



TFE UNIR

TITULACIONES TOKENIZADAS

Experto Universitario en desarrollo de aplicaciones Blockchain

ERC20 y ERC721

Sistema dual con un token identificativo para las asignaturas y un token que representa ECTS adquiridos, ambos ligados a un mecanismo de identidad digital.

José Pastor Galiana
j.pastor.galiana@gmail.com

1 Contenido

2	Objetivo e introducción	2
3	Estado del arte	4
4	El proyecto	10
5	Justificación del uso de blockchain	12
6	Análisis y diseño técnico	15
	Arquitectura en 3 capas	20
	Arquitectura Frontend	22
	Capa lógica	23
	Arquitectura Backend	24
7	La solución	33
8	Despliegue e implantación	55
	Ganache	57
	Testnet y Alastría	58
9	Testing	59
	Ganache	59
	Testnet	64
	Alastria	67
10	Conclusiones	71
11	Referencias	72

2 Objetivo e introducción

Como colofón al trabajo realizado durante este curso, se plantea por parte de la UNIR el reto de desarrollar una aplicación que permite demostrar los conocimientos adquiridos en las diferentes asignaturas

Es destacable la necesidad de conocimientos en diferentes disciplinas para poder enfrentarse al reto, en concreto:

- Conocimiento sobre tecnología blockchain, centrando en este caso el foco en Ethereum/Quorum.
- Programación básica de frontales web (HTML, CSS y JavaScript).
- Conocimiento de las librerías que nos permitirán comunicar la web con nuestra red blockchain, nos centraremos en web3.js.
- Capacidades de desarrollo de Smart contracts con Solidity
- Habilidades de programación de smart contracts (Solidity) y el IDE de desarrollo más habitual Remix
- Y las herramientas de testing Truffle así como sus librerías basadas en Mocha, Chai, etc.

Intentaré detallar los pasos seguidos y las vicisitudes que me he encontrado en el camino para llegar a una aplicación, más o menos funcional, desarrollada en tiempo y plazos, que deberá acabar siendo capaz de interactuar con la red Alastria a través del nodo de la UNIR. Pero también me gustaría dejar constancia de mejoras, nuevas ideas y posibilidades de desarrollo a los que el trabajo me ha llevado y a los que por falta de tiempo no he podido hacer frente, intentaré matizar siempre estas puertas abiertas.

Respecto a esta memoria, intentaré ceñirme al patrón indicado por la UNIR y empezaremos con un pequeño análisis del estado del arte en lo que a Blockchain se refiere.

Posteriormente presentaré detalladamente una descripción de la solución desarrollada a modo también de un gran manual de usuario.

Un pequeño apartado, para mi importante en un momento en el que surgen tantas incertidumbres sobre esta tecnología, en el que intentaré evaluar la justificación del uso de la tecnología blockchain para el caso concreto que abordamos.

Una exposición más o menos detallada de los aspectos tecnológicos en cuanto al análisis funcional y el diseño técnico.

Pasos necesarios para el despliegue y puesta en producción.

Presentación de todo lo referente al plan de pruebas desarrollado.

Y por último cerraremos con un par de conclusiones sobre las que me gustaría dejar constancia y las pocas referencias que he podido incluir en la memoria.

3 Estado del arte

Sin lugar a duda, blockchain creo que será una de las tecnologías que mayor impacto va a suponer en el medio y largo plazo, lo que no tengo claro es el uso real final que llegaremos a darle. Podría llegar a afectar al modelo socioeconómico actual, rompiendo la cadena de valor de uno de los sectores más poderosos en la actualidad, el sistema financiero global, pero también podría afectar a otros sectores como el de la educación, que nos ocupa, en el que el valor aportado no está tan refrendado únicamente en modelos económicos.

Aunque no podemos pretender adivinar el futuro, vamos a repasar un poco la historia de la tecnología y sus herramientas, e intentaremos centrar el foco en la situación actual real.

El 11 de Febrero del año 2009 Satoshi Nakamoto da a conocer sus investigaciones sobre el Bitcoin (<http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source>).

El programador ruso Vitalik Buterin, en un intento inicial de descentralizar la red por un conflicto que le había surgido con un juego online, junto con otros cofundadores, lanza en Febrero del 2014 la primera prueba de concepto de lo que hoy es la red Ethereum, basando su funcionamiento en el de la cadena de bloques que ya se había utilizado en el desarrollo de la red Bitcoin, pero desvinculándolo de las criptomonedas y ligándolo, dado el carácter visionario de Buterín, a los Smart Contracts, objetos vistos como activos digitales que son controlados por organizaciones autónomas descentralizadas. Cabe destacar su pensamiento obsesivo focalizado en la descentralización (<https://futurethinkers.org/vitalik-buterin-ethereum-decentralized-future/>).



Ilustración 1 Ethereum

Cualquier tecnología que se precie, debe ofrecer a los programadores IDEs que les faciliten el trabajo, y el estado del arte de una tecnología, también nos lo da el estado de las herramientas que se utilizan para desarrollarla.

El objetivo de todas estas herramientas es dotar a la comunidad desarrolladora de herramientas que sean capaces de construir secuencias de muy bajo nivel del lenguaje Turing, que es el que interpreta la red Ethereum, a partir de lenguajes de alto nivel más cercanos y sencillos para el programador.

En el ámbito de los Smart Contracts y la red Ethereum/Quorum, tenemos dos grandes aliados en este ámbito:

- **Remix:** Es un IDE de desarrollo de Smart Contracts para la Blockchain Ethereum. Con él se desarrolla en Solidity, un lenguaje de programación de alto nivel (muy influenciado por C++, Python y Javascript). Está accesible en <https://remix.ethereum.net>. Desde este IDE tenemos acceso a los webproviders de las diferentes redes Ethereum (Mainnet, Robsten, Rinkeby o cualquier otro despliegue local que podamos utilizar).
- Otra "suite" que sin lugar a duda nos indica que se trata de una tecnología sana es **Truffle** (<https://www.trufflesuite.com/>). Ofrece varias utilidades, pero en concreto en el proyecto utilizaremos las que son de ayuda en el ámbito de las pruebas unitarias y el servidor local Ganache como veremos ahora.

- Por último, respecto a las utilidades de desarrollo, me gustaría incluir a **Ganache** <https://www.trufflesuite.com/ganache>, que nos permite muy gráficamente y de forma muy sencilla desplegar una red blockchain local. Ofrece una versión en línea de comandos y otra como escritorio. Sin lugar a duda, imprescindible para el desarrollo de este proyecto.

Para entender el actual estado es necesario que veamos los desarrollos posteriores que ha habido entorno a Ethereum y que suponen nuevos ámbitos de trabajo y colaboración que dotan toda la tecnología de contextos de interés común para muchos sectores.

Así, por ejemplo, de la mano de JP Morgan (2016) en el sector financiero, surgió Quorum como la versión empresarial de Ethereum y que originalmente presentaba un carácter muy marcado financiero. Esta solución aporta un alto rendimiento y permite las transacciones privadas entre un grupo autorizado de cuentas. Esta red trabaja principalmente separando los datos privados de lo propio de la red y dispone de nodos con diferentes roles, Constellation (<https://github.com/jpmorganchase/constellation>) y Tesseract (<https://github.com/jpmorganchase/tesseract>).

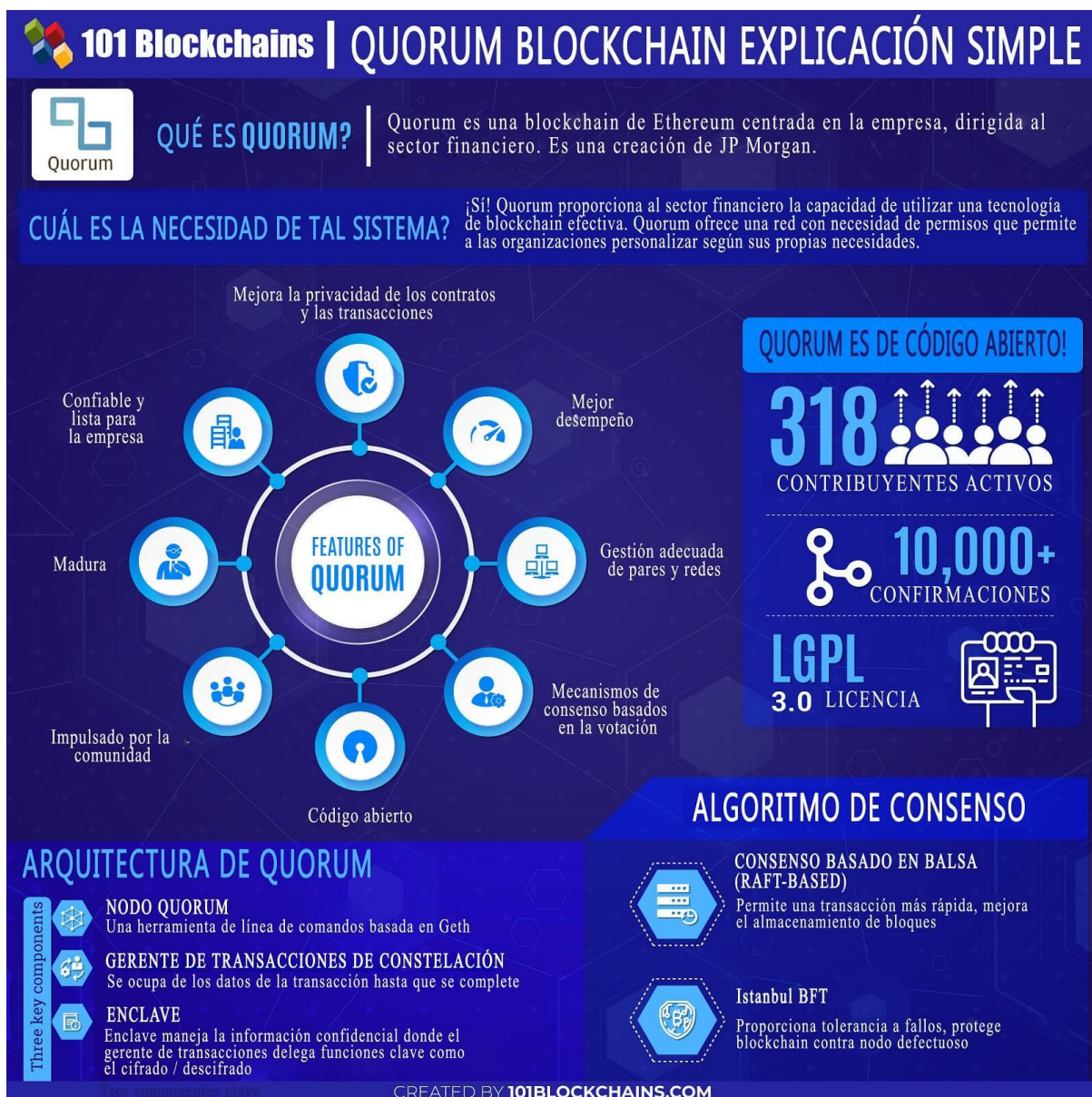


Ilustración 2 Arquitectura Quorum (<https://101blockchains.com/es/quorum-blockchain-guia/>)

A partir de Quorum llegaría Alastría, primer consorcio blockchain, inicialmente español, pero en la actualidad con empresas miembro europeas.

Su propósito se centra en crear una infraestructura para empresas e instituciones. Alastria es semipública, independiente, permissionada y neutral, y está diseñada para ser respetuosa con la legislación europea.

Como objetivo de este proyecto final de curso se incluye el despliegue en la red Alastría, de la que es miembro la UNIR.

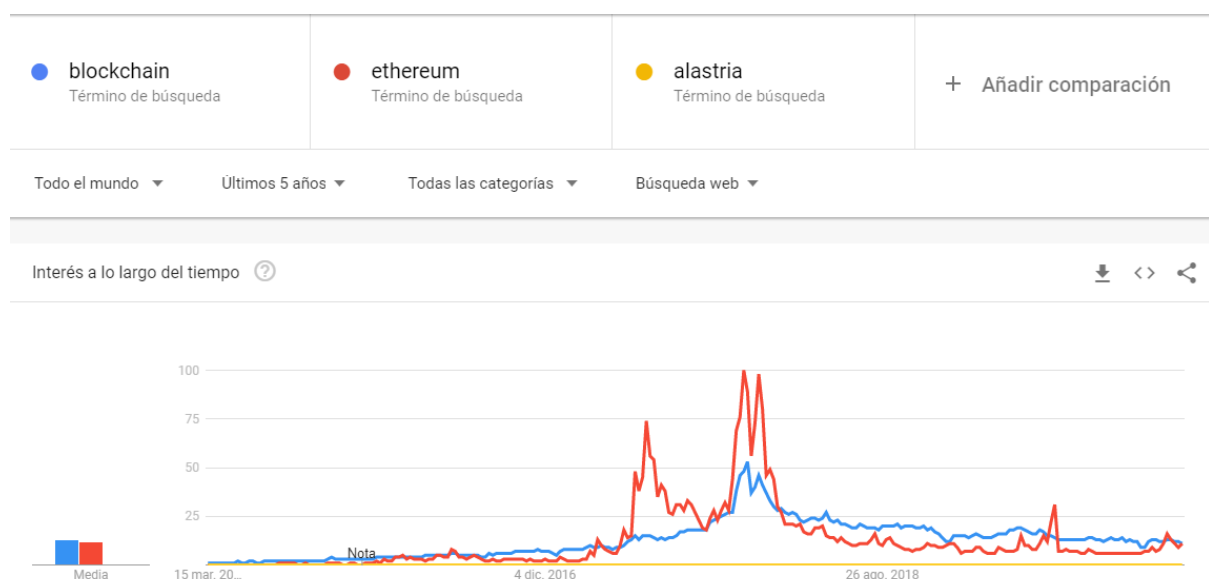


Ilustración 3 Alastria

En el ámbito de las infraestructuras, poco más o menos esto es lo que nos ofrece esta tecnología, suficiente, pero para conocer el verdadero estado del arte necesitamos conocer cual es el sentir del desarrollador y de las empresas que son las que al final acaban pagando los desarrollos.

Me encanta quedarme pensando mientras miro las referencias que Google lanza al buscar sobre alguna tecnología... y en esta ocasión no iba a perder la oportunidad: La palabra "blockchain" ofrece en Google 189.000.000 de referencias, ... interesante... Alastria 96.100, y Ethereum 336.000.000 ...

Echemos un ojo a Google Trends (<https://trends.google.es/trends/explore?date=today%205-y&geo=ES&q=blockchain,ethereum,alastria>)



El interés por estas tecnologías es obvio que no está en sus mejores picos y en los últimos tiempos diría que se encuentra en una situación bastante plana.

Todos los sectores manifiestan su interés por esta tecnología, sobre todo el financiero, pero el desconocimiento, los miedos u otros factores, parece que estén provocando la contención en el despegue de las implantaciones.

Para 2021 se ha planteado el hard fork hacia **Ethereum 2.0**, un paso que probablemente fuerce un nuevo escenario y una nueva situación.

4 El proyecto

El proyecto que se expone atiende al caso número 2 presentado en el ámbito del curso “Experto Universitario en desarrollo de aplicaciones Blockchain”.

El proyecto despliega una **plataforma** donde pueden registrarse **universidades** y **estudiantes**. La plataforma gobierna un token llamado **ECTSToken** que los estudiantes adquieren comprándolos con ether. Las universidades registradas publican su oferta de asignaturas (**SubjectToken**) con precio valorado en ECTSToken. Los estudiantes que quieran matricularse en una asignatura, intercambian sus ECTSToken por matrículas en las asignaturas publicadas en las que están interesados. El modelo económico se cierra con el reembolso de los ECTSToken a las universidades por parte de la plataforma a un precio menor que el coste de adquirirlos por parte de los estudiantes.

Desde el punto de vista técnico de las aplicaciones blockchain, el proyecto se centra en el desarrollo de varios smart contracts donde reside la lógica, los ECTSToken que extienden los tokens **ERC20** y las asignaturas publicadas que extienden el **ERC721**.

Funcionalmente también permite registrar el aprobado de una asignatura por parte de la universidad y abre la puerta a la implementación de un mecanismo de emisión de titulaciones como claims en un modelo de identidad digital basado en ERC725 y ERC735. Lamentablemente la solución se ha tenido que ajustar a los plazos de entrega y este caso de uso se ha dejado fuera del alcance.

El proyecto se desarrollará de acuerdo con tres roles, alumno, universidad y plataforma universitaria, que añadirán al actual modelo de relaciones habitual, un conjunto de evidencias y afirmaciones registradas en la Blockchain.

Los créditos ECTS son una forma de medir el volumen de trabajo que el estudiante debe realizar para superar una asignatura. Su valor es un entero entre 1 y 60. El ECTS se basa en el trabajo total que el estudiante debe realizar y no sólo en las horas de asistencia a determinada asignatura.

Aquí aparecen dos primeras ideas que habría que encajar correctamente junto con el sector para la posible implantación:

- Por una parte, la unificación del precio del ECTS: Al unificar su precio, podría provocar que ya no tuviera que ver tanto con la carga de trabajo del estudiante, sino más bien con la carga de trabajo que la asignatura supusiera a la universidad. Creo que esta memoria no es el lugar para tratar este tema, pues no conozco el funcionamiento del sector, pero son temas que dejo en el tintero y que desde un punto de vista funcional deberían ser consensuados. En el proyecto consideraré en todo momento que el hecho de que todos los ECTS tengan el mismo valor está aceptado y consensuado.
- Por otra, la tokenización de un ECTS como un token ERC20 creo que aportaría importantes ventajas a nivel de transferencia, compatibilidad, homologación, etc

Otra pieza importante del proyecto son las "asignaturas" que tokenizaremos como SubjectToken, un smart contract que hereda del ERC721. La tokenización de asignaturas parece algo mucho más consensuable y nutriría los sistemas actuales con un mecanismo de evidencias fiable, persistente y escalable.

Los SubjectTokens favorecerían numeros procesos de difícil gestión actualmente como el traslado de expedientes, la justificación de asignaturas aprobadas, etc

He planteado también en el SubjectToken desarrollado, la posibilidad de que cada token almacene las actividades relacionadas con el estudiante en esa asignatura, como exámenes, actividades, etc y sus fechas y evaluaciones. Lamentablemente he desarrollado únicamente los smart contracts, pero por falta de tiempo he dejado fuera la implementación del frontend.

Tras aprobar todas las asignaturas, parece inmediato que el alumno solicite con agrupaciones de asignaturas aprobadas los títulos correspondientes y a partir de ello la emisión de un claim por parte de la identidad digital de la universidad en favor del alumno. Dicho claim quedaría unido finalmente a la identidad digital del alumno. Esto ligaría la solución planteada con los estándares de identidad digital **ERC735 y ERC725**, pero nuevamente por problemas de plazos me he visto obligado a dejar esta funcionalidad fuera del alcance de esta versión.

