por dos mappings anidados. En el primer mapping tendrá almacenado como clave el hash del documento y como valor otro mapping que contendrá como claves las direcciones de los wallets de los colaboradores de dicho documento y de valor una variable booleana que nos indicará si un colaborador ha sido invitado a colaborar con dicho documento.
* documentComments: esta estructura igual que la anterior esta formada por dos mappings anidados. En el primer mapping tendrá almacenado como clave el hash del documento y como valor otro mapping que contendrá como claves las direcciones de los wallets de los colaboradores de dicho documento y de valor el comentario que este usuario haya puesto en la aplicación para dicho documento. En el caso que el colaborador valide el documento la aplicación no le pide que escriba ningún texto, por lo tanto, se le pasará por defecto el valor “Validated”.

Una vez analizadas las estructuras de datos del smart contract se ha visto interesante analizar y explicar como las funciones utilizan estas estructuras para obtener el resultado deseado. El número de funciones definidas son cuatro de las cuales tres de ellas son públicas y una de ellas privada, que sea privada implica que esta solo se puede utilizar dentro del smart contract no será accesible desde fuera. Cada una de estas funciones consiste en:

* newDocument: esta función tal y como indica su nombre añadirá un nuevo documento al smart contract. Esta función recibe como parámetro el hash del documento que se desea añadir. En primer lugar, va utilizar la función require que nos proporciona solidity. La función require lo que hace es exigir que se cumpla la condición definida en el primer parámetro para que se pueda seguir ejecutando la función en el caso de que no así, lanzará una excepción con el mensaje de error definido en el segundo parámetro.

En este require la condición defienda es que el nuevo hash que se está intentando añadir no exista ya en el smart contract. Para ello se va a llamar a la función hashExists para conocer si este hash esta registrado o no. En el caso de que esta función devuelva que sí, devolverá el mensaje de error definido en el segundo parámetro. Si se ha cumplido esta condición, es decir, que el documento no está registrado el siguiente paso será almacenar como clave el hash del documento en la estructura de datos hashDocuments y poner como valor “true”. Además, se añadirá también a la estructura de datos hashOwner en la cual ese valor será la dirección del wallet que ha ejecutado esta transacción y que pertenecerá al usuario propietario de este documento. Por último, emitirá el evento que indicará que se ha generado un nuevo documento en el smat contract. Este evento es DocumentAdded que contendrá la información del hash del nuevo documento añadido y la dirección del usuario propietario.

* hashExists: esta función es la única función privada dentro del smart contract se utilizará para saber si el hash de un documento ya existe en el smart contract o no. El hash del documento lo recibirá como parámetro. Para conocerlo va utilizar la estructura de datos hashDocument a la cual pasándole el hash de un documento devolverá una variable booleana diciendo si este hash ha sido encontrado, devolviendo “true” en caso afirmativo o “false” en caso contrario.
* addValidatorUser: será la encargada de añadir un colaborador a un documento. Esta función recibirá como parámetro el hash del documento que el colaborador quiera añadir y la dirección del wallet del colaborador. Para poder llevar a cabo este proceso se debe asegurar en primer lugar la existencia de este documento dentro del smart contract para ello se utilizará la función require con esta condición. La condición será que la función hashExists devuelva el valor booleano indicando si se ha encontrado o no. En segundo lugar, deberá hacer otra comprobación con la función require en la que compruebe que no se esté intentado añadir como colaborador el propietario del documento. Para ello se utiliza la estructura de datos hashOwner que indicará cual es el propietario de dicho hash.

Una vez se han cumplido todos estos requerimientos se van a realizar una serie de comprobaciones adicionales para poder añadir el colaborador.

La primera se comprobará que el usuario que ha mandado la transacción sea el propietario del documento, porque solo él puede añadir colaboradores. La segunda que este usuario no estuvierá ya añadido como colaborador en este documento, en este caso no será necesario añadirlo nuevamente. Una vez realizadas estas comprobaciones se colocará el valor “true” en la estructura hashValidatorUser almacenando desde ese momento a este usuario como colaborador. A continuación, se añadirá la dirección del wallet del usuario colaborador al array de colaboradores del documento en la estructura documentValidators, además se añadirá en la estructura documentComments un comentario por defecto con valor “Pending”, que indicará que el usuario todavía no ha realizado ninguna acción sobre el documento.

Por último, se emitirá el evento que indicará que se ha añadido un nuevo colaborador en el smart contract. El nombre de este evento será NewInvitation y contendrá información acerca de cual es el hash del documento sobre cual se ha añadido el colaborador y la dirección del wallet dicho colaborador.

* addValidationUser: esta será la función a la que llamará la aplicación cuando el colaborador realice una acción sobre un documento y que registrará esta acción en el smart contract. Esta función recibe como parámetros el hash del documento sobre el que se ha realizado la acción, la dirección del wallet del colaborador y el comentario que este usuario ha puesto en la aplicación. Para añadir esta acción primero tiene que realizar una serie de comprobaciones utilizando la función require. Al igual que en la función anterior se van hacer las comprobaciones acerca de si el documento indicado se encuentra en el smart contract y si el usuario que ha realizado dicha acción puede o no ser el propietario del documento. Pero además de estas comprobaciones se va hacer una adicional que comprobará sí el usuario que ha realizado la acción era un colaborador del documento indicado. Una vez completadas todas estas comprobaciones se añadirá en la estructura de datos documentComments el comentario realizado por el colaborador. En el caso del que colaborador haya validado el documento en el comentario aparecerá el valor “Validated”. Una vez se haya añadido el comentario se emitirá el evento ActionAdded indicando sobre que documento se ha realizado la acción, que usuario la ha realizado y que comentario ha dejado.