5 Justificación del uso de blockchain

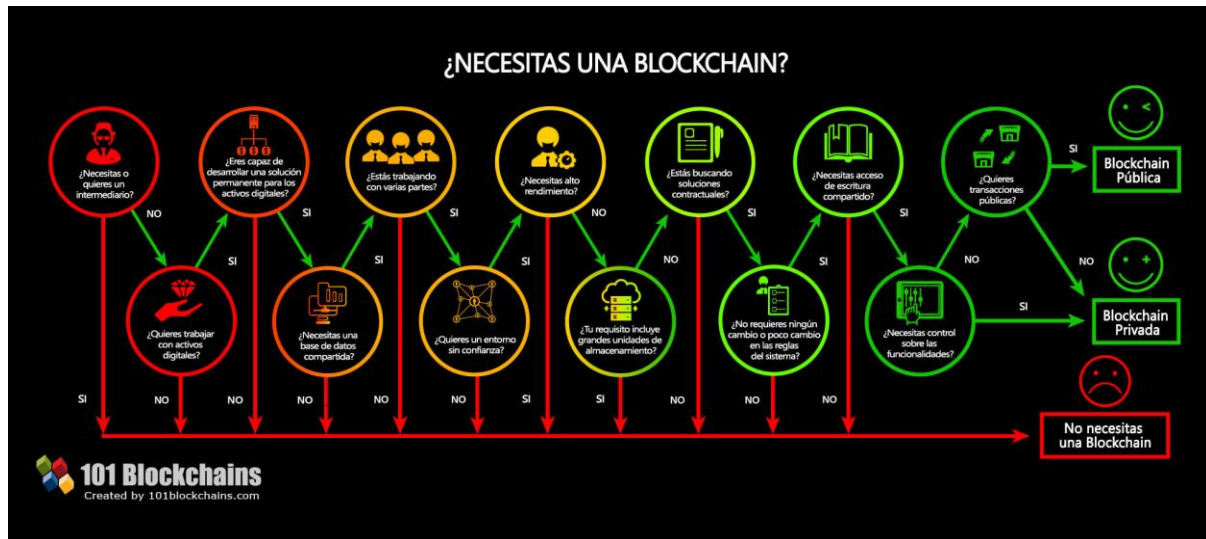


Ilustración 4 ¿Necesitas un Blockchain? (<https://101blockchains.com/es/necesitas-una-solucion-blockchain/>)

Nunca es fácil responder la pregunta ¿necesitas una blockchain?. La gráfica que muestro arriba de 101blockchains.com me ha resultado bastante clara <https://101blockchains.com/wp-content/uploads/2018/12/Necesitas-una-blockchain.jpg>

En mi opinión los patrones habituales de decisión son demasiado flexibles y eso hace que sea muy difícil responder.

Hay cuestiones sobre las que cualquiera se preguntaría, cuya respuesta es evidente y nadie la cuestionaría, pero existen otras mucho más complejas, por ejemplo ¿necesitas una base de datos? Si respondo que mi aplicación necesita una base de datos y eso ya me lo ofrecen otras tecnologías, ¿tengo que olvidarme de blockchain?

En mi opinión, probablemente la respuesta está en el gris, al igual que en la actualidad prácticamente todos los grandes sistemas contemplan sistemas relacionales y otros NoSQL y hemos llegado al consenso de que lo mejor es guardar en cada base de datos la información más adecuada haciendo convivir todas las fuentes de datos, creo que en el futuro pasará lo mismo con blockchain, y algunos de los datos de una aplicación serán almacenados por sus características y lo que esperamos de ellos en la Blockchain, y otros en cambio, que nos lo permitirán, lo harán sobre bases de datos SQL/NoSQL.

Le he dado muchas vueltas a la idea de lo que me aportaba la blockchain en este caso y obviamente dado que se trata de un ejercicio académico y por tanto la libertad de diseño y recursos está limitada, no podría complementar la aplicación con un modelo de datos relacional, pero la convivencia de tipos de datos de la que hablaba antes, creo que sí sería operativa en este caso.

Las cuestiones que podríamos plantearnos son:

- ¿Una base de datos tradicional (relacional o no relacional) es insuficiente para cubrir las necesidades del proyecto? Sí, porque nos gustaría eliminar a los intermediarios y descentralizar totalmente ciertos aspectos de nuestro sistema.
- ¿Podemos identificar varios escritores? Obviamente, como ya hemos comentado trabajamos con tres roles, alumnos, universidades y plataforma universitaria.
- ¿Podemos replicar los datos? No tenemos limitaciones por el hecho de que la información sea confidencial y no pueda estar replicada en diversos nodos.
- ¿Nuestro sistema requiere un alto rendimiento y una elevada cantidad de información? Este aspecto me sugiere dudas. En las partes que he dejado fuera del alcance, como comentaba, el tratamiento de actividades y valoración de las mismas por asignatura, podría suponer un incremento importante de información, y por tanto en este aspecto me cuestionaría la idoneidad de la tecnología, pero en el límite que he fijado al alcance, creo que no alcanzamos una gran cantidad de información como una necesidad.
- Otro aspecto que se suele cuestionar es la desconfianza entre las partes. En el actual sistema vigente, un empleado de la universidad podría por ejemplo manipular las bases de datos para, a priori o posteriori, modificar manualmente la información.
- ¿Nuestro sistema requiere información inmutable? Por supuesto, si he aprobado una asignatura, es impensable que con el tiempo esto suscite dudas. Nada mejor que una blockchain para que esto fuese perpétuo.
- ¿Eliminamos los intermediarios? Obviamente de esta forma estamos eliminando la necesidad de los intermediarios, pues los tokens de los estudiantes o los ECTSToken recaudados por las universidades serán propiedad de ellos sin necesidad de que nadie se convierta en un banco intermediario para ello.

A partir de este pequeño análisis, podemos concluir que este sistema sí se vería beneficiado por el uso de una cadena de bloques, no obstante, me gustaría resaltar que se trata de un ejercicio académico limitado en recursos y por tal motivo el diseño que he planteado probablemente no sería el que usaría en un proyecto real, donde combinaría todo el sistema con otras fuentes de datos, aunque perdería en algún grado la descentralización.

6 Análisis y diseño técnico

Como hemos comentado antes, se interactuará con el sistema con uno de los siguientes roles:

- Plataforma universitaria
- Universidad
- Estudiante

Aunque no hemos incidido en este aspecto y probablemente sea fácilmente complementable con otras fuentes de datos e incluso vinculable al concepto de identidad digital como planteaba en capítulos anteriores, la universidad quedaría definida por:

- Una cuenta de Alastria (una dirección pública válida) como identificador único.
- Y su nombre

Insisto, esto podríamos ampliarlo muchísimo, añadiendo por ejemplo la dirección física, etc., pero estaríamos ampliando las necesidades de volumen de información de nuestro sistema, y en estos casos quizá el diseño más conveniente pudiera ser complementar la información de la cuenta como identificador único, con un sistema de base de datos relacional ajeno a la blockchain. Dejo este aspecto como un punto abierto en el diseño.

Para el estudiante, igualmente:

- Una cuenta de Alastria (una dirección pública válida) como su identificador único.
- Y su nombre.

La plataforma universitaria únicamente cuenta con su cuenta de Alastria, que le permitirá desplegar los Smart Contracts que implementarán la infraestructura necesaria para todo lo demás. La plataforma se convierte en el *owner* de esa infraestructura primaria que constituirá el core de la aplicación y sólo ella, como propietaria, podrá interactuar con los aspectos gestores de dicha infraestructura.

Como vemos, nuestro sistema requerirá al menos 3 cuentas en Alastria, una para la plataforma, otra para al menos una universidad y la tercera para el estudiante. Durante todo el proyecto consideraremos la siguiente asignación:

- accounts[0] es la cuenta administradora de la plataforma.
- accounts[1] cuenta que representa la universidad.
- accounts[2] cuenta que representa al usuario.

Aunque en las pruebas la asignación ha sido realizada de esta forma, lo único cierto en una implantación es que la cuenta de la plataforma será la accounts[0], el resto de cuentas dependerán del orden de registro en la plataforma.

La infraestructura core estará formada por 2 Smart Contracts que se relacionarán entre sí:

- Un Smart Contract que representará el valor de una asignatura, los ECTS, al que llamaremos ECTSToken.
- El Smart Contract que controla los grupos de estudiantes y universidades registrados en el sistema e inicia y cierra el business case.

Aunque en el experto es un tema que no hemos tratado mucho, he querido darle cierta importancia dentro de mi proyecto; creo que no podemos plantearnos la viabilidad de un sistema sin tener en cuenta los aspectos económicos del mismo.

En este aspecto el workflow de valor económico sería el siguiente:

1. El estudiante adquiere en la plataforma los ECTSToken pagando con su ether.
2. Con sus ECTSToken paga las matriculaciones en las asignaturas de su interés.
3. Cuando se matricula en una asignatura, el valor se ha transferido del estudiante a la universidad que publicó esa asignatura.
4. Por último, la universidad puede reembolsar sus ECTSToken contra la plataforma con el intercambio inverso, ECTSToken por ether.

Mi planteamiento, y esto es una decisión funcional, ha sido optar por

El precio de compra del ECTSToken por parte del estudiante es superior al precio de reembolso de esos ECTSToken que cobra la universidad, de esta forma la plataforma es sostenible económicamente.

Dado que el ECTSToken es un token que tal y como se ha planteado no variará su valor en función de su escasez, he considerado que no hay necesidad de hacer un "*mintado*" masivo por parte de la plataforma inicialmente, y es suficiente con mintar los ECTSToken necesarios a medida que el estudiante los demanda.

Respecto a las asignaturas, desde un principio el proyecto me supuso muchísimas dudas, bien por desconocimiento del negocio o por desconocimiento de la tecnología, más bien me inclino por lo primero, los posibles enfoques me parecían correctos y no me decidía por uno como más adecuado.

La exposición de los distintos enfoques en este ámbito del mentor, Iñigo García de Mata, fue clara y dejó las posibilidades reducidas a dos enfoques:

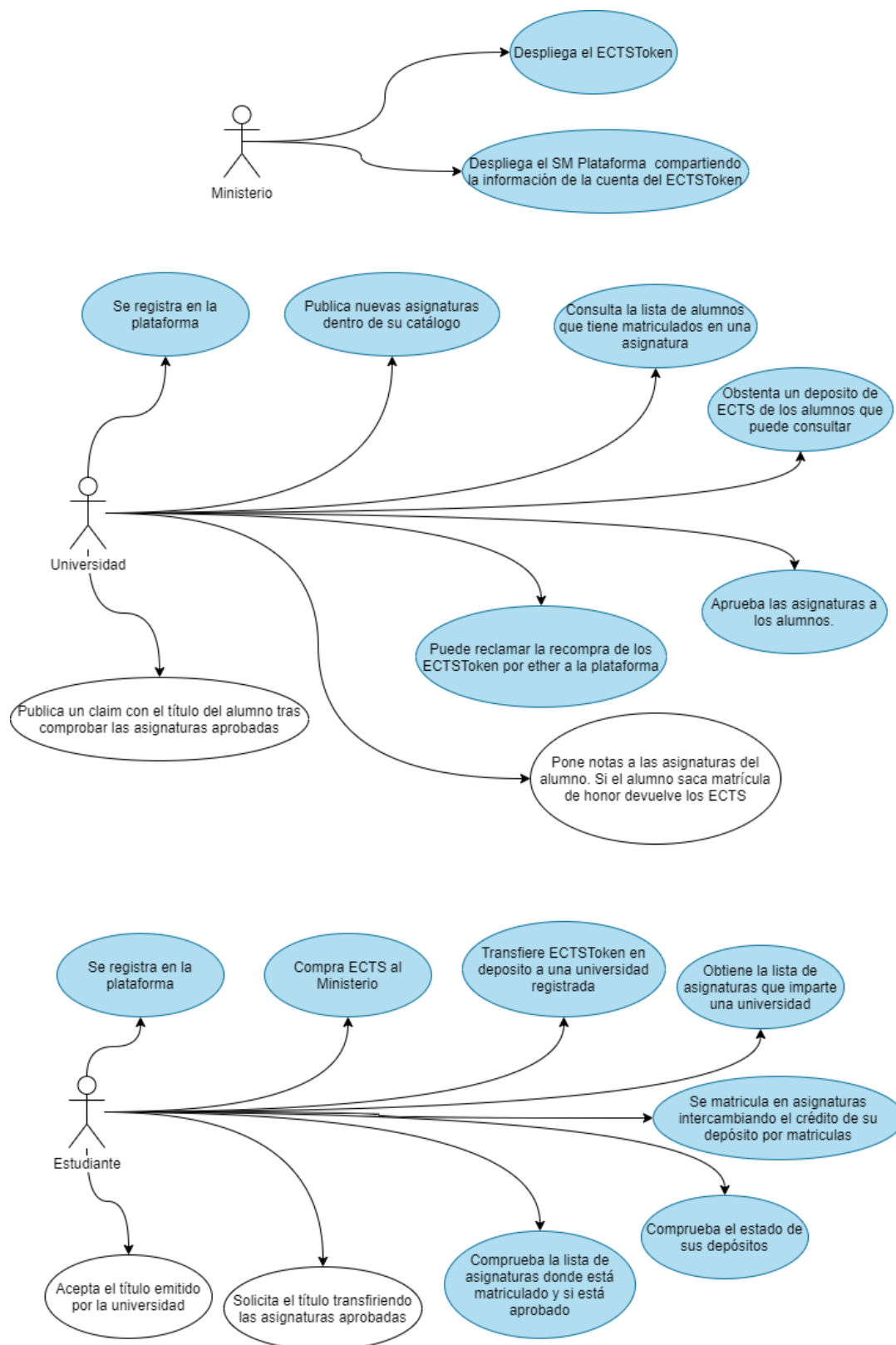
- La universidad como emisora de un token que representaba las asignaturas
- O la plataforma universitaria como emisora de dichos tokens.

Finalmente opté por la primera idea, justificándome en la escalabilidad por volumen de dicho sistema. No es lo mismo que un token soporte la carga de todos los universitarios de España, que el hecho de que por cada asignatura de cada universidad tengamos que soportar la carga de sus propios matriculados.

A partir de esta decisión, aparece un token interoperable para uso interno en la relación universidad/estudiante y que supone una garantía como evidencia de la matrícula de un estudiante en una asignatura y el posterior aprobado del mismo.

Por las limitaciones de tiempo no he cubierto otras posibilidades que se plantean como sería la devolución de los ECTSToken a los estudiantes con matrícula de honor en una asignatura, esto complicaría la lógica muchísimo para estos plazos, pero es cierto que este sistema es tremendamente potente en los aspectos relacionados con la propiedad del ECTSToken y las posibilidades que abre el uso de la blockchain.

Tras estas decisiones previas, llegamos al diagrama de casos de uso para identificar claramente las acciones que puede realizar cada uno de los roles.



Las acciones que se han dejado en blanco en el diagrama indican que no han podido ser abordadas dentro del TFE por dificultades de plazos, aunque me hubiese gustado que fuera de otra forma. En cualquier caso, se han cubierto la mayoría de ellas.

A continuación, se muestra el diagrama de secuencia de todo el flujo económico por considerarlo el más amplio y representativo.

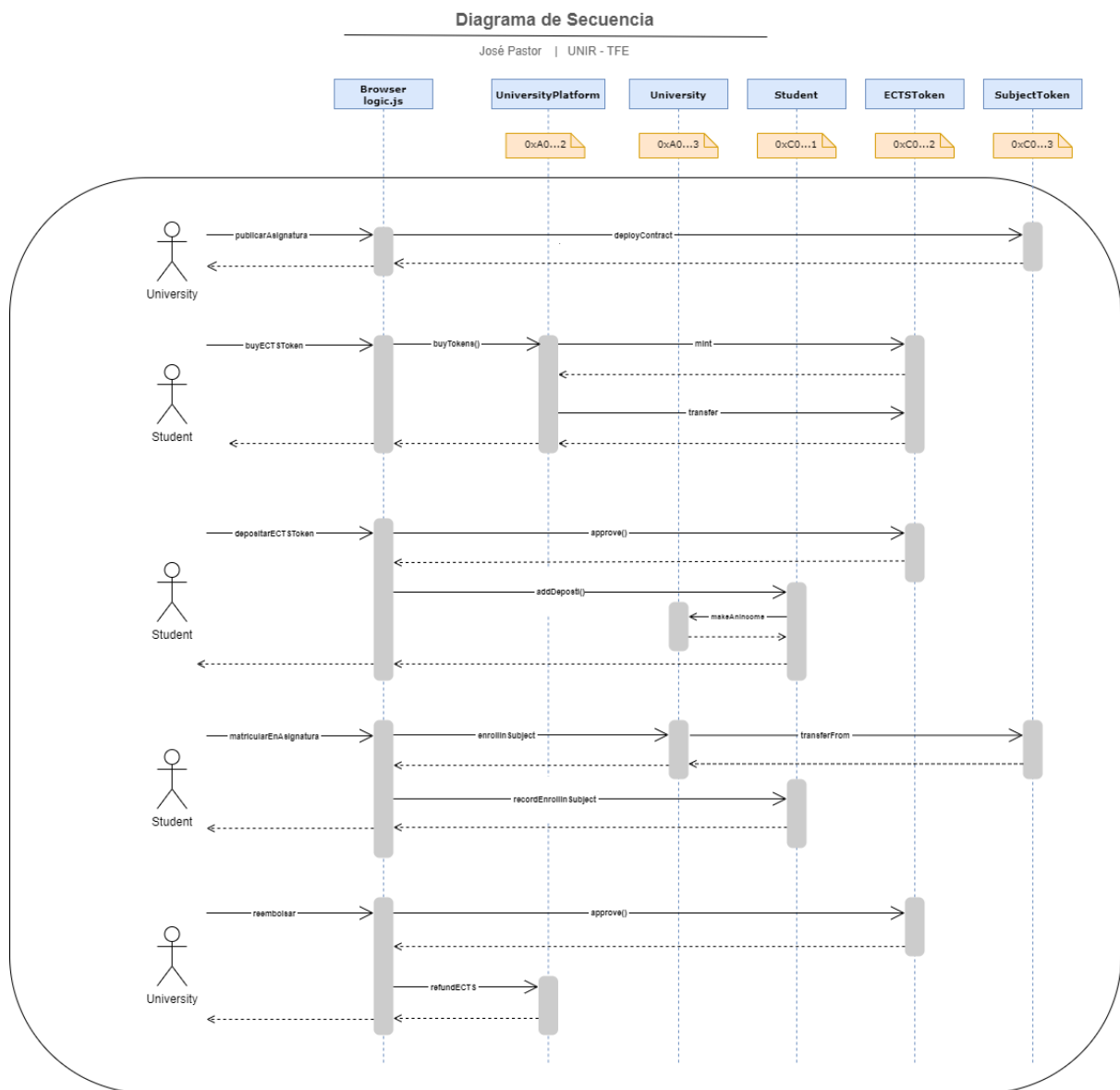


Ilustración 5 Diagrama de secuencia

Si no se aprecia, puede consultarse la versión original en <https://github.com/jpges/UNIR-TFE/blob/master/docs/DigramaSecuencia.png>

Arquitectura en 3 capas

Aunque no se puede hablar de arquitectura Modelo-Vista-Controlador ni nada similar tradicionalmente hablando, siguiendo un poco los patrones que hemos establecido en las prácticas realizadas durante el curso, he mantenido esta estructura de componentes en la que:

- Hablamos de modelo como todo aquello que almacena cosas en la Blockchain.
- La lógica de nuestra aplicación la hemos mantenido en los ficheros javascript.
- Y, por último, la vista incluiría todo el HTML y estilos relacionados que ve el usuario en su navegador.

No voy a ocultar que al igual que pasaba en el modelo tradicional, la tendencia inmediata a pasar lógica desde el javascript al modelo Blockchain es innegable y en numerosas ocasiones he dudado entre diferentes opciones por intentar dotar al sistema, un sistema no transaccional por otra parte, de cierta atomicidad. Pensemos que una transacción fraccionada en diversas operaciones sobre la Blockchain implica la imposibilidad de rollback en alguna operación simple anterior, de ahí mi tendencia a agrupar operaciones buscando esa atomicidad y con ello la deslocalización de la lógica de la capa javascript a la blockchain. Esto último no lo considero adecuado, aunque comprendo y creo que hay mucho que debatir en esta área. Supongo que en un primer paso habría que buscar una transaccionalidad básica con operaciones muy simples.

A continuación, se muestra un diagrama de las piezas desarrolladas atendiendo a esta arquitectura. Se han coloreado los componentes de acuerdo con esta leyenda:

- Azul: Elementos que utiliza el role universidad.
- Verde: Elementos que utiliza el role estudiante.
- Naranja: Elementos que utiliza el role plataforma.
- Blanco: Elementos comunes que utilizan indistintamente.

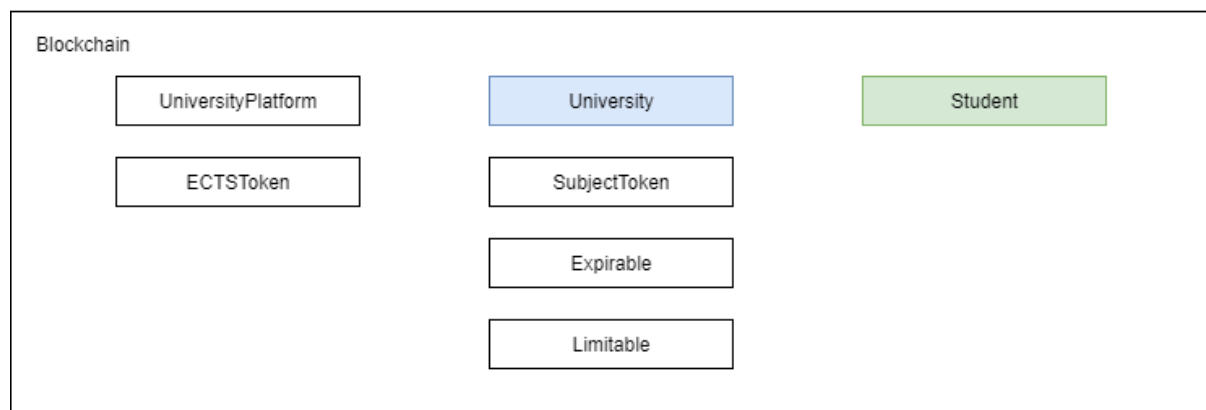
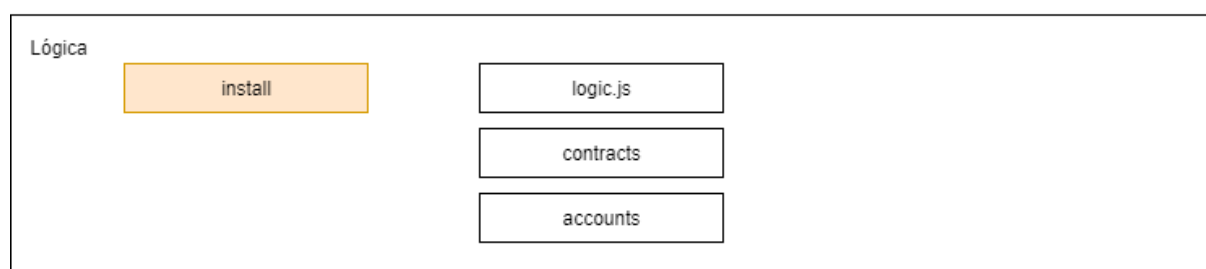
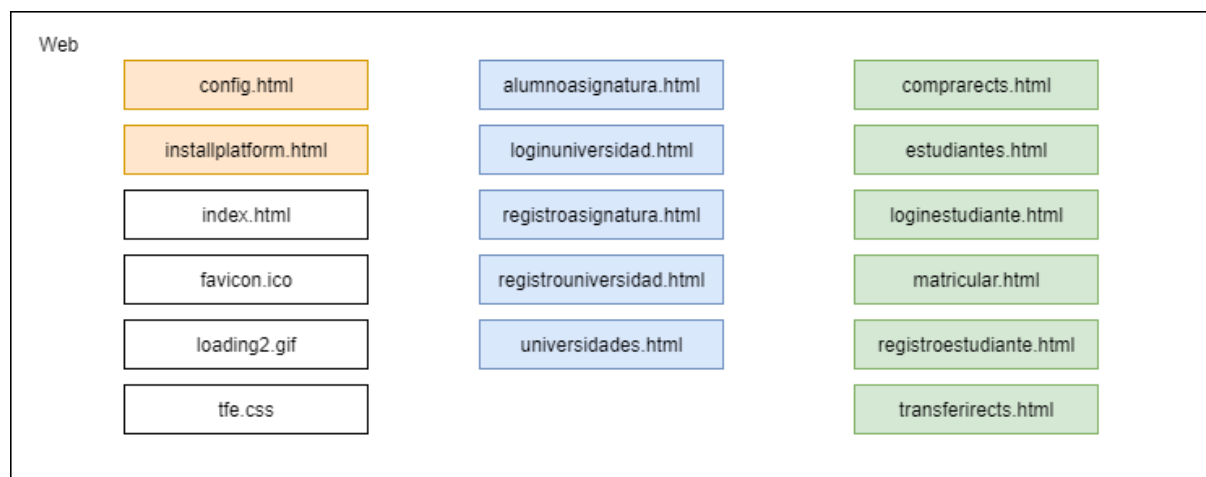


Ilustración 6 Arquitectura 3 capas

Arquitectura Frontend

La interfaz web ha sido desarrollada en HTML, CSS y JavaScript. Una primera tarea fue elaborar el mapa de navegación que expongo a continuación y que daría aspecto a toda la funcionalidad.

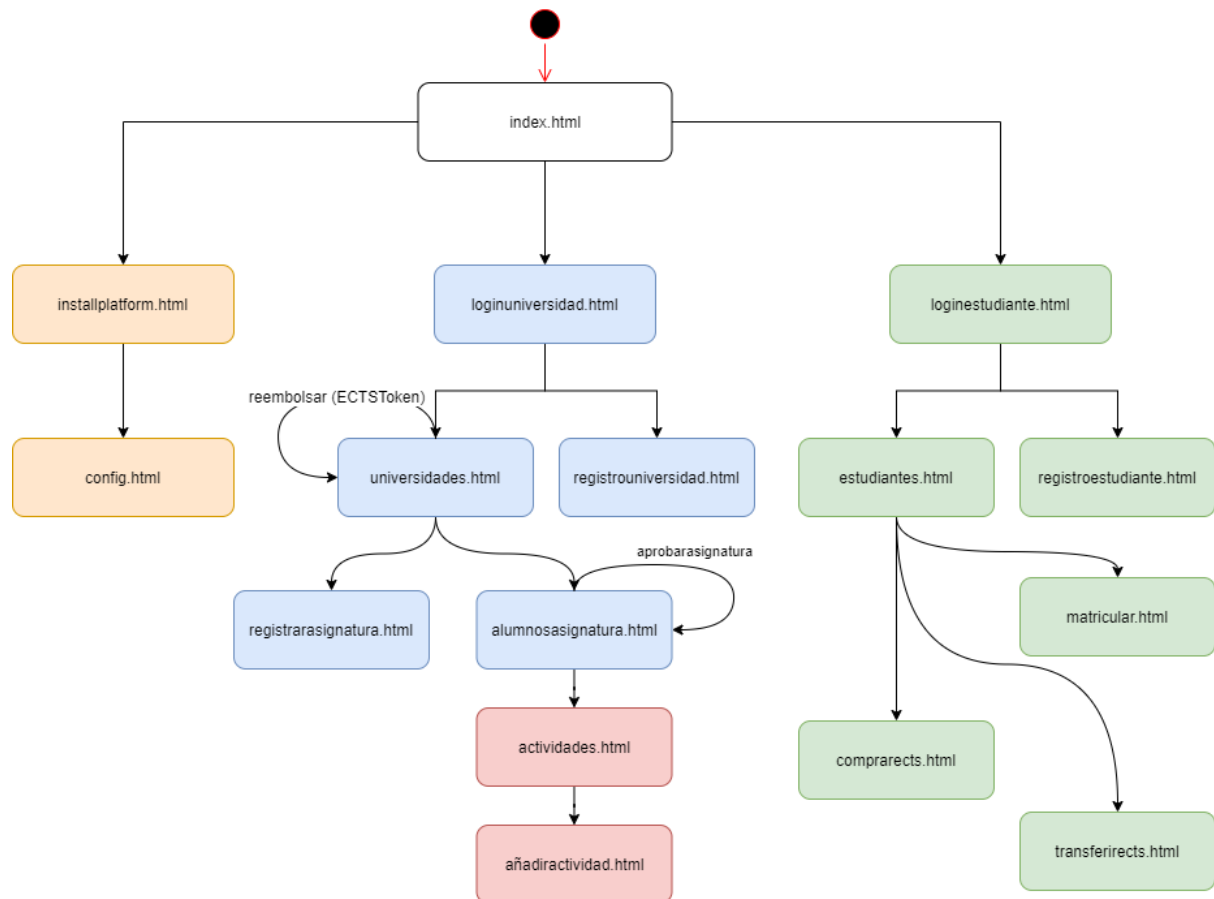


Ilustración 7 Mapa de navegación

En este mapa se exponen las diferentes páginas coloreadas de acuerdo con:

- Naranja: Páginas web utilizadas por el role plataforma.
- Azul: Páginas web utilizadas por el role universidad.
- Verde: Páginas web utilizadas por el role estudiante.
- Blanco: Páginas web comunes.
- Rojo: Páginas web que no han sido desarrolladas dentro del alcance del TFE.

También se preparó algún diseño general del estilo a seguir por las páginas, pero creo que es innecesario mostrarlo en esta memoria, no obstante, puede consultarse en el repositorio del proyecto.

Adicionalmente a estos elementos que se han desarrollado para el proyecto, se utilizan librerías y los siguientes componentes de terceros:

- jQuery 3.2.1: Que utiliza alguna de las funciones que se han desarrollado.
- Web3.js 1.2.6: Que se utiliza para realizar todas las comunicaciones con la Blockchain.
- Bootstrap.css 4.4.1: Que aporta el estilo general de la aplicación, adicional al estilo particular de algunos elementos que aporta tfe.css.
- Fontawesome.css 5.6.3: Para incluir algunos iconos en el diseño de la aplicación.

No se han utilizado validadores desarrollados a medida, pero aprovechan el propio framework Bootstrap que se ha incluido, se han utilizado correctamente las clases de validación para que esta librería nos aporte este comportamiento localmente, así por ejemplo si un campo es requerido la propia infraestructura controlará todos estos aspectos.

Capa lógica

Como se ha comentado con anterioridad, la lógica ha intentado localizarse en varios archivos JavaScript que agrupan las operaciones.

A grandes rasgos, las tareas que agrupan los ficheros son las siguientes:

- Install.js: Incorpora las funciones necesarias para la instalación de la plataforma.
- Config.js: Incorpora únicamente las variables básicas que el sistema necesita para arrancar. Se genera en la instalación.
- Accounts.js: Agrupa las funciones relativas a las cuentas, que son dependientes de la red en la que se encuentre desplegada la solución.
- Contracts.js: Únicamente contiene variables globales que informan de los ABI y el DATA de los diferentes Smart Contracts.
- Logic.js: Agrupamos en este fichero todas las funciones que se corresponden propiamente con la lógica de la aplicación.

Para una mayor seguridad y sobre todo por eficiencia, se ha realizado comprobaciones previas antes de enviar peticiones a la Blockchain para ahorrar llamadas innecesarias. Así por ejemplo se controlará en el código que la cantidad de tokens que se quieren transferir sea inferior a la cantidad de tokens que tenemos en propiedad y cosas de este estilo.

Rompiendo un poco la tónica de las practicas realizadas durante el curso, el resultado de las peticiones realizadas a la blockchain desencadenará navegación real sobre la aplicación, al margen de que se ha mantenido un log con carácter "verboso" sobre la consola del navegador para poder controlar todo lo que está sucediendo en el código. La consola puede consultarse en cualquier momento durante la ejecución.

Arquitectura Backend

Comenzaré mostrando el diagrama de clases general de los Smart Contracts desarrollados.

Dado que el tamaño deja mucho que desear para una lectura cómoda, se recomienda acudir a la fuente original en

https://github.com/jpges/UNIR_TFE/blob/master/docs/DiagramaClases.pdf

El código ha sido documentado a conciencia, por lo que creo que es autoexplicativo, pero vamos a revisar en este apartado la utilidad y patrones seguidos en su diseño.

Como puede verse en el diagrama, he extendido diversos contratos del paquete OpenZeppelin:

- SafeMath: para utilizar uint256 de forma segura evitando el overflow.
- ERC20Detailed: Aporta variables adicionales como el nombre o el símbolo al ERC20.
- ERC20: El token estándar para tokens fungibles.
- ERC721: Implementación básica del token no fungible.
- ERC721Metadata: Una extensión del ERC721 que nos aporta variables como el nombre, símbolo, etc.
- Ownable: Es un módulo que nos aporta un mecanismo básico de control de accesos únicamente para el propietario de la clase.

Revisaremos los aspectos más destacables de cada uno de los Smart Contracts diseñados expresamente en el TFE.

[UniversityPlatform.sol](#)

Se trata de uno de los SC principales de la solución. Se encarga principalmente de mantener los registros de universidades y estudiantes. También es el responsable del gobierno de los ECTSToken, los mintea, vende a los estudiantes y reembolsa a las universidades.

En los arrays privados `_listUniversities` y `_listStudents` guarda la lista de cuentas registradas y para acceder rápidamente a la información de estudiantes y universidades mantiene un par de mappings contra las direcciones SC correspondientes.

```
// Guardamos la información de las universidades como un mapping de cuentas contra las instancias del SmartContract
mapping(address => address) private _universities;
// Listado rápido de todas las address de las universidades
address[] private _listUniversities;

// Guardamos la información de los alumnos como un mapping de cuentas contra las instancias del SmartContract
mapping(address => address) private _students;
// Listado rápido de todas las address de los estudiantes registrados
address[] private _listStudents;
```

Dispone de un par de métodos para modificar los precios de compra y venta de ECTS, aunque esa funcionalidad no ha sido implementada en el front.

Implementa también los métodos necesarios para adquirir tokens ECTSToken:

```
/**
 * @dev Es la función que sirve para comprar tokens por parte del estudiante.
 * Requiere que el que compra sea un estudiante registrado en la plataforma.
 * Emite un evento {BuyTokens} con `tokens` comprados al precio {priceOfOneTokenInWei} por {beneficiary}.
 * Los tokens se mintan en el momento en el que son adquiridos.
 * @param beneficiary Es la cuenta que adquiere los tokens
 */
function buyTokens(address beneficiary) public payable {
    require(
        beneficiary != address(0),
        "buyTokens: Beneficiary is account 0."
    );
    require(msg.value != 0, "buyTokens: value is 0.");
```

Y para solicitar el reembolso de los ECTSToken por parte de las universidades:

```
/**
 * @dev Recibe la petición de reembolso de ECTS desde una universidad que los ha aprobado previamente
 * Emite un evento ECTS reembolsados {refundECTS}
 * @param account Cuenta a la que hay que enviar el ether
 * @param amount Cantidad de ECTS que se quieren compensar
 * @return uint256 Cantidad de wei recompensado
 */
function refundECTS(address account, uint256 amount)
    public
    onlyRegisteredUser
    returns (uint256)
{
    _token.transferFrom(account, address(this), amount);
    uint256 amountWei = _priceOfOneTokenInWeiForRefund.mul(amount);
    address payable accountpayable = address(uint160(account));
```

Aunque lo veremos más tarde en un pequeño apartado dedicado a la privacidad, implementa un modificador que garantiza que el que ejecuta un método es un usuario registrado de la plataforma, estudiante, universidad o la propia plataforma.

University.sol

Sin entrar en excesivos detalles pues el código se encuentra suficientemente comentado, define una estructura para almacenar los depósitos de ECTSToken que los estudiantes dejan en las universidades.

La idea de los depósitos surge de la necesidad de matricular en las asignaturas con la seguridad de tener los ECTSToken como propiedad de la universidad previamente, algo a modo de lo que hicieron los exchangers para gestionar los depósitos de sus clientes. En la actualidad existen ERCs como el ERC223 (<https://github.com/Dexaran/ERC223-token->

[standard](#)) que permitiría realizar este tipo de transacciones de una cosa por la otra con seguridad, pero no he querido usarlos porque me ha parecido que era más conveniente focalizarnos sobre lo tratado en el curso diseñando soluciones para hacerlo viable con ello.

Se define una estructura para mantener un mapping de los depósitos:

```
struct StructDeposit {  
    address accountstudent;  
    string studentname;  
    uint256 balance;  
    bool exists;  
}
```

Y diversos mapeos y arrays que nos ayudarán a acceder rápidamente a la información, en este caso de asignaturas publicadas y depósitos gestionados:

```
// Guardamos las asignaturas emitidas por la universidad  
address[] private _subjectsUniversity;  
mapping (address => bool) private _subjects;  
  
// Guardamos un deposito de tokens ECTSToken para cada estudiante que lo adquiere  
address[] private _deposits;  
mapping (address => StructDeposit ) private _universityStudentBalances;
```

Student.sol

Muy similar al caso de las universidades, pero en este caso para estudiantes.