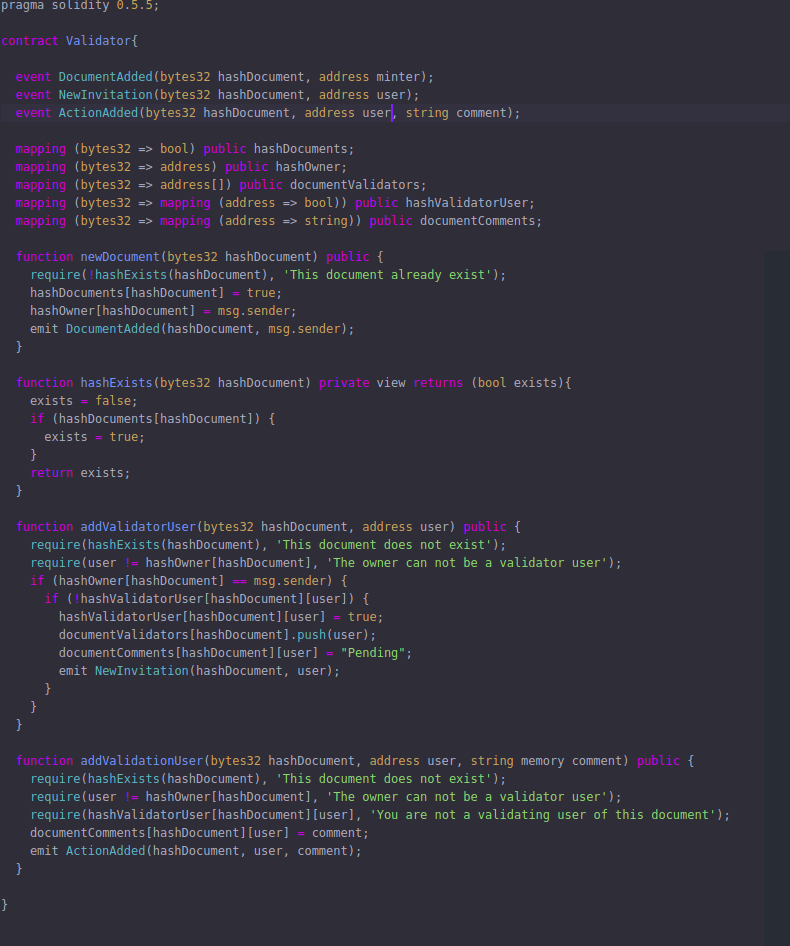


Ilustración Smart Contract

## Front-end

Es la parte que el cliente ve de la aplicación, es decir, la página web. Esta, está elaborada con HTML5, CSS3 y JavaScript para darle un mejor diseño. La aplicación está pensada para que su utilización sea sencilla y reporte al usuario una experiencia de uso cómodo y rápido. Constará de una página principal y diferentes pestañas dentro de su página de usuario donde tendrá a su disposición las diferentes funcionalidades de la aplicación. A continuación, se detalla el funcionamiento interno y las bases en las que están fundamentadas las diferentes acciones que son posibles de realizar en la aplicación.

En primer lugar, el usuario deberá registrase o identificarse para poder acceder a la aplicación. Una vez identificado accederá a su página personal. En la Ilustración 14 se muestra una captura de pantalla de inicio de la aplicación.