En este caso se controlan dos estructuras, una que mantiene los depósitos realizados por el estudiante:

```
//Estructura para guardar depositos  
struct Deposit {  
    uint256 balance;  
    bool exists;  
}
```

Y otra que se corresponde con las asignaturas en las que se ha matriculado:

```
//Estructura para guardar matriculas
struct Enrollement {
    uint256 tokenid;
    bool exists;
}
```

En ambos casos se ha utilizado el recurso “exists” para consultar fácilmente la existencia de un valor, evitando tener que iterar sobre una lista.

Al igual que en los casos anteriores se han utilizado algunos arrays y mappings para mejorar la eficiencia en las búsquedas:

```
// Guardamos las asignaturas en las que el estudiante está matriculado
address[] private _subjectsInWitchEnrolled;
mapping(address => Enrollement) private _subjectsEnrollements;

// Guardamos el deposito de tokens ECTSToken que tenemos en cada universidad
mapping(address => Deposit) private _universityDepositBalance;
mapping(uint256 => address) private _universitiesWithDeposit;
uint256 private _totalDeposits;
```

Para depositar los ECTSToken, igual que para recuperarlos, en los depósitos de las universidades, se ha seguido un patrón de doble transacción bastante común en este tipo de diseños, lo que se conoce como el “**Escrowing ERC20 tokens**”, más o menos algo así como “*El deposito de tokens ERC20*”.

La idea entorno a esto surge de la necesidad de que el contrato que actuará como deposito se entere de cuando ha recibido una nueva transferencia de tokens.

La solución es crear dos transacciones originadas por el que transfiere los tokens. En una primera transacción, el emisor aprueba al receptor para la cantidad exacta de tokens que quiere depositarle

```
ERC20(tracker_0x_address).approve(address spender, uint tokens)
```

La segunda transacción consiste en llamar a un método del contrato deposito que gestionará la recepción de los tokens y los registrará:

```
mapping ( address => uint256 ) public balances;
deposit(uint tokens) {
    // add the deposited tokens into existing balance
    balances[msg.sender]+= tokens;
    // transfer the tokens from the sender to this contract
    ERC20(tracker_0x_address).transferFrom(msg.sender, address(this), tokens);
}
```

De esta forma se actualiza el balance del depositario en el contrato que controla el depósito.

Este tipo de patrones podemos encontrarlo en el fichero logic.js, por ejemplo en el método

```
function depositarECTSTokens(studentname) {
    let cantidadECTS = document.getElementById("cantidad").value;
    let accountSCUniv = document.getElementById("universidad").value;
    let accountStudent = localStorage.getItem("accountStudent");
    let accountSCStudent = localStorage.getItem("accountSCStudent");
    let SCStudent = new web3.eth.Contract(ABI_Student, accountSCStudent);

    return SPECTSToken.methods.approve(accountSCUniv, cantidadECTS).send({
        gas: 5000000,
        from: accountStudent
    }).then(function (result) {
        return SCStudent.methods.addDeposit(studentname, accountSCUniv, cantidadECTS).send({
            gas: 5000000,
            from: accountStudent
        }).then(v => {
            console.log("Ejecutada transferencia de deposito.");
            console.log(v);
        });
    });
}
```

En este caso vemos que se llama al propio contrato Student, pero se debe a que él lleva su registro propio de depósitos y llamar al contrato University para avisarle de la transferencia.

ECTSToken.sol

No voy a entrar en más detalles porque simplemente se trata de la extensión de un ERC20 que garantiza que el único que minte el token es el propietario

```
*/
function mint(address beneficiary, uint256 amount) public onlyOwner {
    _mint(beneficiary, amount);
}
```

SubjectToken.sol

Básicamente se trata de un ERC721 que representa una asignatura.

Como características propias cabe destacar la utilización de ERC721Metadata para dotarle de una estructura de información más completa.

Así mismo se ha desarrollado un módulo, llamado **Expirable.sol**, siguiendo el patrón del modificador Ownable de OpenZeppelin, que aporta fecha límite de mintado a nuestro SubjectToken. De esta forma, por extender Expirable, las asignaturas estarán dotadas de una fecha máxima de matriculación.

```
/**
 * @dev Contrato que añade un modificador que nos permite autorizar el uso de
 * por ejemplo una función durante un tiempo desde que se despliega el smart
 * contract.
 */
contract Expirable {
    uint256 private _expirationtime;
```

Obviamente, aunque hablamos en tiempos de bloque, para la precisión que requiere la matriculación en una asignatura es más que suficiente.

Un patrón similar se ha seguido con el módulo **Limitable.sol**, que también es extendido por subjecttoken. En este caso nos aporta la posibilidad de mintar un número máximo de tokens de un determinado ERC721.

```
/**
 * @dev Contrato que añade a los tokens ERC721 un mecanismo para controlar
 * el número de tokens mintado y limitar el mintado por encima de un numero
 * controlado inicialmente.
 *
 * Principalmente lo he desarrollado para crear el modificador "belowLimit" y
 * utilizarlo antes de mintar.
 */
contract Limitable {
    uint256 private _minted;
    uint256 private _limit;
```

Aplicado a las asignaturas nos permitiría limitar el número máximo de matriculas que queremos ofrecer de una asignatura.

Seguridad y privacidad

Uno de los primeros aspectos en los que se pensó era la privacidad y para ello:

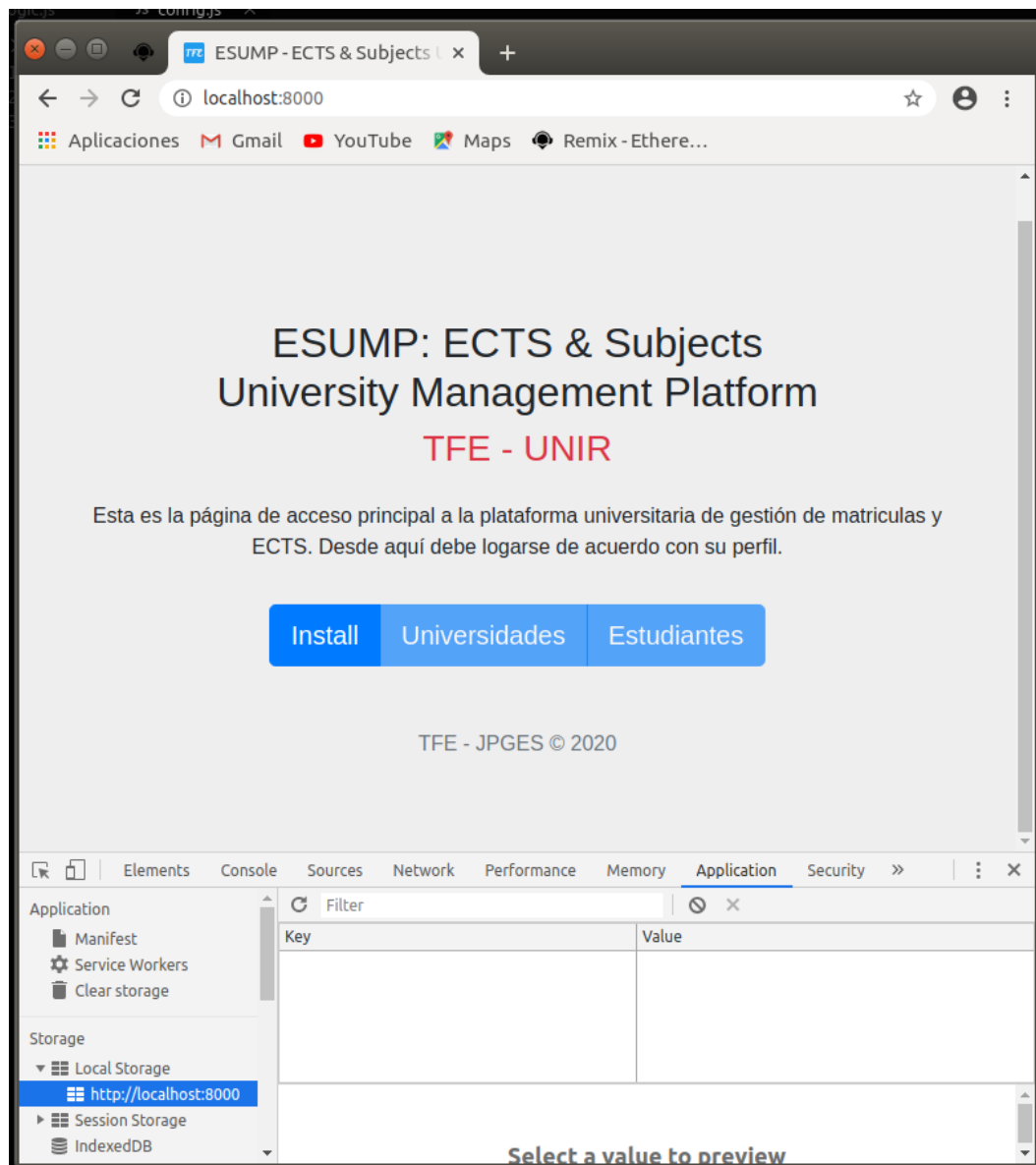
- Se ha hecho un uso intensivo del Smart Contract Ownable (<https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/ownership/Ownable.sol>) que nos permite controlar en todo momento la capacidad de interacción con los diversos métodos únicamente al propietario mediante el modificador *"onlyOwner"*.
- Y por otra se ha desarrollado un modificador a medida para la plataforma *"onlyRegisteredUser"* que garantizará que los métodos sean utilizados por estudiantes, universidades o la propia cuenta de la plataforma únicamente. Puede revisarse en el Smart Contract UniversityPlatform.sol

```
/*
 * @dev Modifier que se utiliza para comprobar que el _msgSender() es una cuenta registrada, universidad o alumno, en la
 * plataforma o la propia cuenta gestora.
 */
modifier onlyRegisteredUser() {
    require(
        (_universities[msg.sender] != address(0x0)) ||
        (_students[msg.sender] != address(0x0)) ||
        (msg.sender == owner())
    ), "RegisteredUser: caller is not a registred user";
    _;
}
```

7 La solución

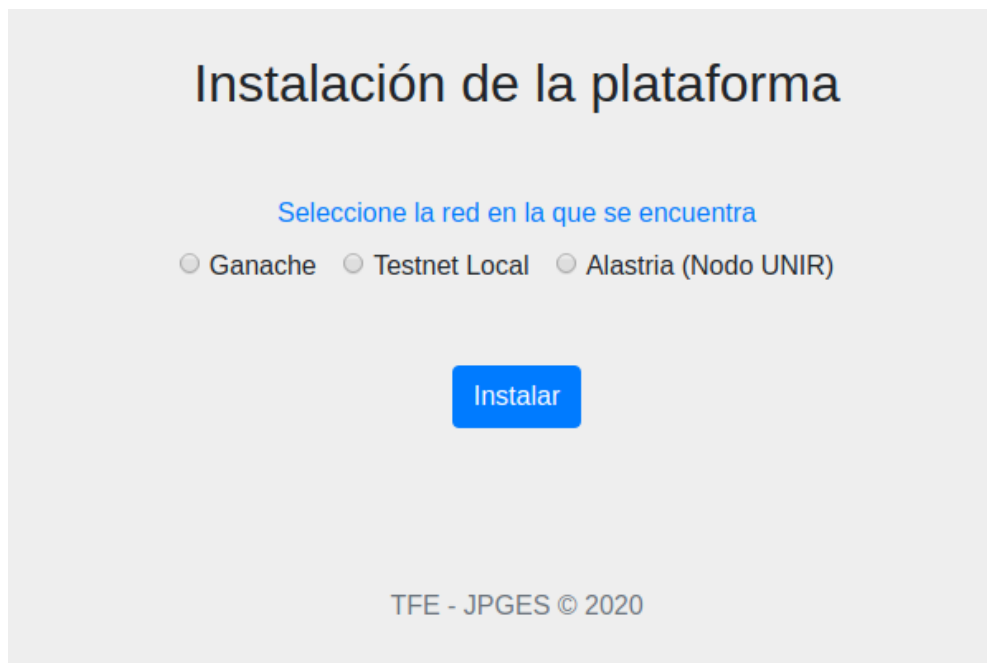
Básicamente en este apartado vamos a presentar todas las pantallas que se han desarrollado indicando la funcionalidad de cada una de ellas a modo de manual de usuario.

En el primer acceso a la aplicación, llegamos a la siguiente pantalla:



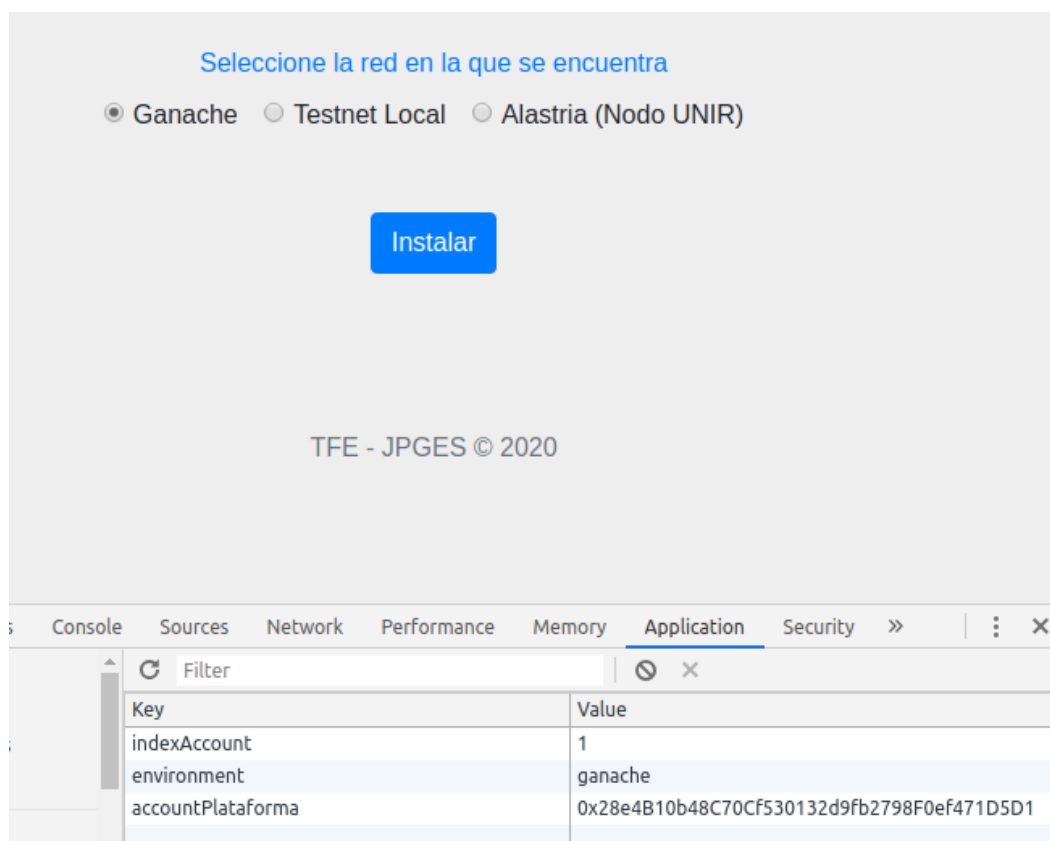
Nos presenta 3 accesos, Install, Universidades y Estudiantes, pero inicialmente únicamente Install se encuentra activo, pues se supone que únicamente el gestor de la plataforma puede acceder y se debe a que todavía no se ha configurado.

Pulsamos "Install"

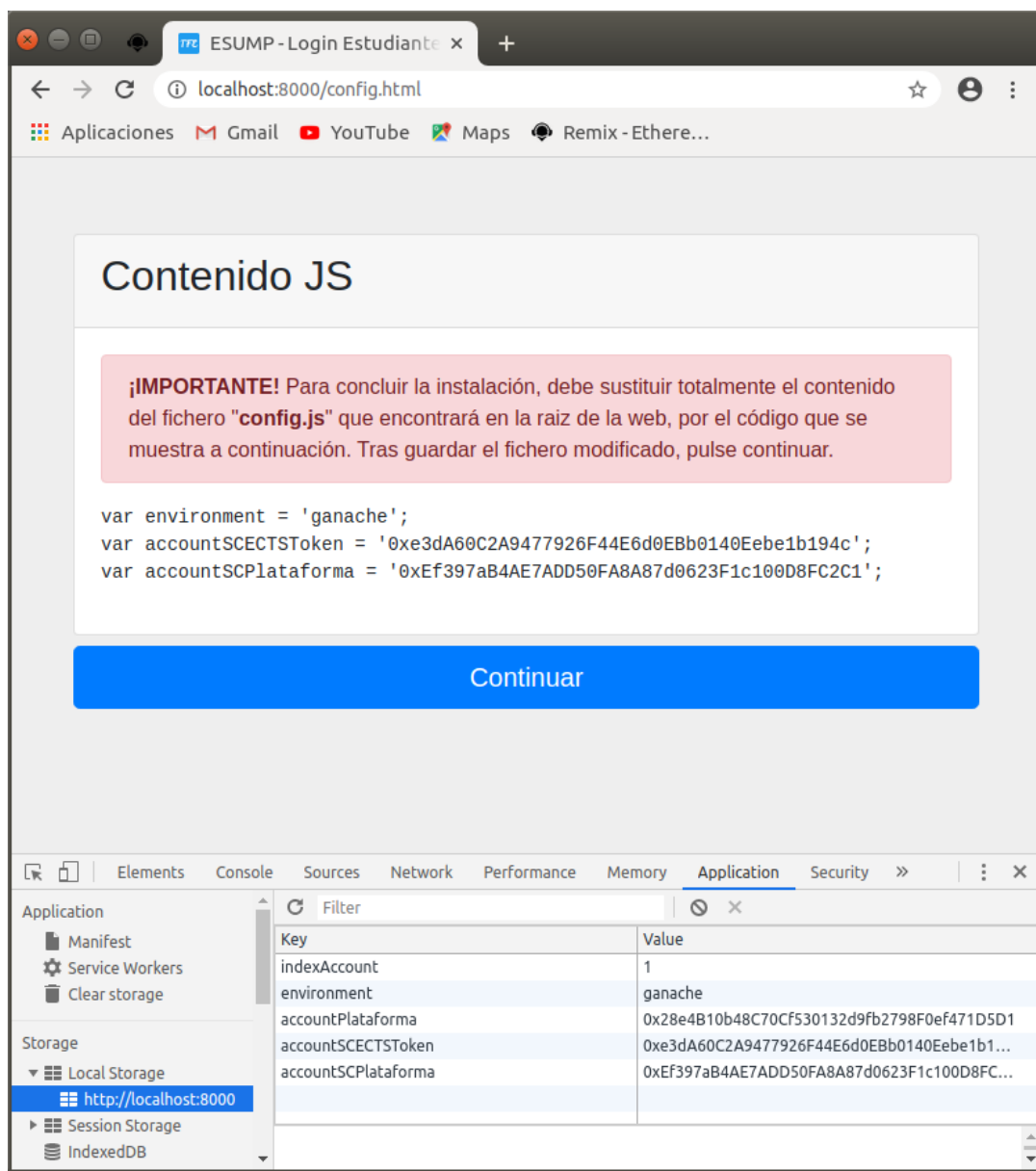


La pantalla nos indica que seleccionemos la red sobre la que estamos instalando y pulsemos "Instalar".

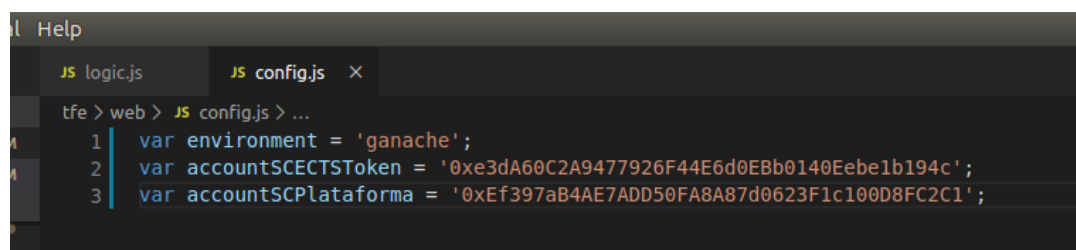
Simplemente con pulsar una de las redes podemos comprobar en la consola como ya estamos estableciendo valores:



Tras pulsar sobre instalar, llegamos a la siguiente pantalla:



Ojo, es muy importante que al copiar la información que aparece en la pantalla sobre el fichero, nos fijemos que se trata del fichero **config.js** que aparece dentro del directorio **/web**, no el que hay en el directorio raíz del proyecto.



Copiamos el contenido en el fichero /web/config.js y tras guardarlo pulsamos sobre "Continuar".

Llegaremos con ello de nuevo a la pantalla inicial, pero en este momento el botón "Install" ya se encontrará inhabilitado y por el contrario los botones "universidades" y "estudiantes" se mostrarán habilitados (podría ser necesario refrescar el navegador con Ctrl F5 para evitar el cacheo, es algo que no ha quedado bien resuelto en esta iteración).



Accedemos ahora pulsando sobre "Universidades"

The image shows the "Acceso Universidades" login form. It has a title "Acceso Universidades" in black. Below it are two input fields: "Cuenta" and "Password". There is a "Registrarse" button (disabled, light gray) and a large blue "Entrar" button. At the bottom, it says "TFE - JPGES © 2020".

Puesto que todavía no tenemos ninguna universidad registrada, pulsamos sobre "Registrarse"

Registro de universidades

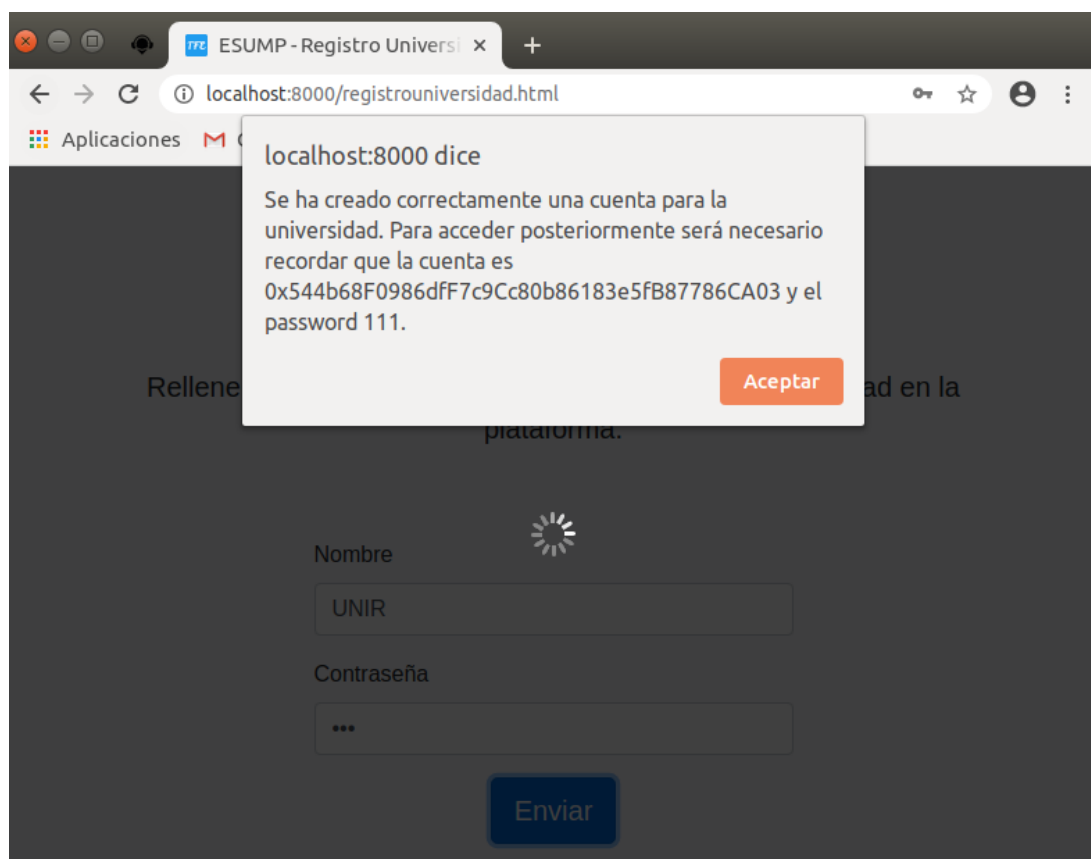
Rellene la siguiente información para registrar la universidad en la plataforma.

Nombre

Contraseña

[Enviar](#)

Tras registrarse indicando un nombre y una contraseña, el sistema nos indicará un número de cuenta y la contraseña utilizada que deberemos recordar para futuros accesos.



Accedemos ahora desde el login de universidades con los datos que hemos obtenido.

Acceso Universidades

0x544b68F0986dfF7c9Cc80b86183e5f

...

Registrarse

Entrar

TFE - JPGES © 2020

Y con ello llegaremos al panel de control de las universidades.

ESUMP - Panel Control Un x

localhost:8000/universidades.html

Aplicaciones Gmail YouTube Maps Remix - Ethere...

UNIR

Cerrar Sesión

99971.40
miliether

0 ECTS
reembolsables

Reembolsar ECTS

Lista de asignaturas

Publicar nueva asignatura

Asignatura	Símbolo	ECTS	Temario

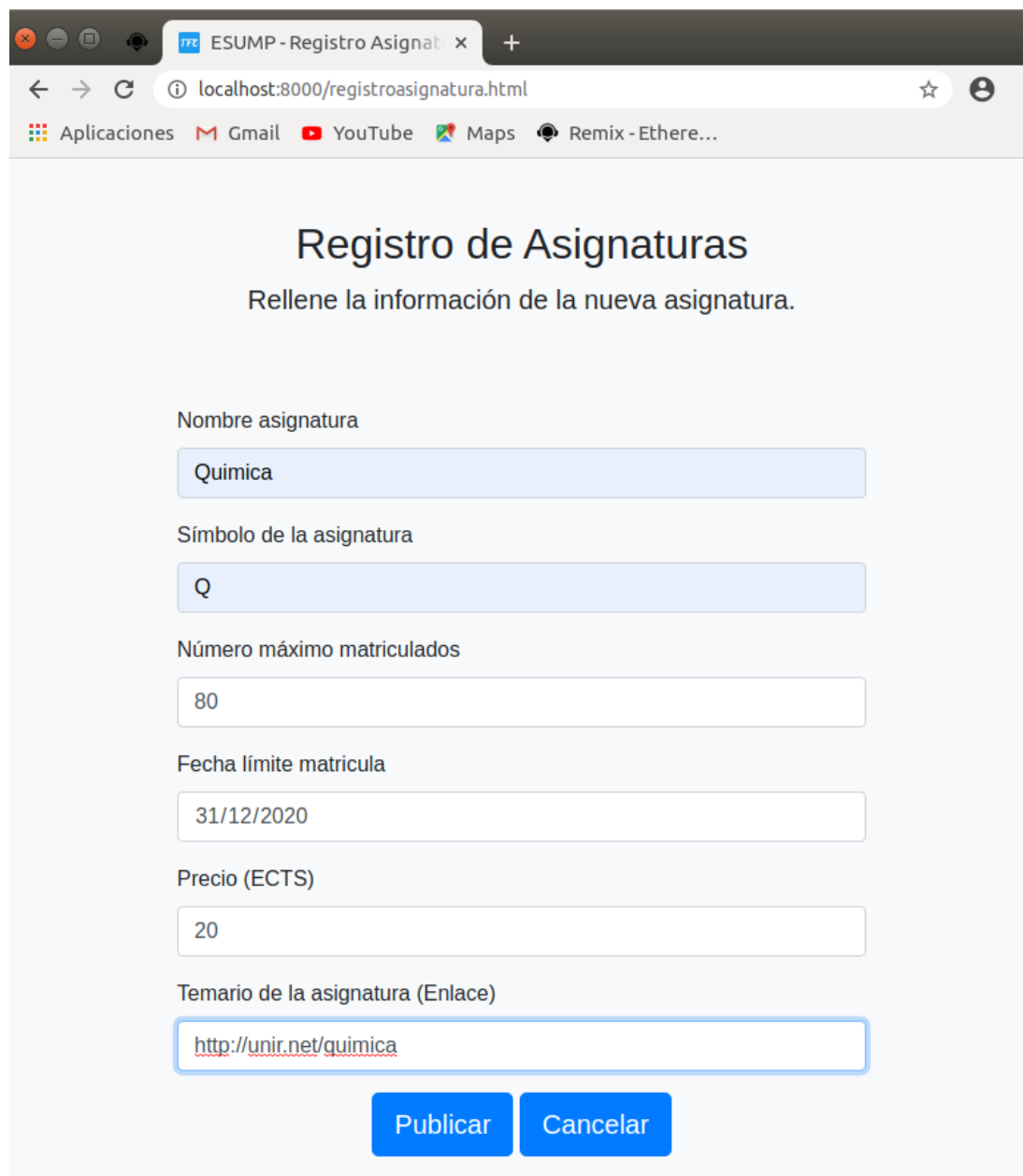
Lista de depósitos

Cuenta	Nombre	Saldo

La pantalla nos indica en este caso:

- Cantidad de ether que tenemos.
- Los ECTSToken reembolsable que tiene la universidad.
- La lista de asignaturas que tenemos publicadas.
- La lista de depósitos de alumnos que tenemos registrados.

Empezaremos publicando algunas asignaturas. Al pulsar sobre el botón "Publicar nueva asignatura" llegaremos a la pantalla:



The screenshot shows a web browser window with the title 'ESUMP - Registro Asignat'. The address bar shows 'localhost:8000/registroasignatura.html'. The browser's taskbar includes 'Aplicaciones', 'Gmail', 'YouTube', 'Maps', and 'Remix - Ethere...'. The main content area has the heading 'Registro de Asignaturas' and the instruction 'Rellene la información de la nueva asignatura.' Below this are several form fields: 'Nombre asignatura' with the value 'Quimica', 'Símbolo de la asignatura' with the value 'Q', 'Número máximo matriculados' with the value '80', 'Fecha límite matricula' with the value '31/12/2020', 'Precio (ECTS)' with the value '20', and 'Temario de la asignatura (Enlace)' with the value 'http://unir.net/quimica'. At the bottom are two blue buttons: 'Publicar' and 'Cancelar'.

Registro de Asignaturas

Rellene la información de la nueva asignatura.

Nombre asignatura

Quimica

Símbolo de la asignatura

Q

Número máximo matriculados

80

Fecha límite matricula

31/12/2020

Precio (ECTS)

20

Temario de la asignatura (Enlace)

<http://unir.net/quimica>

Publicar Cancelar

En esta pantalla registramos una nueva asignatura y la ofrecemos a los estudiantes registrados en el sistema. Tras concluir el proceso, podemos comprobar como la nueva asignatura ya aparece en la pantalla principal.

UNIR Cerrar Sesión

99913.52 miliether 0 ECTS reembolsables Reembolsar ECTS

Lista de asignaturas Publicar nueva asignatura

Asignatura	Símbolo	ECTS	Temario
Quimica	Q	20	http://unir.net/quimica

Alumnos

Lista de depósitos

Cuenta	Nombre	Saldo
--------	--------	-------

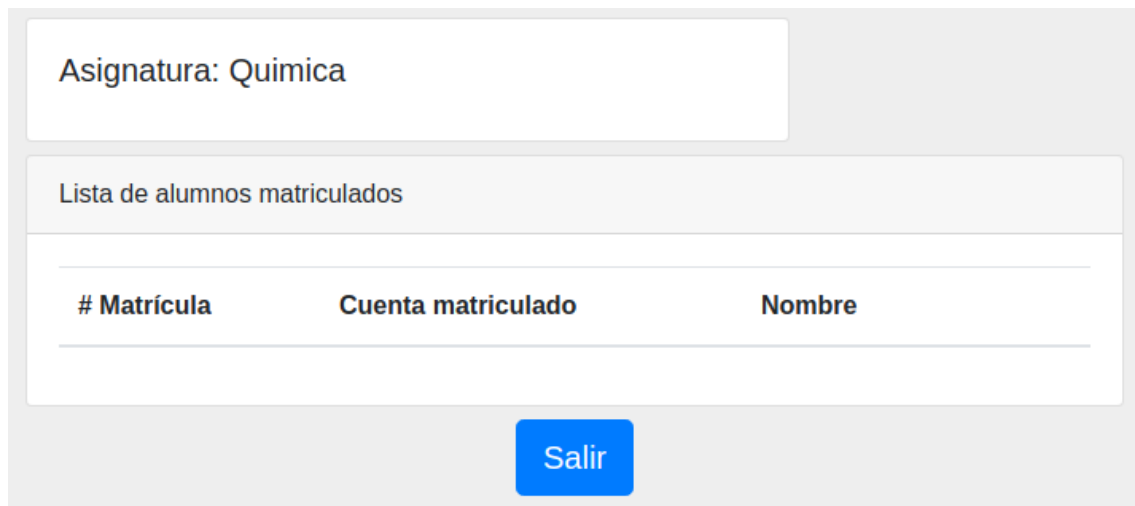
Repetiremos el proceso para algunas asignaturas más.

Lista de asignaturas Publicar nueva asignatura

Asignatura	Símbolo	ECTS	Temario
Quimica	Q	20	http://unir.net/quimica
Fisica	F	20	http://unir.net/fisica
Matematicas	M	20	http://unir.net/matematicas

Alumnos Alumnos Alumnos

Si intentamos comprobar la lista de alumnos de alguna de las asignaturas pulsando sobre el botón "Alumnos" de la misma, verificaremos que en este momento todavía no tenemos alumnos matriculados en las asignaturas.



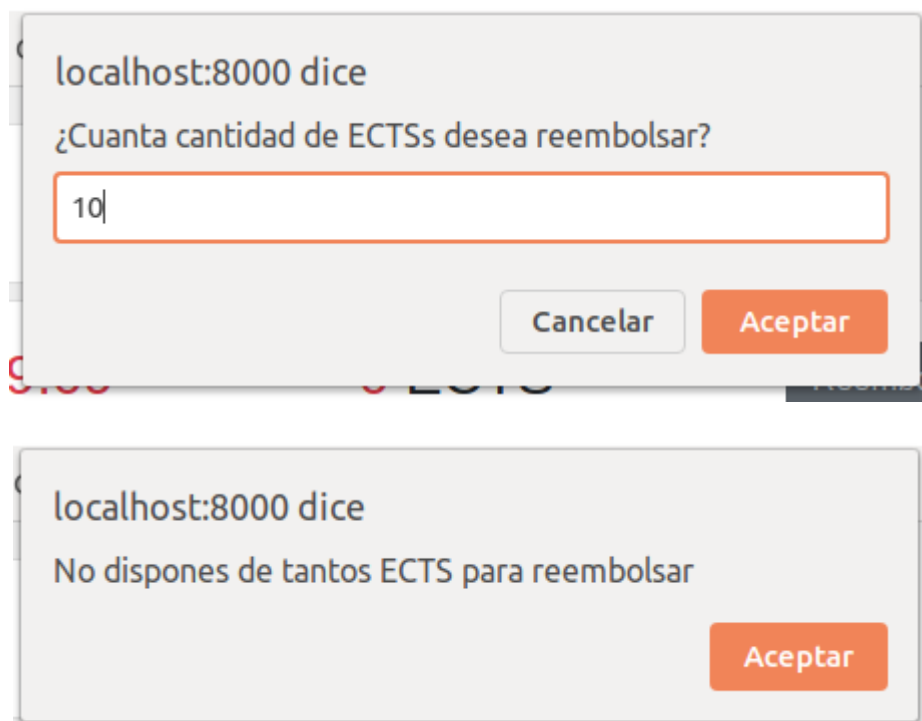
Asignatura: Quimica

Lista de alumnos matriculados

# Matrícula	Cuenta matriculado	Nombre
-------------	--------------------	--------

Salir

Igualmente, si pulsando el botón "Reembolsar ECTS", intentamos reembolsar ECTSToken cuando todavía vemos que no nos aparece ninguno como reembolsable en la pantalla principal, obtendremos diferentes errores, tanto desde front como desde back.



localhost:8000 dice
¿Cuanta cantidad de ECTSs desea reembolsar?

10

Cancelar Aceptar

localhost:8000 dice
No dispones de tantos ECTS para reembolsar

Aceptar

Nos salimos de la sesión pulsando "Cerrar sesión" y, desde la página de inicio nuevamente, nos dirigiremos a la pantalla de login de estudiantes.

Nuevamente la estructura es muy similar al caso anterior.