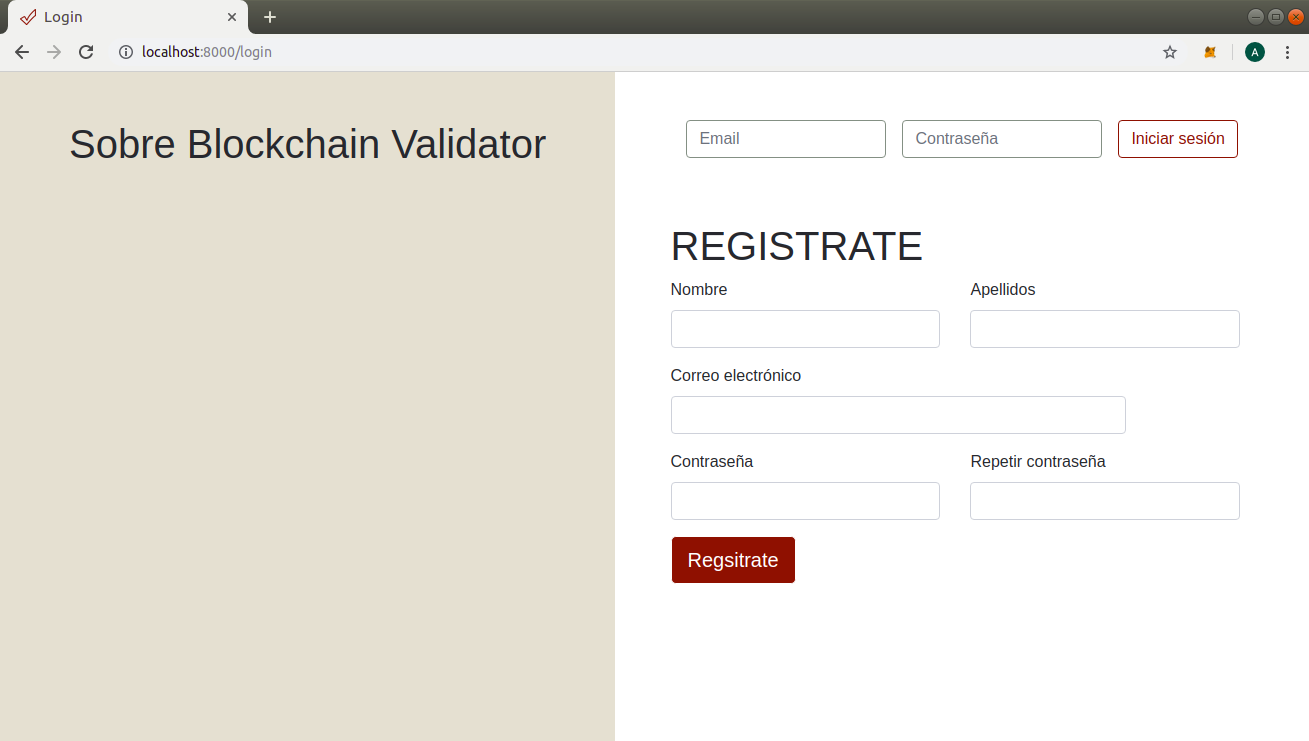


Ilustración Pantalla de inicio de la Aplicación

Cuando el usuario se registra en la aplicación se le crea un wallet en la red blockchain a la que esta conectada. En este caso al haber seleccionado Rinkeby como red se le creará un wallet dentro de esta. Este wallet se convertirá desde este momento en su identificador dentro de la red. Este identificador se encontrará en la pestaña del perfil de usuario con el título “Address”.

Para poder realizar acciones en la aplicación es necesario tener un wallet con fondos dado que como se ha comentado en anteriores capítulos tienen un coste, en este caso la moneda que nos permite realizar estas transacciones se denomina ether. En este caso al estar sobre una testnet de ethereum conseguir ether va a ser totalmente gratuito. Dentro de la página de perfil de usuario encontramos un apartado donde se mostrará la cantidad de ether de los que dispone y un botón definido como “Depositar” que nos llevara a una página donde conseguir estos fondos de manera gratuita.

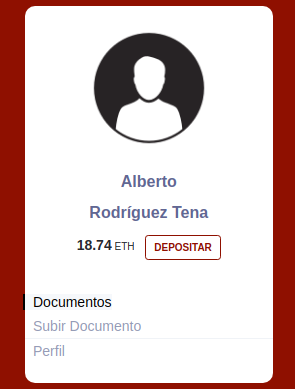


Ilustración Información y acceso a la obtención de ether.

Esta página nos pedirá que pongamos una publicación en Twitter o en Facebook con la dirección de nuestro wallet. Una vez realizada la publicación pondremos el link de la publicación en el formulario y seleccionaremos la opcion “Give me Ether” donde tendremos que elegir la cantidad de ether que queremos obtener.

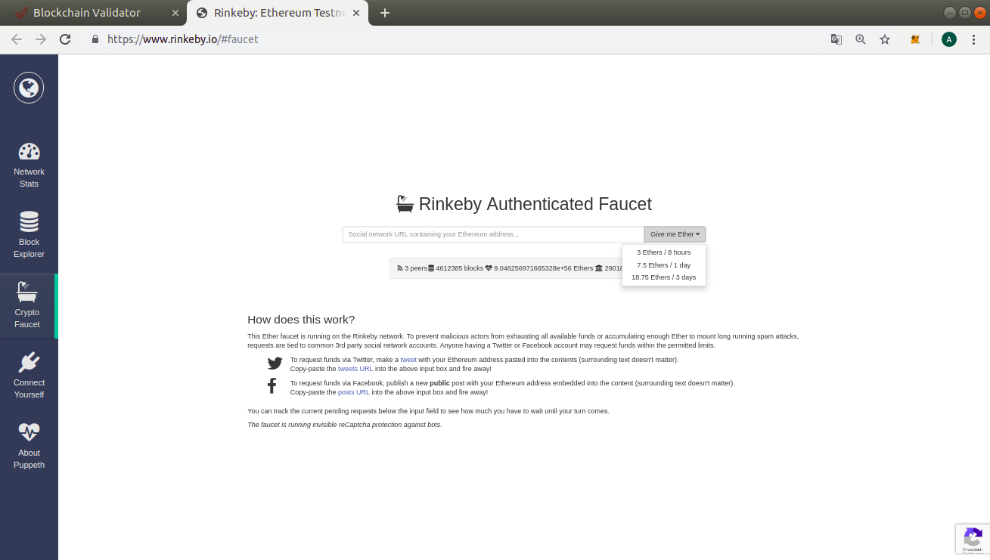


Ilustración Página de Rinkeby para conseguir ether

Una vez realizado todo este proceso el usuario tendrá a su disposición el número de ethers solicitados y la posibilidad de subir documentos. Para subir documentos el usuario deberá acceder al apartado “Subir documentos” y seleccionar la acción subir dentro de esta pestaña, seleccionará el documento el cual quiere subir a la aplicación una vez subido se le calcula y asigna un hash el cual será único y unidireccional. Una vez calculado el hash se manda la transacción a la Blockchain. A continuación, en la ilustración 17 e ilustración 18 se muestra como la transacción en la blockchain se ha llevado a cabo de manera correcta. Se puede ver que en ambas el identificador de la transacción es igual. En la ilustración 17 se observa que la transacción ha sido confirmada y que en el apartado Transacción Hash el identificador que aparece en la ilustración 18 en el apartado Tx con el mismo identificador