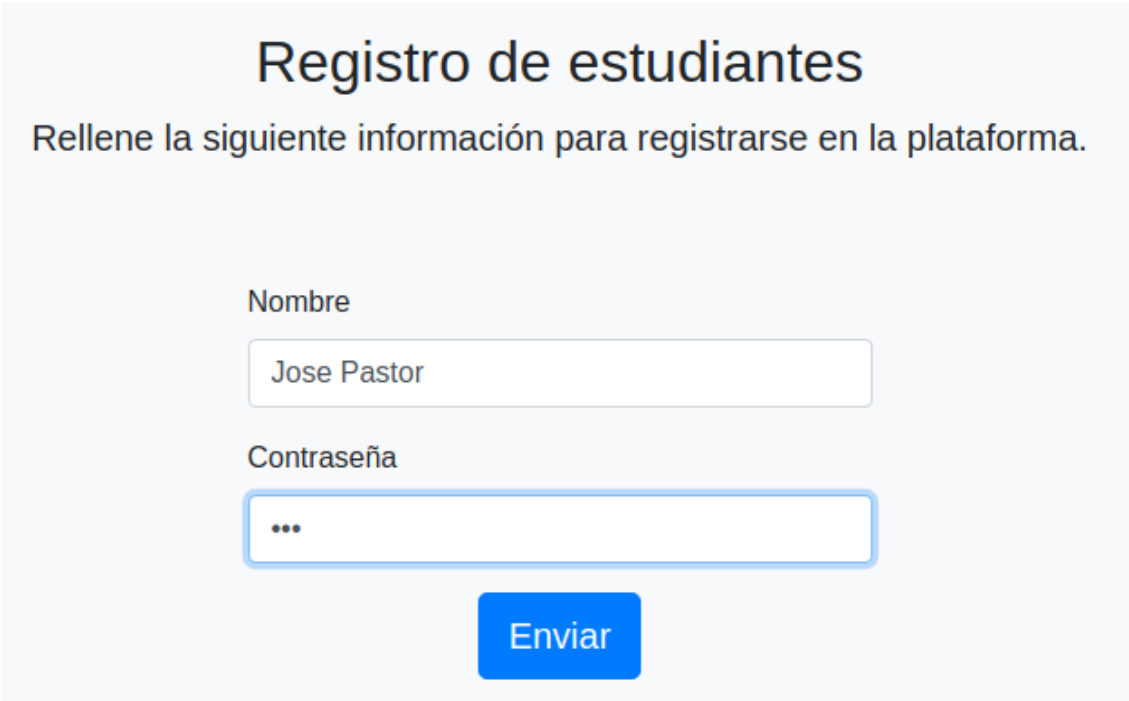


A login form titled "Acceso Estudiantes" on a light gray background. It features two input fields: "Cuenta" and "Password". Below these fields is a "Registrarse" button. A large blue "Entrar" button is positioned below the registration button. At the bottom, the text "TFE - JPGES © 2020" is displayed.

Acceso Estudiantes

TFE - JPGES © 2020

Lo primero nuevamente sería registrar un estudiante.



A registration form titled "Registro de estudiantes" on a light gray background. It includes the instruction "Rellene la siguiente información para registrarse en la plataforma." followed by two input fields: "Nombre" (containing "Jose Pastor") and "Contraseña" (containing three dots). A blue "Enviar" button is located at the bottom.

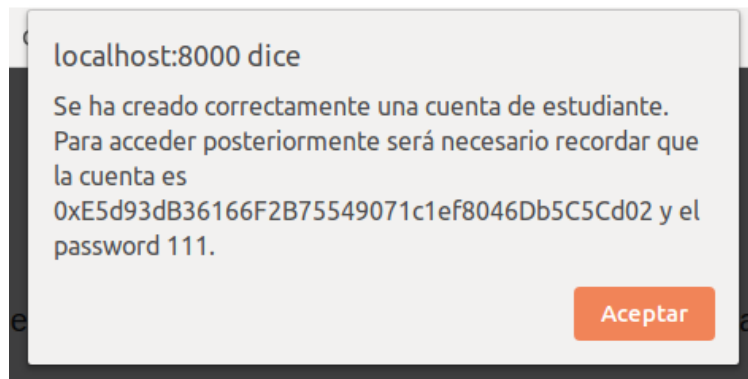
Registro de estudiantes

Rellene la siguiente información para registrarse en la plataforma.

Nombre

Contraseña

Tras rellenar la información de nombre y contraseña, pulsamos sobre Enviar y el sistema nos informará con una cuenta y la contraseña de acceso.



De nuevo iniciamos sesión, en esta ocasión como estudiante.

A login form titled "Acceso Estudiantes" in a large, bold, black font. Below the title are two input fields: the first contains the hexadecimal string "0xE5d93dB36166F2B75549071c1ef80", and the second is empty with a blue border and a cursor. Below the inputs is a gray button labeled "Registrarse". Further down is a large blue button labeled "Entrar". At the bottom, in a smaller font, is the text "TFE - JPGES © 2020".

Tras acceder, llegamos la pantalla principal del estudiante.

The screenshot shows a student dashboard with the following elements:

- User Profile:** A white box containing the name "Jose Pastor" and a red "Cerrar Sesión" button.
- ECTS Balance:** A section stating "Tienes 0 tokens ECTS" with a conversion rate "(1 ECTS = 760 miliether)".
- Courses List:** A section titled "Lista de matrículas en las que estás matriculado" containing a table with headers: "Asignatura", "Símbolo", "ECTSs", and "Temario".
- Deposits List:** A section titled "Lista de depósitos" with two blue buttons: "Transferir nuevo depósito" and "Comprar ECTS". Below the buttons is a table with headers: "Cuenta universidad", "Nombre", and "Deposito (ECTS)".

Esta pantalla nos ofrece mucha información:

- ECTSToken que el estudiante tiene en su poder.
- Listado de asignaturas en las que el estudiante está matriculado.
- Listado de depositos que el estudiante ha realizado.

El precio que aparece para el ECTS de 760 miliethers ha sido fijado en el front y en el back se ha establecido ese precio. Aunque en el back se han implementado las funciones necesarias para modificarlo, en el front no se ha implementado esta funcionalidad por falta de tiempo y se ha considerado este valor fijo.

Lo primero que vamos a hacer es "Comprar ECTS" pulsando sobre el botón.

Comprar ECTS

Tienes **99980.26** miliether (1 ECTS = 760 miliether)

¿Cuántos ECTSToken quieres comprar?

En la pantalla de compra de ECTS, simplemente debemos indicar la cantidad que queremos adquirir. Si nos pasamos en nuestra solicitud y no tuviesemos suficiente ether para ello, el sistema nos avisaría.



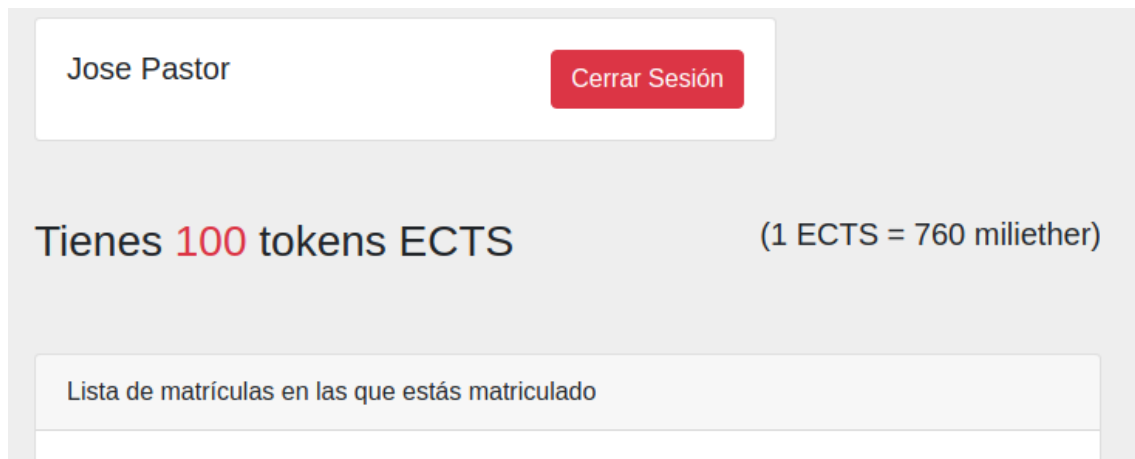
The screenshot shows a web browser window with a modal dialog box in the foreground. The dialog box has a title bar that says "localhost:8000 dice" and contains the text: "Dispones únicamente de 99980.26382 miliEther y para esa compra necesitas 152000." There is an "Aceptar" button in the bottom right corner of the dialog. Behind the dialog, the "Comprar ECTS" interface is visible, showing the user's balance as "99980.26" and a text input field containing the number "200".

Comprar ECTS

Tienes **99980.26** miliether (1 ECTS = 760 miliether)

¿Cuántos ECTSToken quieres comprar?

Cuando la cantidad sea suficiente, veremos que la cantidad queda adquirida y así es indicado en la página principal.



Jose Pastor Cerrar Sesión

Tienes **100** tokens ECTS (1 ECTS = 760 miliether)

Lista de matrículas en las que estás matriculado

Nuevamente desde la pantalla principal, intentaremos transferir un deposito a la universidad que hemos registrado con anterioridad. Pulsamos sobre "Transferir nuevo deposito"



Transferir deposito ECTS

Tienes **100** tokens ECTS (1 ECTS = 760 miliether)

Universidades UNIR ▼

¿Cuantos tokens ECTS quieres transferir?

Transferir Cancelar

En esta pantalla debemos indicar la cantidad de ECTS que queremos transferir y al deposito de qué universidad.

Universidades **UNIR** ▼

¿Cuántos tokens ECTS quieres transferir?

60

Transferir **Cancelar**

Tras ejecutarse la transferencia, el sistema volverá a la pantalla principal y podremos verificar como todo ha quedado registrado.

Jose Pastor **Cerrar Sesión**

Tienes **40** tokens ECTS (1 ECTS = 760 miliether)

Lista de matrículas en las que estás matriculado

Asignatura	Símbolo	ECTSs	Temario
------------	---------	-------	---------

Lista de depósitos **Transferir nuevo depósito** **Comprar ECTS**

Cuenta universidad	Nombre	Deposito (ECTS)
0x9ed9993dDBFD2467E2c204Bfc58FAF84F2eDC065	UNIR	60

Asignaturas

Podemos acceder ahora a la pantalla de asignaturas ofrecidas por la universidad pulsando sobre el botón “Asignaturas”.

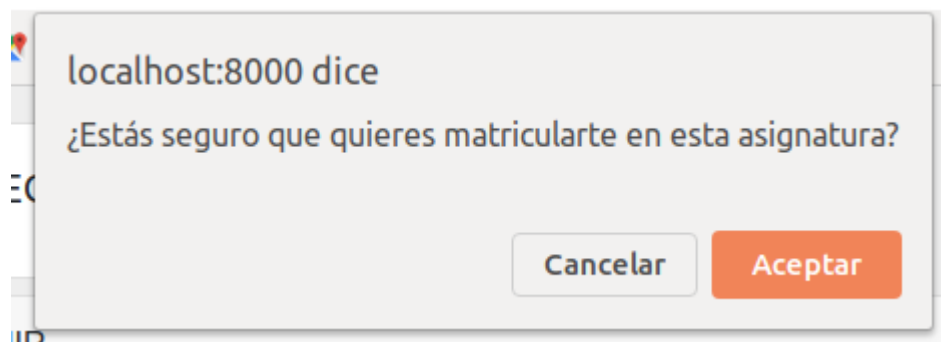
Deposito en UNIR: **60** ECTS.

Asignaturas ofrecidas por UNIR.

Cuenta despliegue	Asignatura	Símbolo	ECTS	Temario
Quimica	Q	20	http://unir.net/quimica	<button>Matricular</button>
Fisica	F	20	http://unir.net/fisica	<button>Matricular</button>
Matematicas	M	20	http://unir.net/matematicas	<button>Matricular</button>

Salir

Esta pantalla nos muestra el deposito ECTS que tenemos con esta universidad y las asignaturas que ofrece, y nos permite matricularnos en una asignatura pulsando el botón “Matricular”. Al hacerlo nos pedirá que confirmemos la acción.



Y tras ello nos llevará de nuevo a la pantalla principal donde ya podremos comprobar como aparecemos matriculados en la asignatura.

Tienes **40** tokens ECTS (1 ECTS = 760 miliether)

Lista de matrículas en las que estás matriculado

Asignatura	Símbolo	ECTSs	Temario
Quimica	Q	20	http://unir.net/quimica




Suspense o no evaluado

Repetimos la acción para las 3 asignaturas que habíamos publicado en la universidad y comprobaremos en la pantalla principal la matrícula en las 3 asignaturas.

Tienes **40** tokens ECTS

(1 ECTS = 760 miliether)

Lista de matrículas en las que estás matriculado

Asignatura	Símbolo	ECTSs	Temario	
Química	Q	20	http://unir.net/quimica	
Física	F	20	http://unir.net/fisica	
Matematicas	M	20	http://unir.net/matematicas	

Lista de depósitos

Transferir nuevo depósito

Comprar ECTS

Cuenta universidad	Nombre	Deposito (ECTS)	
0xad6d814268C02d45b631e44Bb69AAb3afb88fCCd	UNIR	0	<div>Asignaturas</div>

Vemos también en esta pantalla como al matricularnos en las 3 asignaturas, con un coste cada una de 20 ECTS, nos hemos quedado sin deposito en este universidad y de hecho comprobamos como en la tabla inferior nos aparece el valor de deposito igual a 0.

Salimos de la sesión del estudiante pulsando el botón "Cerrar sesión".

Nos logamos de nuevo como universidad.

UNIR

Cerrar Sesión

99799.09 miliether

60 ECTS
reembolsables

Reembolsar ECTS

Lista de asignaturas


Publicar nueva asignatura

Asignatura	Símbolo	ECTS	Temario	
Quimica	Q	20	http://unir.net/quimica	Alumnos
Fisica	F	20	http://unir.net/fisica	Alumnos
Matematicas	M	20	http://unir.net/matematicas	Alumnos

Nos dirigimos al listado de asignaturas y accedemos a la lista de estudiantes matriculados pulsando "Alumnos"

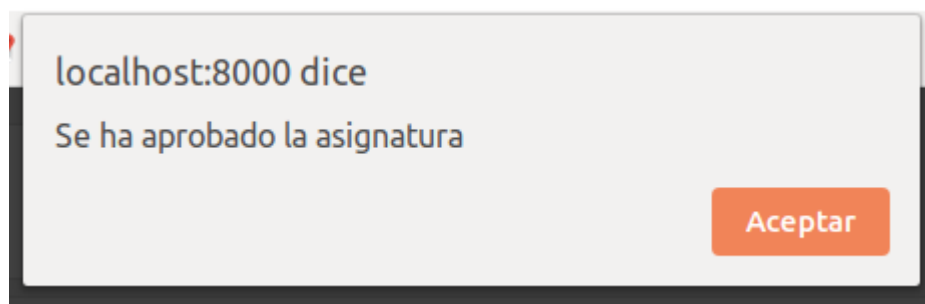
Asignatura: Quimica

Lista de alumnos matriculados

# Matrícula	Cuenta matriculado	Nombre	
1	0xB421c44059EA3cF85396b0f3276E7057D1af7Db5	Jose Pastor	 <div>Aprobar</div>

Salir


Podemos comprobar por ejemplo como en la asignatura "Química" nos aparece en este caso matriculado el alumno de ejemplo que se ha matriculado. Si pulsamos sobre el botón "Aprobar", el sistema nos pedirá que confirmemos la acción.



Veremos ahora que el icono que aparece en la lista indicando el estado de la matrícula ha cambiado y ahora se muestra como aprobado.

Asignatura: Química

Lista de alumnos matriculados

# Matrícula	Cuenta matriculado	Nombre	
1	0xB421c44059EA3cF85396b0f3276E7057D1af7Db5	Jose Pastor	 Aprobar




Salir

Igualmente, si accedemos de nuevo con la sesión del estudiante podremos comprobar como a él también le aparece como aprobada.

Jose Pastor
Cerrar Sesión

Tienes **40** tokens ECTS
(1 ECTS = 760 miliether)

Lista de matrículas en las que estás matriculado

Asignatura	Símbolo	ECTSs	Temario	
Quimica	Q	20	http://unir.net/quimica	
Fisica	F	20	http://unir.net/fisica	
Matematicas	M	20	http://unir.net/matematicas	

Por último y para cerrar el ciclo económico, volvemos a la pantalla de la universidad y podremos comprobar como ahora sí que aparecen ECTS reembolsables, los que corresponden a las 3 matrículas en las que se ha suscrito nuestro estudiante.

UNIR
Cerrar Sesión

99799.09 miliether
60 ECTS reembolsables

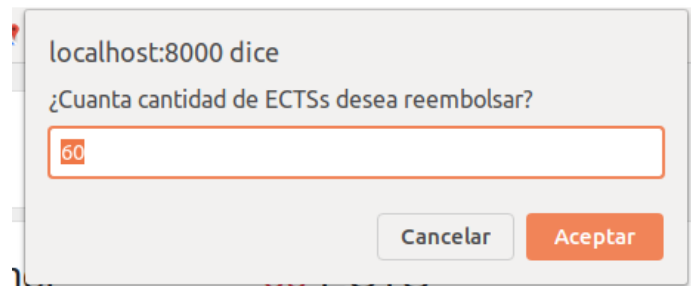
Reembolsar ECTS

Lista de asignaturas

Publicar nueva asignatura

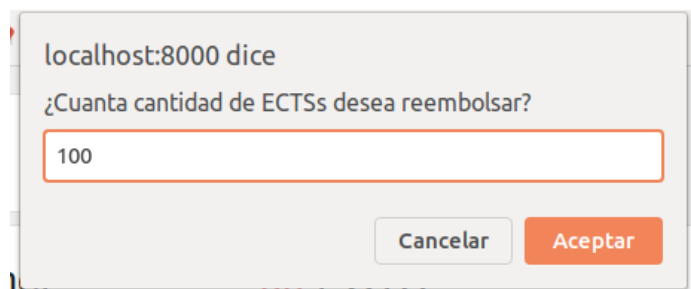
Asignatura	Símbolo	ECTS	Temario	
Quimica	Q	20	http://unir.net/quimica	<div>Alumnos</div>
Fisica	F	20	http://unir.net/fisica	<div>Alumnos</div>
Matematicas	M	20	http://unir.net/matematicas	<div>Alumnos</div>

Si ahora pulsamos sobre el botón "Reembolsar ECTS", el sistema nos sugerirá reembolsar la cantidad total disponible, aunque nos permitirá cambiarla.