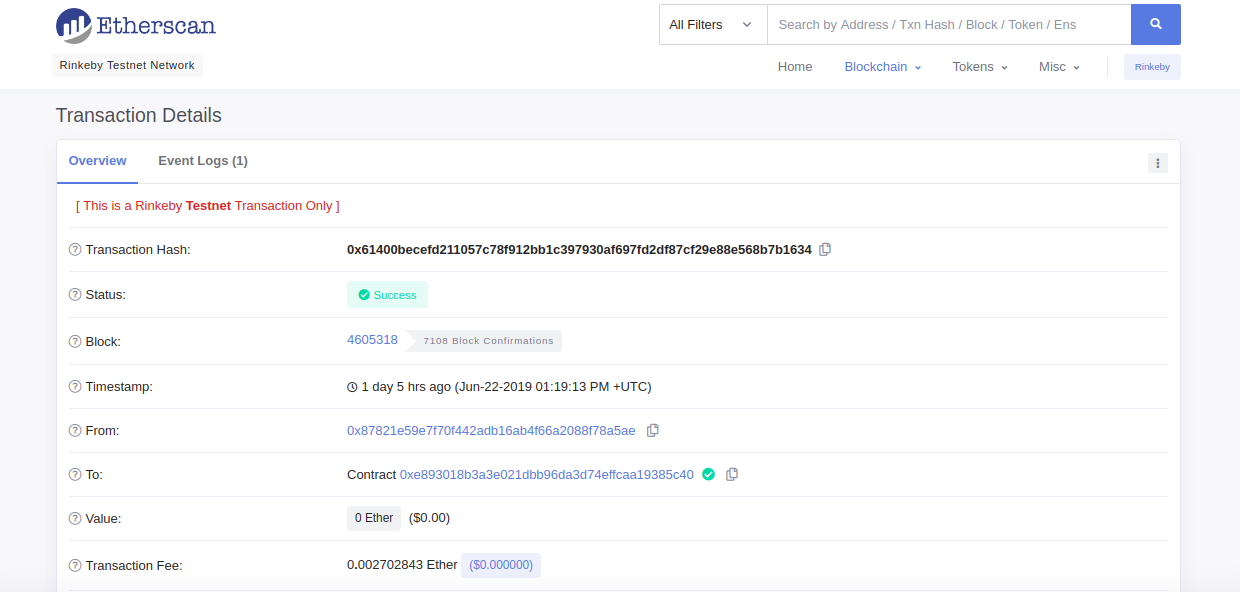


Ilustración Transacción en el explorador de la blockchain

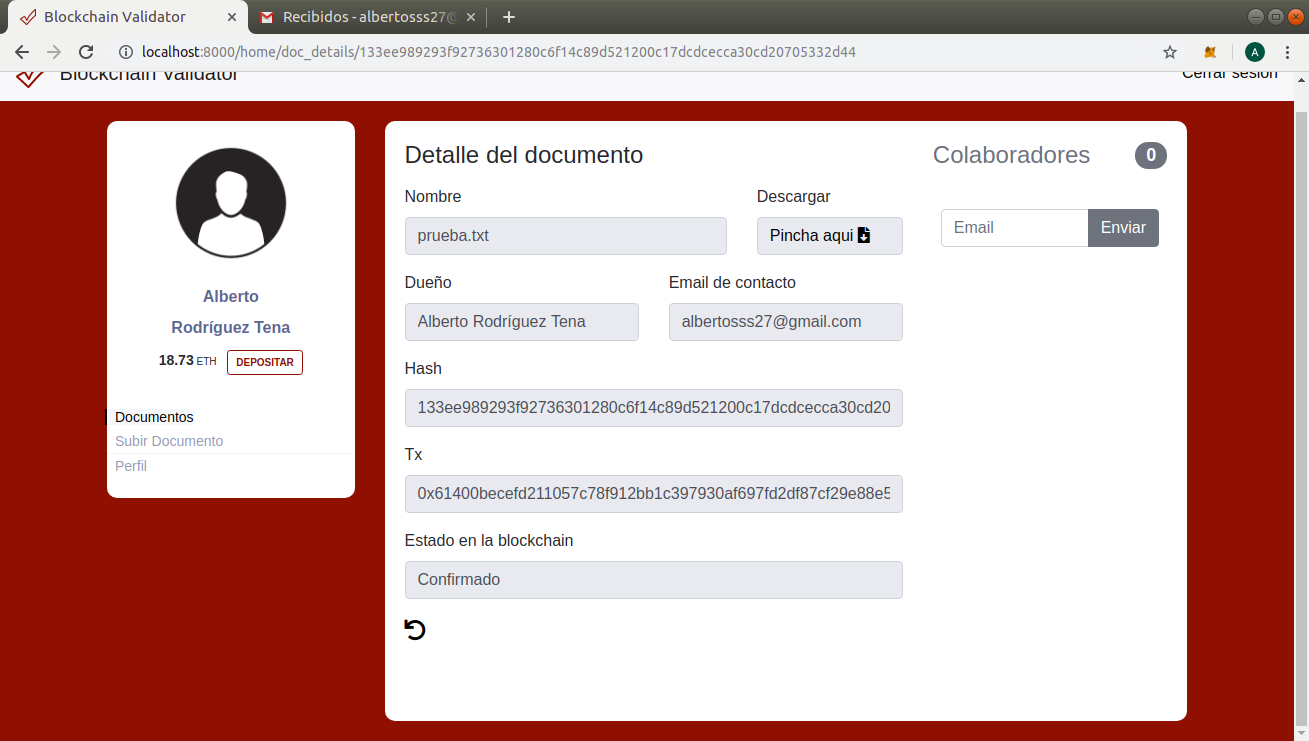


Ilustración Representación del identificador en la aplicación

El hash creado se guardará en el Smart contract creado para la aplicación, por tanto, desde ese momento el documento quedará asociado al wallet del usuario el cual creado en el momento que el usuario se registra en la aplicación, haciendo así que si otro usuario intenta subir el mismo documento este será avisado de que este documento ya está registrado por otro usuario (especificando además el nombre del usuario propietario). Esta transacción transcurridos unos segundos será confirmada en la blockchain por los por los nodos validadores de la red. A continuación, se adjunta captura de la aplicación donde se puede observar el hash calculado para subir un documento.

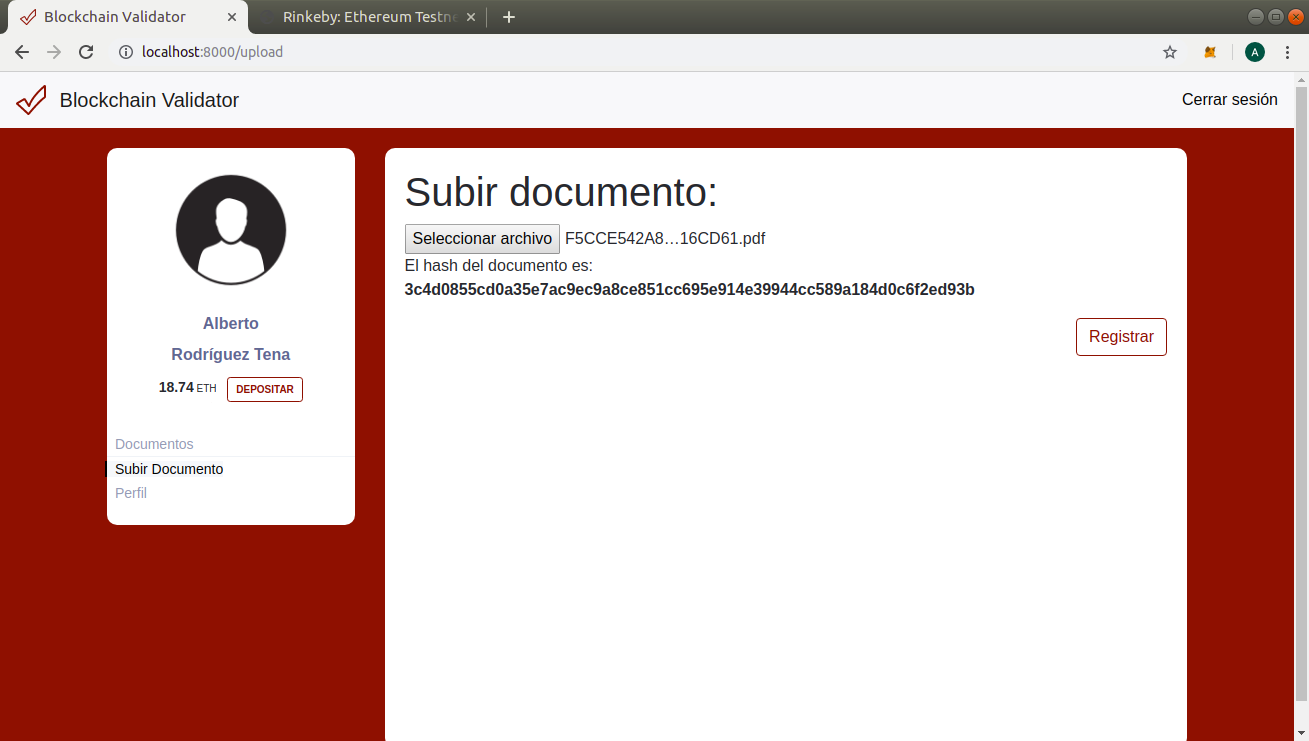


Ilustración Apartado de la aplicación para subir docuementos.

Completado el proceso de registro y confirmación en la blockchain del documento la aplicación permitirá al usuario la posibilidad de enviar a los usuarios implicados una solicitud de colaboración.

Esta solicitud la enviará el usuario propietario del documento a través de un formulario que se encontrará dentro del apartado “Documentos” accediendo a detalles del documento aparecerá este formulario donde se introducirá el email del usuario al que quiera añadir como colaborador para verificar el documento.