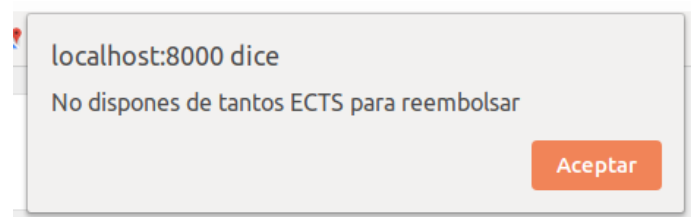
A modal dialog box with a light gray background. At the top, it says "localhost:8000 dice". Below that is the question "¿Cuanta cantidad de ECTSs desea reembolsar?". There is a text input field containing the number "60". At the bottom right, there are two buttons: "Cancelar" (disabled) and "Aceptar" (active).

Obviamente, si no queremos la totalidad deberíamos indicar una cifra menor, aunque si intentamos algo mayor



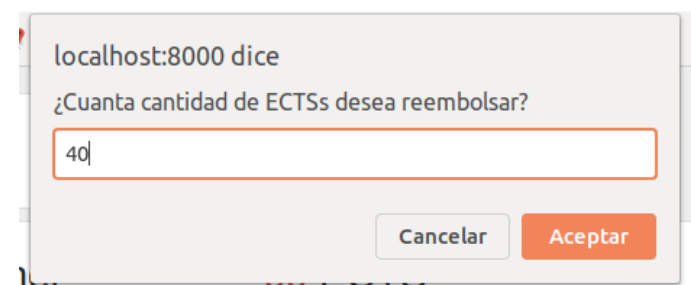
A modal dialog box with a light gray background. At the top, it says "localhost:8000 dice". Below that is the question "¿Cuanta cantidad de ECTSs desea reembolsar?". There is a text input field containing the number "100". At the bottom right, there are two buttons: "Cancelar" (disabled) and "Aceptar" (active).

El sistema nos advertirá del error

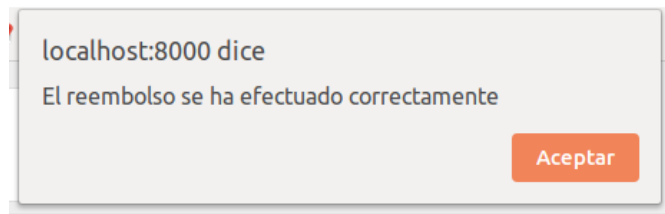


A modal dialog box with a light gray background. At the top, it says "localhost:8000 dice". Below that is the error message "No dispones de tantos ECTS para reembolsar". At the bottom right, there is a single button labeled "Aceptar".

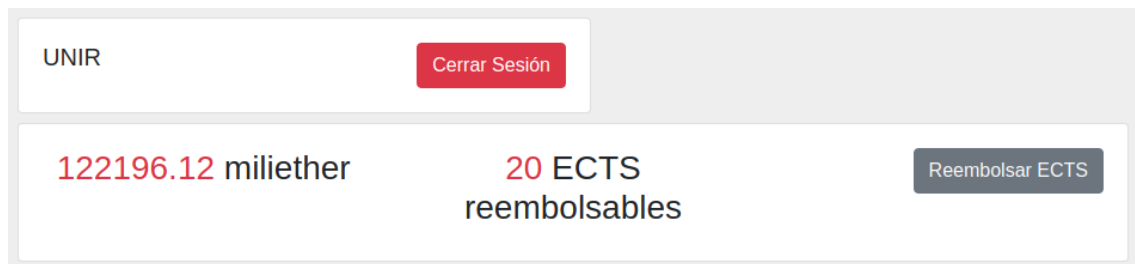
Finalmente, si la cantidad es aceptable, el sistema reembolsará los ECTS por ether.



A modal dialog box with a light gray background. At the top, it says "localhost:8000 dice". Below that is the question "¿Cuanta cantidad de ECTSs desea reembolsar?". There is a text input field containing the number "40". At the bottom right, there are two buttons: "Cancelar" (disabled) and "Aceptar" (active).



Podemos comprobar en la pantalla principal de la universidad nuevamente, como la cantidad total de ether se ha incrementado y por el contrario la cantidad de ECTS reembolsables ha disminuido.



Tal y como hemos indicado en el análisis, el sistema se sustentará a partir de la diferencia entre el precio de coste del ECTSToken para el estudiante y el precio de reembolso del ECTSToken para la universidad.

8 Despliegue e implantación

En este asunto lo más sencillo es remitirse a la información que se ha incorporado al fichero https://github.com/jpges/UNIR_TFE/blob/master/README.md, no obstante, voy a hacer un pequeño resumen.

Para el despliegue de la solución debemos seguir los siguientes pasos:

1. Instalamos **node.js** y **npm** sobre nuestro sistema (ver README.md).
2. Clonar el repositorio del proyecto con
\$git clone https://github.com/jpges/UNIR_TFE.git
O descargarlo de https://github.com/jpges/UNIR_TFE/archive/master.zip
3. En los casos de "ganache" o "testnet" iniciar la red Blockchain sobre la que se quiere conectar. En el caso de Alastria obviamente no es necesario iniciar nada.
4. Instala el proyecto lanzando desde el directorio raíz el comando:
\$npm install
5. Inicia el proyecto lanzando desde el directorio raíz el comando:
\$npm start
A partir de este momento, si has seguido estos pasos, podrás acceder a la aplicación en localhost:8000

Si el despliegue se realiza en un entorno de desarrollo porque se quieren realizar modificaciones, compilar y/o ejecutar planes de prueba, será necesario también:

6. Instalar truffle (ver README.md).
7. Conectar Remix con la carpeta local ./contracts del proyecto utilizando el comando:
taskset -c 0 remixd -s [/PATH/TO/PROJECT]/contracts --remix-ide <http://remix.ethereum.org>

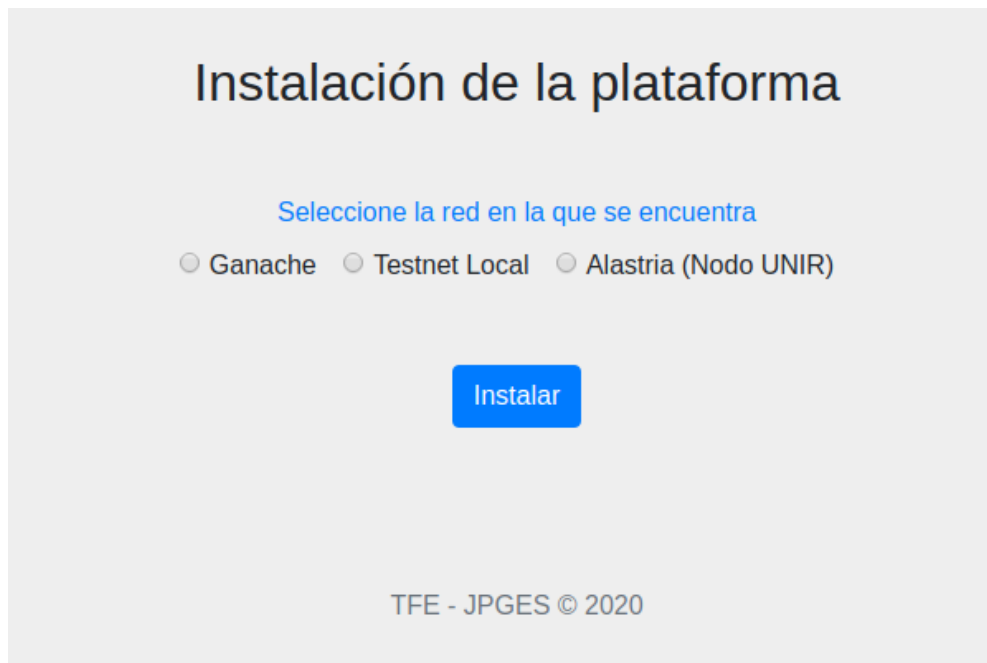
Donde sustituiremos [/PATH/TO/PROJECT] por la ruta absoluta de nuestro proyecto.

Si no tenemos el remixd instalado, lo haremos con el comando:

sudo npm install -g remixd

Esta solución se ofrece con la configuración necesaria para tres entornos distintos: Ganache, una testnet local y la red Arrakis de Alastria a través del nodo de la UNIR.

Como vimos en el capítulo anterior, la solución ya nos ofrece una pantalla donde nos permite configurar la aplicación para conectar con uno de estos entornos.



Para entornos de desarrollo se ha incluido en el proyecto el fichero truffle-config.js, que incluye las configuraciones para las tres redes, llamándolas ganache, testnet y alastria.

```
networks: {
  ganache: {
    host: "127.0.0.1",
    port: 7545,
    network_id: "*",
  },

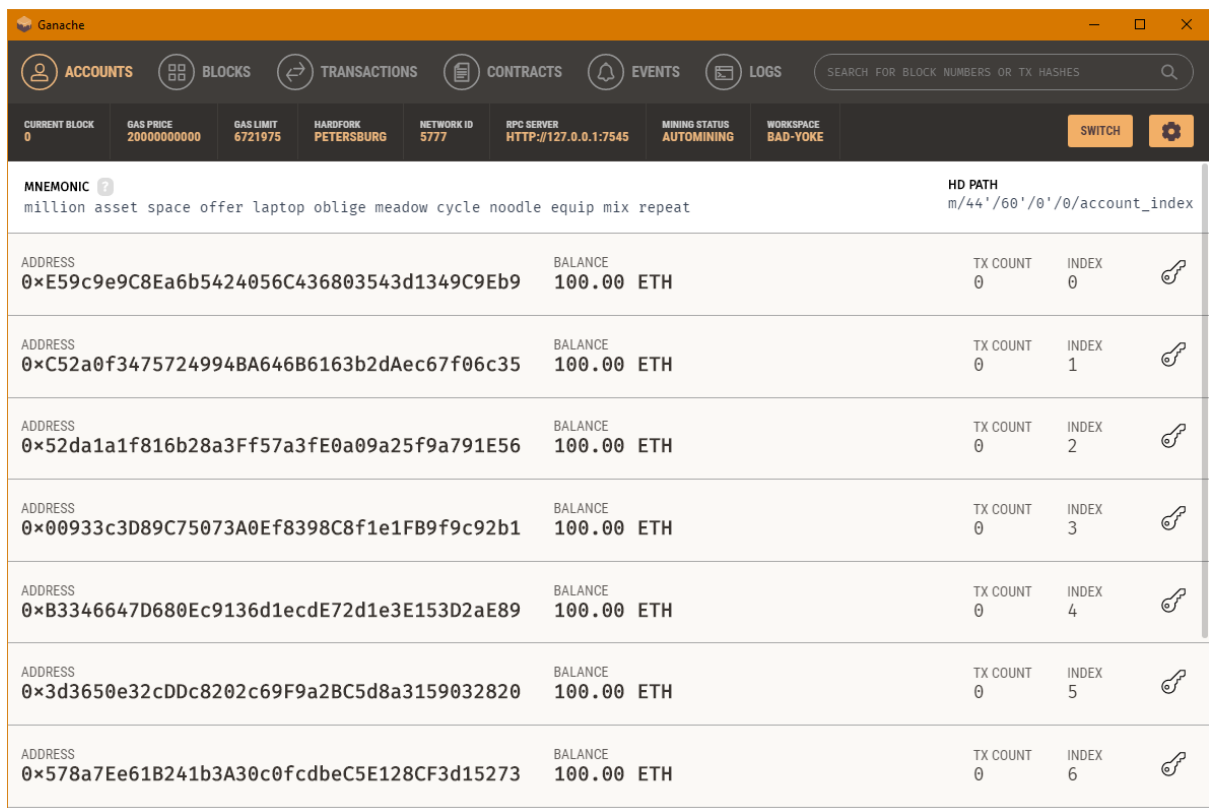
  testnet: {
    host: "127.0.0.1",
    port: 22001,
    network_id: "*",
    gas: 0xffffffff,
    gasPrice: 0x0,
    from: "0x74d4c56d8dbc10a567341bfac6da0a8f04dc41d"
  },

  alastria: {
    host: "10.141.8.11",
    port: "8545",
    network_id: "*",
    gas: 0xffffffff,
    gasPrice: 0x0,
    from: "0xabc869c21d631e122d35789942a573241ec04d2e4"
  }
}
```

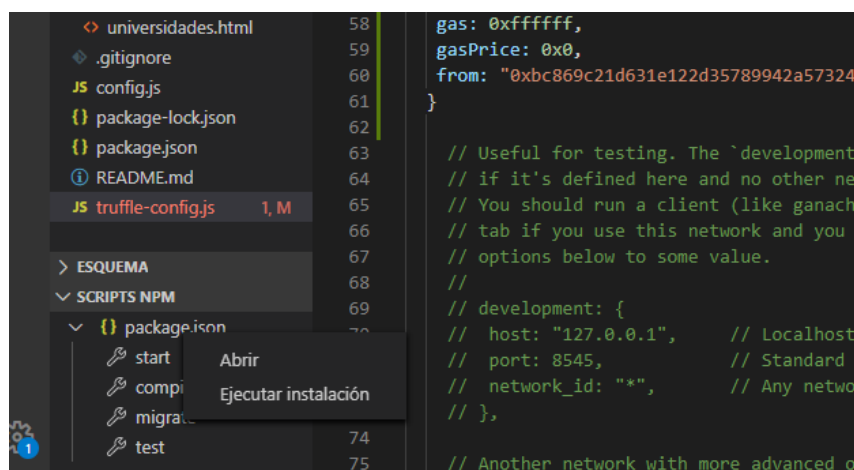
Ganache

Todo el desarrollo se ha realizado sobre este entorno. Todas las imágenes que se han mostrado en el presente documento han sido tomadas de la ejecución sobre este entorno.

Hemos utilizado la aplicación Desktop de Ganache:



Instalamos el proyecto, como decíamos con “npm install” o si estamos en un Visual Studio Code, pulsando botón derecho sobre “package.json” y seleccionando “Ejecutar instalación”.



Posteriormente podemos pulsar sobre "start" o ejecutar el comando "npm start", con esto nos arranca el servidor en <http://localhost:8000> y ya podemos instalar la aplicación desde el propio servidor siguiendo sus indicaciones.

Testnet y Alastría

Los detallaré en el apartado de pruebas.

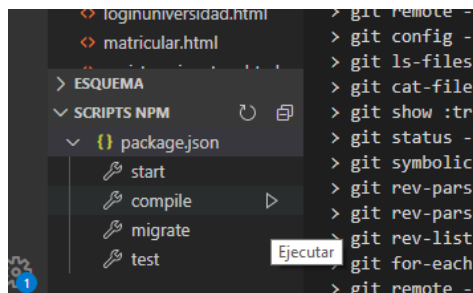
9 Testing

Ganache

Para ejecutar el plan de pruebas que se ha incluido en “./test” contra Ganache, es tan sencillo como recurrir a los scripts package.json que se han facilitado.

En el entorno de Visual Code, simplemente pulsando el “play” será suficiente para hacerlos funcionar.

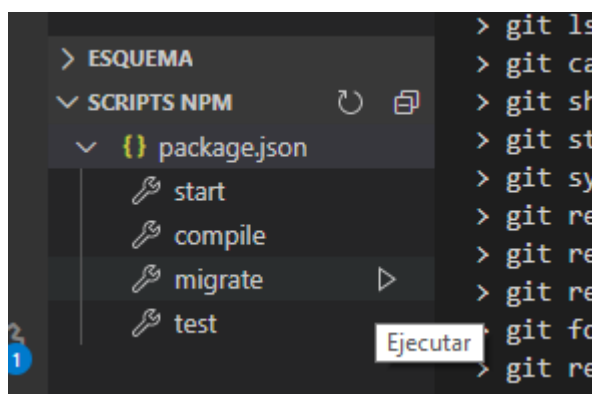
Para **compilar**:



O bien por línea de comandos con “npm compile” o por el mecanismo tradicional lanzando “truffle compile --all” siempre desde la raíz del proyecto.

```
13 | "migrate": "truffle migrate --network ganache --reset",
    |
    | OUTPUT  DEBUG CONSOLE  PROBLEMS  TERMINAL
    |-----|
    |
    | alastria@alastria-VirtualBox:~/Escritorio/UNIR_TFE$ truffle compile --all
    |
    | Compiling your contracts...
    | =====
    | > Compiling ./contracts/ECTSToken.sol
    | > Compiling ./contracts/Expirable.sol
    | > Compiling ./contracts/Limitable.sol
    | > Compiling ./contracts/Migrations.sol
    | > Compiling ./contracts/Student.sol
    | > Compiling ./contracts/SubjectToken.sol
    | > Compiling ./contracts/University.sol
    | > Compiling ./contracts/UniversityPlatform.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/drafts/Counters.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/introspection/ERC165.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/math/SafeMath.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/ownership/Ownable.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC20/ERC20Detailed.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC721/ERC721.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC721/ERC721Metadata.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC721/IERC721.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC721/IERC721Metadata.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
    | > Compiling ./node_modules/@openzeppelin/contracts/utils/Address.sol
    | > Artifacts written to /home/alastria/Escritorio/UNIR_TFE/build/contracts
    | > Compiled successfully using:
    |   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
    |
    | alastria@alastria-VirtualBox:~/Escritorio/UNIR_TFE$
```

Para realizar la **migración**:



O bien por línea de comandos con "npm migrate" o por el mecanismo tradicional lanzando "truffle migrate --network ganache --reset" siempre desde la raíz del proyecto.