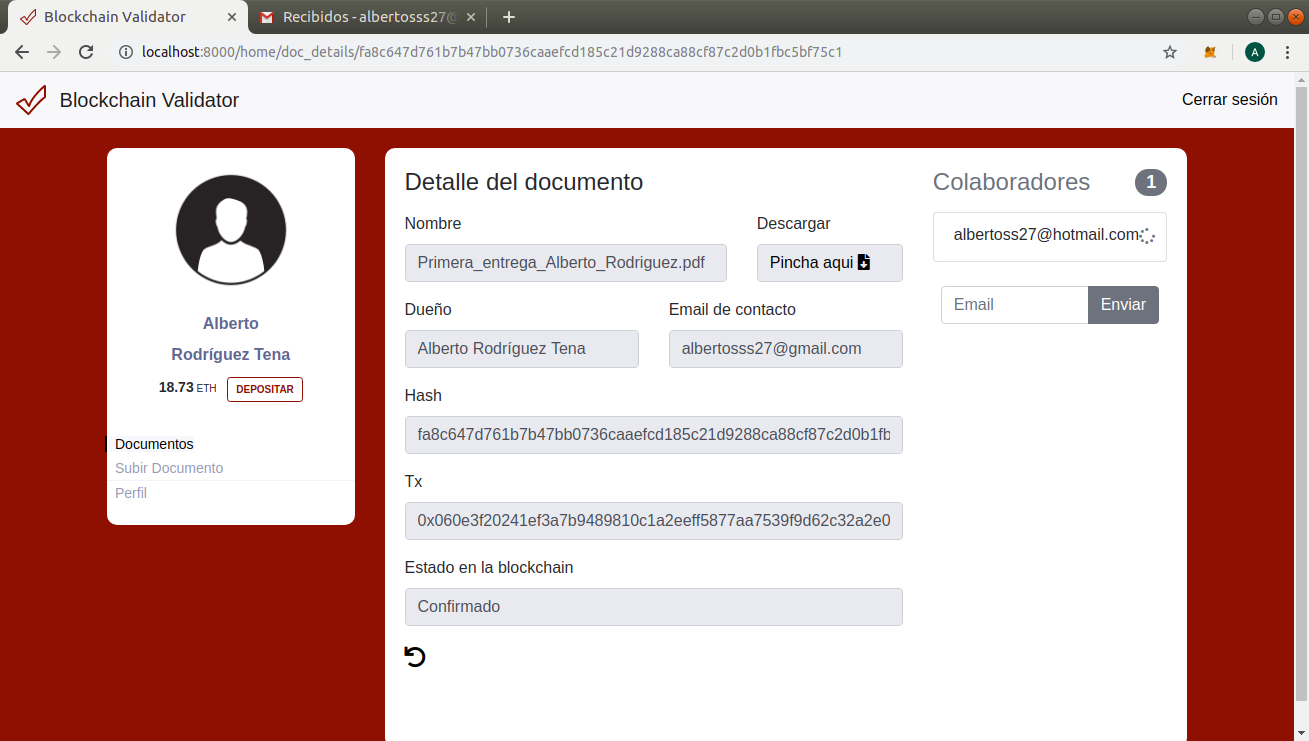


Ilustración Formulario para enviar solicitud para colaborar

Esto generará un correo de invitación que le llegará al usuario colaborador con un enlace a la aplicación. En el caso de que este usuario no esté registrado en la aplicación este enlace le llevará a la página de registro donde tendrá que registrarse con el email al cual iba asociada la invitación para poder realizar cualquier acción dentro de la aplicación ya sea validar o comentar cualquier documento al que haya sido invitado como colaborador.

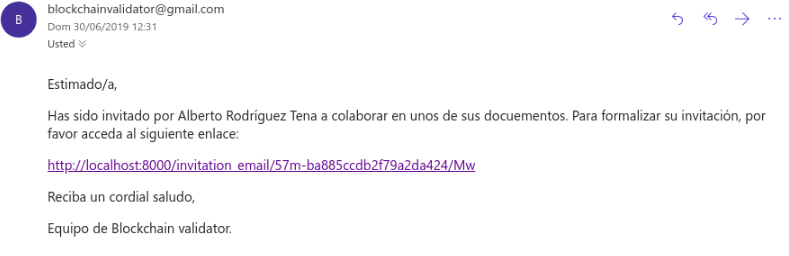


Ilustración Correo solicitud para colaborar

Cualquier documento subido a la aplicación quedará registrado y estará a disposición de los usuarios colaboradores. El perfil del usuario cuenta además con un apartado donde podrá ver todos aquellos documentos a los que el usuario ha sido invitado a colaborar.

Esta aplicación no solo ha sido creada para que el documento sea validado por los colaboradores, sino también para permitir al usuario creador del documento la posibilidad de ofrecer documentos con la mayor veracidad posible obteniendo no solo validación por parte de los colaboradores sino también la posibilidad de ver los comentarios que los colaboradores consideran relevantes para la posterior validación del documento. Estas validaciones se mandarán a la blockchain para que también queden reflejadas en ella.

### Estados del documento

Una vez los documentos han sido subidos a la aplicación podrán encontrarse en los siguientes estados que estarán representados por colores en función de la criticidad del estado en el que se encuentren. A continuación, se detallan los estados por orden de criticidad partiendo de un nivel menor a uno mayor.

* Documentos pendientes de ser confirmados por la Blockchain cuando un documento se encuentra en este estado implica que los nodos validadores todavía no han confirmado la transacción de registro del documento. Este estado estará representado por el color blanco.
* Documentos confirmados por la blockchain que no tienen colaborador, en este estado se encontraran cuando no se haya solicitado colaboración por parte del usuario propietario del documento. Este estado estará representado por el color azul.
* Documentos que tienen colaboradores, pero estos no han colaborado con el documento. En este estado se encontrarán los documentos en los cuales el usuario propietario haya mandado una solicitud de colaboración y esta solicitud no ha obtenido respuesta. Este estado estará representado por el color morado.
* Documentos validados por colaboradores son los documentos validados por los usuarios colaboradores a los que el usuario propietario del documento ha enviado una solicitud para colaborar con el documento. Este estado estará representado por el color verde.
* Documentos con comentarios de colaboradores que un documento tenga asociado esta anotación implica que el usuario colaborador ha dejado un comentario con aquellos aspectos en los que considere necesario una mejora por parte del usuario propietario del documento. Este estado estará representado por el color rojo.

En la siguiente imagen se observa una captura de la aplicación donde se puede ver el estado de algunos documentos.

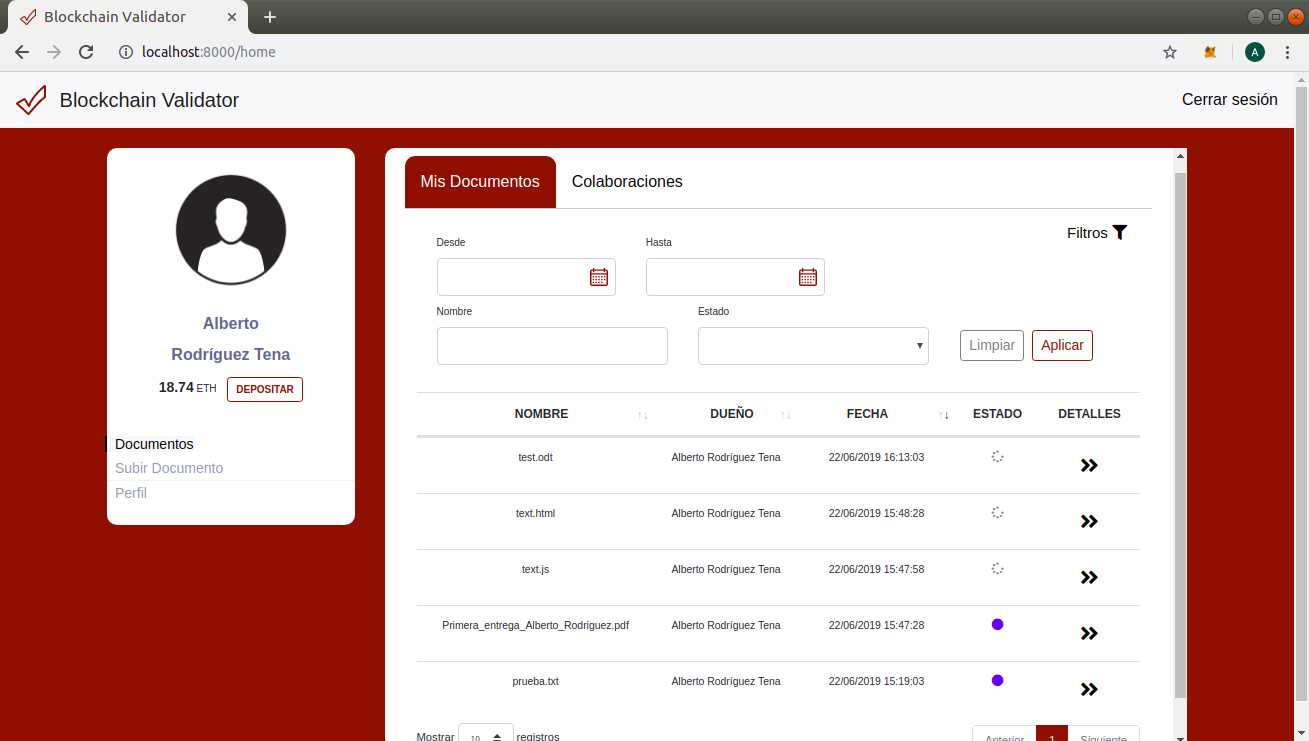


Ilustración Representación de estados de los documentos

# UTILIDADES

En la actualidad la tecnología blockchain es una tecnología en auge que está empezando a introducirse dentro de los diferentes sectores. Cada vez es más común que cualquier proceso de gestión o de contacto desde una empresa con un usuario conlleve un intercambio de información. Este intercambio se suele realizar a través de algún medio electrónico ya sea la plataforma propia o vía correo electrónico. Esto complica en cierta manera el proceso de validar la veracidad de estos contratos o documentos importantes en los cuales es necesario que ambas partes hayan validado.

Este proyecto intenta facilitar la manera en que se validan estos documentos, siendo de utilidad para cualquier empresa que preste servicio. En este caso se ha decidido contextualizarlo para ser utilizado por una start up. No solo facilitara la validación de documentos dentro de la propia empresa sino también en la validación de documentos en los que haya que contactar con gente externa a la compañía.

En este capítulo vemos interesante representar en que punto y para que estaría involucrada nuestra aplicación en una empresa, para hacer esto hemos elegido una empresa de poner en contacto a personas que trabajan por su cuenta “Freelance” con empresas que necesitan trabajadores externos a su plantilla.

En la actualidad el modelo de trabajador freelance se esta extendiendo dado que aporta al trabajador la posibilidad de poder dedicarse a aquella actividad que consideran su pasión sin dejar de lado recibir una retribución económica.

En este punto se puede dudar que cual sería el papel de nuestra aplicación para esta comunicación. La respuesta es sencilla, la aplicación además basarse en la validación de documento se podría entender como una aplicación en la que el usuario pueda generar un historial de aquellos documentos en lo que ha participado. Esto le aporta valor al usuario que utilice esta plataforma porque es una manera de demostrar en que contratos ha participado. Aportando a las empresas la capacidad de poder buscar en la aplicación mediante la dirección del wallet del usuario y poder ver la cantidad de contratos en los que ha estado implicado y con que empresas. Debido a que se podría crear un nuevo rol en el que las empresas solo pudiesen subir ellas documentos a la plataforma y que además les permitiera ver el address de otras empresas.

# APÉNDICE

## Manual de usuario

En este apartado se explicará como instalar este proyecto en una máquina con un sistema operativo Unix. Para ello se va a suponer anteriormente se debe tener el código fuente en la máquina.

Para poder realizar la instalación de este proyecto se necesita en primer lugar tener la versión de Python 3.6 o superiores. Una vez verificado que es esta la versión instalada tendra que crear un entorno virtual que contendrá todas las librerías necesarias para el correcto funcionamiento de la aplicación.

Para crear un entorno virtual con Python 3.6 se ejecutará el siguiente comando:

* *python3 –m venv my\_env*

Una vez se tenga el entorno virtual creado se procederá con la instalación de las librerías que necesita la aplicación. Para ello se ha elaborado un fichero llamado *requirements.txt* el cual contiene la información sobre todas las librerías y las versiones que necesita la aplicación.

* *pip install –r requirements.txt*

Una vez terminado esto, se dispondrá de todas las librerias necesarias para poder arrancar, pero todavía quedará por configurar una serie de parámetros necesarios para el correcto funcionamiento de ella.

A continuación, se explicará como compilar y desplegar el smart contract que utiliza la aplicación. Para ello se ha utilizado truffle, aunque existen otras herramientas para poder hacer esto. En primer lugar, será necesario instalar truffle para ello se deberá tener instalado Node.js v6+ LTS y NPM. Node.js es un entorno multiplataforma que nos permite ejecutar código JavaScript en la capa de servidor. NPM es el gestor de paquetes de NodeJS. Una vez instalados procedemos con la instalación de Truffle.

* *npm install –g* [*truffle@5.0.2*](mailto:truffle@5.0.2)

Una vez instalado habrá que desplazarse con la terminal hasta el directorio *validator\_contract* dentro de nuestro proyecto y ejecutar el comando que se muestra a continuación, el cual nos compilara y desplegará en el smart contract Validator.sol en la red de Rinkeby

* *truffle migrate –network rinkeby.*

Este comando comenzará compilando nuestro smart contract y acto seguido procederá a desplegarlo dentro de la red blockchain. Una vez se haya validado la transacción que despliega el smart contract devolverá su dirección dentro de la red. Este valor va a ser necesario para la configuración de la aplicación con este valor se le indicará a que smart contract se debe conectar.

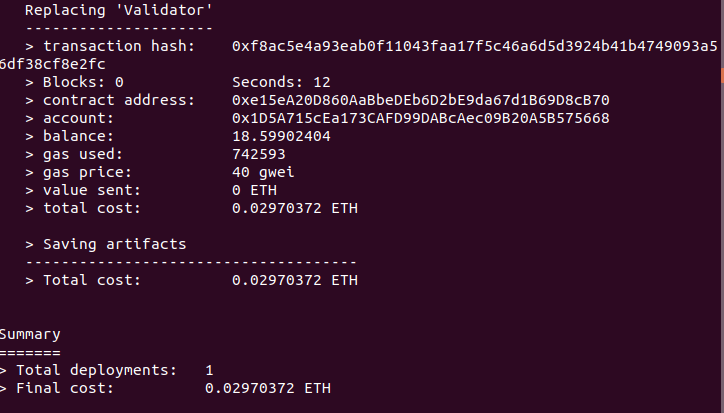


Ilustración Ejemplo dirección smart contract

Una vez se ha obtenido el valor necesario para configurar la aplicación. Se va proceder a la explicación de como configurar la aplicación. Para ello se tiene que tener el entorno virtual activado, ahora necesitará desplazarse por la terminal hasta la carpeta blocksign dentro de nuestro proyecto. Una vez en ella ejecutaremos la siguiente instrucción para aplicar todas las migraciones.

* *Python manage.py migrate*

Este comando lo que hará será crear todos los modelos de datos que la aplicación necesita para su funcionamiento. Se ha elaborado un command el cual nos inicializa la base de datos con todos los parámetros que la aplicación necesita. Este command recibe un parámetro el cual es la dirección del smart contract al cual se va a conectar. Este commad se ejecuta de la siguiente manera

>>*python manage.py init\_db –address* >>0xe15eA20D860AaBbeDEb6D2bE9da67d1B69D8cB70

Una vez terminado esto ya estará la aplicación lista para poder ser utilizada. Para lanzarla hay que ejecutar el siguiente comando

* *python manage.py runserver*

Esto nos lanza un servidor local en el puerto 8000 por defecto. Se puede cambiar el puerto si se desea. Y ya se podrá acceder a través del navegador accediendo a la url http://localhost:8000 en la cual debería aparecer aplicación si todo ha funcionado correctamente.

# BIBLIOGRAFÍA

BootStrap. Web sobre bootstrap. <https://getbootstrap.com/docs/4.3/getting-started/introduction/>

Css3. Web sobre css3.<https://devdocs.io/css-color/>.

Django. Web oficial de django project. <https://docs.djangoproject.com/en/2.2/>.

Html5. Web sobre html5. <https://www.w3.org/TR/html52/>.

SQlite. Web oficial de sqlite. <https://www.sqlite.org/>

Infura. Web oficial de Infura. <https://infura.io/docs>

Truffle. Web oficial de Truffle. <https://truffleframework.com/docs/truffle/overview>.

1. Copia idéntica de una blockchain con el objetivo de probar cambios y nuevos desarrollos. [↑](#footnote-ref-1)
2. Nodos que tiene el rol de la creación de nuevos bloques y la verificación de los bloques añadidos a la cadena. [↑](#footnote-ref-2)
3. El acto de procesar transacciones en el sistema identificados como bloques [↑](#footnote-ref-3)
4. En programación un diccionario son un tipo de estructuras de datos que permite guardar un conjunto no ordenado de pares clave-valor, siendo las claves únicas dentro de un mismo diccionario. [↑](#footnote-ref-4)