El resultado de la migración en este entorno nos dice:

```
Starting migrations...
=====
> Network name:      'ganache'
> Network id:       5777
> Block gas limit:  0x6691b7

1_initial_migration.js
=====

Deploying 'Migrations'
-----
> transaction hash:  0xa674fbf8e0b0c75fc6f36aac66f06c1fec835a707167ec4a6fa62db9e6147ad1
> Blocks: 0         Seconds: 0
> contract address: 0xD62412B60eBcf1EbB5a8a9C72254447744601e8A
> block number:     1
> block timestamp:  1584364264
> account:          0x32567936F6E7cF7325856504c800F9BFAB9c0cc6
> balance:          99.99713516
> gas used:         143242
> gas price:        20 gwei
> value sent:       0 ETH
> total cost:       0.00286484 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:       0.00286484 ETH

2_migration_TFE.js
=====

Deploying 'SubjectToken'
-----
> transaction hash:  0x67c2dc6b54121dbcdb5215c429bc9a95aef4f01ca6de2e27ad7261b817bf5795
> Blocks: 0         Seconds: 0
> contract address: 0x19dC8A6011b48BC494bF7dd458B2007b1d5176ae
> block number:     3
> block timestamp:  1584364266
> account:          0x32567936F6E7cF7325856504c800F9BFAB9c0cc6
> balance:          99.94113398
> gas used:         2758133
> gas price:        20 gwei
> value sent:       0 ETH
> total cost:       0.05516266 ETH

Deploying 'Student'
-----
```

```
> transaction hash: 0x885efb1f218f2c75eaf425f658e87cbfe21fc0dc41fda1f34c7d35d057c146fa
> Blocks: 0 Seconds: 0
> contract address: 0xCB8E6ECaEf27e1e85cd018a7d91Cf2B18e73C2aa
> block number: 4
> block timestamp: 1584364267
> account: 0x32567936F6E7cF7325856504c800F9BFAB9c0cc6
> balance: 99.92095754
> gas used: 1008822
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.02017644 ETH
```

Deploying 'ECTSToken'

```
-----
> transaction hash: 0xa829ec0dfcc492efb0522141bed2b22ae9ef66590fc7a62b966ad21e921864d8
> Blocks: 0 Seconds: 0
> contract address: 0xE7AcAdA8e50bCF4B84b15E7F6CB66ccb78F9703B
> block number: 5
> block timestamp: 1584364267
> account: 0x32567936F6E7cF7325856504c800F9BFAB9c0cc6
> balance: 99.90051296
> gas used: 1022229
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.02044458 ETH
```

Deploying 'UniversityPlatform'

```
-----
> transaction hash: 0xf582a8383ccde4d78977bdfec7c308f43dea55ad855e5e46d5b6ba8ace5db9558
> Blocks: 0 Seconds: 0
> contract address: 0x6DBAe09Ed234778428e3e6D715d610278B4280a7
> block number: 6
> block timestamp: 1584364267
> account: 0x32567936F6E7cF7325856504c800F9BFAB9c0cc6
> balance: 99.872512
> gas used: 1400048
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.02800096 ETH
```

Deploying 'University'

```
-----
> transaction hash: 0x9ec696c1e3a613b246fcfa09f5de8d3432896e21359da90a5e936841f360074a
> Blocks: 0 Seconds: 0
> contract address: 0x0EE2535033bb6c002861131f9C70db0e47CC52A4
> block number: 7
> block timestamp: 1584364268
> account: 0x32567936F6E7cF7325856504c800F9BFAB9c0cc6
> balance: 99.8435478
> gas used: 1448210
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.0289642 ETH
```

```
> Saving migration to chain.
> Saving artifacts
```

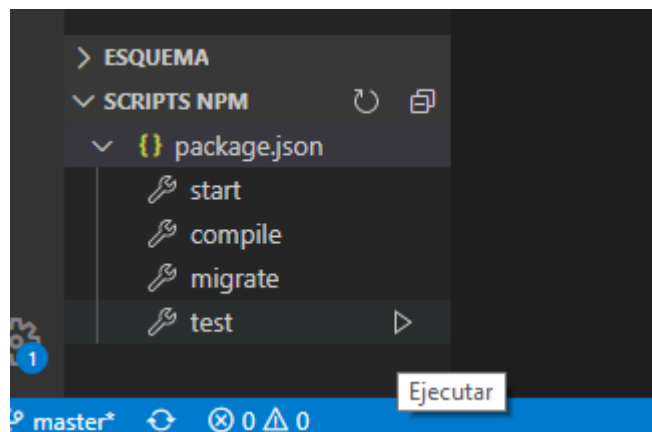
```
-----
> Total cost: 0.15274884 ETH
```

Summary

=====

```
> Total deployments: 6
> Final cost: 0.15561368 ETH
```

Finalmente, para ejecutar el **plan de pruebas**:



O bien por línea de comandos con "npm test" o por el mecanismo tradicional lanzando "truffle test --network ganache" siempre desde la raíz del proyecto.

El resultado de la ejecución ha sido:

```
> tfe@1.0.0 test /home/alastria/Escritorio/UNIR_TFE
> truffle test --network ganache

Using network 'ganache'.

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Contract: UniversityPlatform
  ✓ Test1: Despliegue ECTSToken correcto y sin totalSupply inicial (93ms)
  ✓ Test2: Despliegue de la plataforma correcto (222ms)
  ✓ Test3: La plataforma puede leer la lista de universidades en el sistema vacia al principio (91ms)
  ✓ Test4: Solo puede registrar una universidad desde la cuenta de la plataforma (151ms)
Ya tenemos 1 universidad registrada en la plataforma
  ✓ Test5: Se registra una universidad correctamente desde la cuenta de la plataforma (58ms)
  ✓ Test6: Un usuario no registrado no puede leer la lista de universidades (52ms)
  ✓ Test7: Una universidad registrada puede leer la lista de universidades (91ms)
  ✓ Test8: Lista de estudiantes en el sistema vacia al principio (50ms)
  ✓ Test9: Solo puede registrar estudiantes la cuenta de la plataforma (93ms)
Ya tenemos 1 estudiante registrado en la plataforma
  ✓ Test10: Se registra un estudiante desde la cuenta de la plataforma (107ms)
  ✓ Test11: Un estudiante registrado puede leer la lista de universidades (87ms)
  ✓ Test12: Una universidad registrada puede leer la lista de estudiantes (51ms)
  ✓ Test13: Conseguir el SC de una cuenta de universidad desde una universidad (51ms)
  ✓ Test14: Un usuario no registrado no puede conseguir el SC de una universidad
  ✓ Test15: Conseguir el SC de una cuenta de universidad desde una estudiante
  ✓ Test16: Conseguir el SC de una cuenta de estudiante desde un estudiante (38ms)
  ✓ Test17: Conseguir el SC de una cuenta de estudiante desde una universidad (41ms)
  ✓ Test18: Se rechaza conseguir el SC de una cuenta de estudiante desde una cuenta no registrada (46ms)
El nuevo precio de venta al estudiante de los ECTS queda en: 1000000000000000000
  ✓ Test19: Sube el precio de venta de los ECTS (84ms)
El nuevo precio de recompra a las universidades de los ECTS queda en: 500000000000000000
```

```
✓ Test20: Sube el precio de recompra de los ECTS (90ms)
✓ Test21: Solo la plataforma puede subir el precio de venta de los ECTS (129ms)
✓ Test22: Solo la plataforma puede subir el precio de recompra de los ECTS (217ms)
✓ Test23: Un estudiante no puede comprar con más Eter del que tiene (92ms)
El estudiante ahora tiene en ECTS: 5
✓ Test24: Un estudiante compra con su Ether los ECTS (317ms)
✓ Test25: Solo la universidad puede publicar sus asignaturas (143ms)
Ya tenemos 1 asignatura publicada
✓ Test26: La universidad publica la asignatura (113ms)
✓ Test27: El estudiante genera un deposito de 5 ECTS en la universidad (486ms)
✓ Test28: El estudiante verifica que la universidad se ha añadido a las universidades con deposito
✓ Test29: Se comprueba que la lista de universidades donde se tiene deposito solo la puede leer el
estudiante (66ms)
✓ Test30: Se comprueba que la universidad tiene depositados los 5 ECTS del Test27
✓ Test31: Matricularse en la asignatura publicada por la universidad (416ms)
✓ Test32: El estudiante verifica que está matriculado en la asignatura (74ms)
✓ Test33: El estudiante obtiene el tokenID de su matrícula (39ms)
✓ Test34: La universidad recupera la lista de matriculados en una asignatura (94ms)
El estudiante ha aprobado la asignatura
✓ Test35: La universidad aprueba la asignatura al estudiante (94ms)
✓ Test36: El estudiante comprueba que tiene la asignatura aprobada (126ms)
✓ Test37: La universidad comprueba que la asignatura aprobada (50ms)
✓ Test38: La universidad reclama la compensación en ether de sus ECTS (423ms)

38 passing (6s)
```

El plan de pruebas que se ha elaborado contempla un ciclo económico completo, desde el registro en la plataforma, hasta el reembolso de los ECTSToken por parte de la universidad.

Testnet

En este caso deberemos instalar la testnet localmente de acuerdo con las enseñanzas de la asignatura "Redes Blockchain" de este mismo curso.

Tras hacerlo deberíamos obrar igual que en el caso de Ganache, pero nos han surgido inconvenientes que no hemos podido solucionar.

1000 problemas para que la aplicación se autoinstale contra esta red. Finalmente pude comprobar que existen muchas discrepancias de versión que impiden el buen funcionamiento. Entre otros problemas, la activación en el servidor de CORS modificando el GLOBAL_ARGS del fichero start_node.sh:

```
GLOBAL_ARGS="--networkid $NETID --identity $IDENTITY --rpc --rpcaddr 0.0.0.0 --rpcapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,istanbul --rpcport 2200$PUERTO --port 2100$PUERTO --targetgaslimit 18446744073709551615 --ethstats $IDENTITY:bb98a0b6442386d0cdf8a31b267892c1@$ETH_STATS_IP:3000 --rpccorsdomain http://localhost:8000 "
```

La red se levanta correctamente y las cuentas se desbloquean:

```
alastria@alastria-VirtualBox:~/test-environment/infrastructure/testnet$ sudo ./bin/stop_network.sh

alastria@alastria-VirtualBox:~/test-environment/infrastructure/testnet$ sudo geth attach ipc:network/general1/geth.ipc
Welcome to the Geth JavaScript console!

alastria@alastria-VirtualBox:~/test-environment/infrastructure/testnet$ sudo ./bin/start_network.sh
clean 1 2

alastria@alastria-VirtualBox:~/test-environment/infrastructure/testnet$ ps -a
  PID TTY          TIME CMD
  6866 pts/17    00:00:00 http-server
 17553 pts/4      00:00:00 geth
 17564 pts/4      00:00:10 constellation-n
 17753 pts/4      00:00:00 geth
 17764 pts/4      00:00:10 constellation-n
 17954 pts/4      00:00:00 geth
 17963 pts/4      00:00:00 ps

sudo geth attach ipc:network/general1/geth.ipc
Welcome to the Geth JavaScript console!

instance: Geth/general1/v1.7.2-stable-94e1e31e/linux-amd64/go1.9.5
coinbase: 0x74d4c56d8dcbc10a567341bfac6da0a8f04dc41d
at block: 302 (Thu, 12 Mar 2020 12:26:32 CET)
 datadir: /home/alastria/test-environment/infrastructure/testnet/network/general1
 modules: admin:1.0 debug:1.0 eth:1.0 istanbul:1.0 miner:1.0 net:1.0 personal:1.0 rpc:1.0 txpool:1.0 web3:1.0

> web3.eth.accounts
["0x74d4c56d8dcbc10a567341bfac6da0a8f04dc41d"]
> web3.personal.unlockAccount(web3.eth.accounts[0], "Password",0);
(anonymous): Line 1:51 Unexpected token ILLEGAL (and 3 more errors)
> web3.personal.unlockAccount(web3.eth.accounts[0], "Password",0)
(anonymous): Line 1:51 Unexpected token ILLEGAL (and 3 more errors)
```

```

> web3.eth.accounts
["0x74d4c56d8dcbc10a567341bfac6da0a8f04dc41d"]
> web3.personal.unlockAccount(web3.eth.accounts[0], 'Passw0rd',0)
true

```

Pero sigue sin funcionar correctamente. He comprobado como la versión Web3 que ofrece la red tesnet es mucho más antigua que la utilizada en el proyecto:

```

> web3.personal.unlockAccount(
... ^C
> web3.personal.unlockAccount("0x442ba86d2da230fded071b497d1395e171dfc076","jose
")
true
> web3.version
{
  api: "0.20.1",
  ethereum: "0x40",
  network: "9535753591",
  node: "Geth/generall/v1.7.2-stable-94ele31e/linux-amd64/go1.9.5",
  whisper: undefined,
  getEthereum: function(callback),
  getNetwork: function(callback),
  getNode: function(callback),
  getWhisper: function(callback)
}
> web3.eth.personal.newAccount("jose")
TypeError: Cannot access member 'newAccount' of undefined
    at <anonymous>:1:1
> web3.version.api
"0.20.1"
>

```

Ilustración 9 Web3 utilizada por la testnet

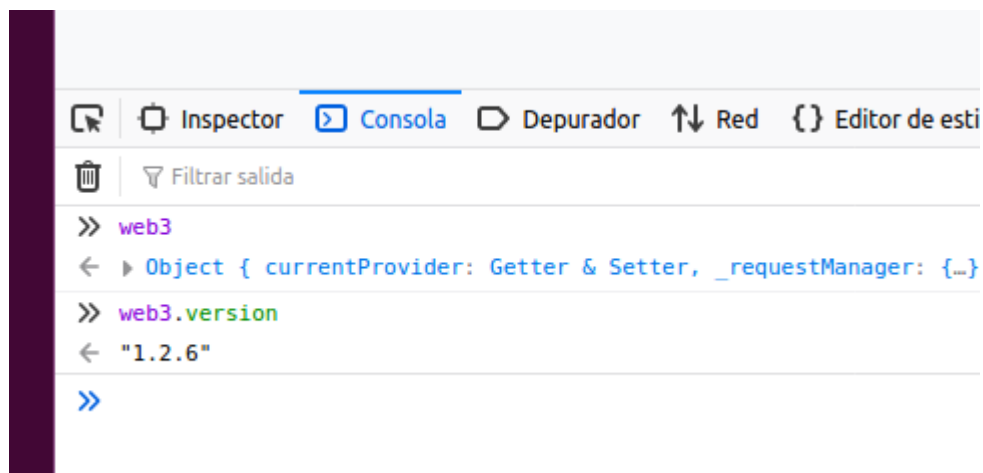



Ilustración 10 Web3 utilizada por la aplicación

Además de este aspecto, se experimentan muchos problemas debidos a las llamadas entre Smart Contracts que se han desarrollado en la solución, y el comentario del profesor José Luis Nieto en el foro me hizo desestimar el continuar con esta red:



[Re: Consumo de gas en nodo alastria local](#)
JOSE LUIS NIETO GARCIA (mar 11, 2020 9:17 PM) - Leído por: 7
[Marcar como no leído](#) [Responder](#)

Buenas,

Consumen mucho gas en el nodo local o en el nodo online?

Se llega a ejecutar la transacción o siempre se queda sin gas la ejecución?

Creo que se debe a un problema con el nodo local del Alastria, la versión anterior de Alastria no permitía llamar a un contrato desde otro contrato.

Un Saludo.

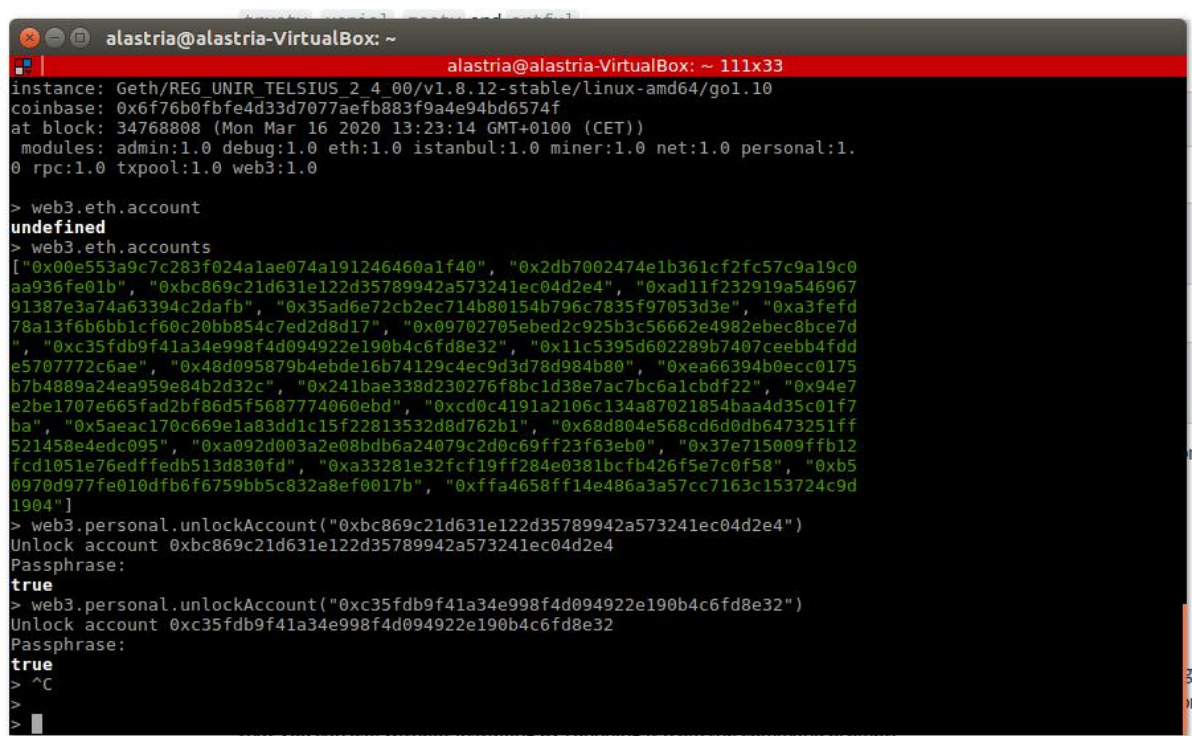
Alastria

Respecto a la realización de las pruebas con el framework truffle, he iniciado el trabajo instalado la versión 1.9.12 de GETH en local, se ha instalado correctamente y todo parece haber funcionado bien.

Posteriormente me he enganchado a su consola con

```
sudo geth attach http://10.141.8.11:8545
```

La conexión ha sido correcta y he podido desbloquear las cuentas:



```
alastria@alastria-VirtualBox: ~
instance: Geth/REG_UNIR_TELSIUS_2_4_00/v1.8.12-stable/linux-amd64/gol.10
coinbase: 0x6f76b0fbfe4d33d7077aefb883f9a4e94bd6574f
at block: 34768808 (Mon Mar 16 2020 13:23:14 GMT+0100 (CET))
modules: admin:1.0 debug:1.0 eth:1.0 istanbul:1.0 miner:1.0 net:1.0 personal:1.0
rpc:1.0 txpool:1.0 web3:1.0

> web3.eth.account
undefined
> web3.eth.accounts
[["0x00e553a9c7c283f024a1ae074a191246460a1f40", "0x2db7002474e1b361cf2fc57c9a19c0
aa936fe01b", "0xbc869c21d631e122d35789942a573241ec04d2e4", "0xad11f232919a546967
91387e3a74a63394c2dafb", "0x35ad6e72cb2ec714b80154b796c7835f97053d3e", "0xa3fed
78a13f6b6bb1cf60c20bb854c7ed2d8d17", "0x09702705ebed2c925b3c56662e4982ebec8bce7d
", "0xc35fdb9f41a34e998f4d094922e190b4c6fd8e32", "0x11c5395d602289b7407ceebb4fdd
e5707772c6ae", "0x48d095879b4ebde16b74129c4ec9d3d78d984b80", "0xea66394b0ecc0175
b7b4889a24ea959e84b2d32c", "0x241bae338d230276f8bc1d38e7ac7bc6a1cbdf22", "0x94e7
e2be1707e665fad2bf86d5f5687774060ebd", "0xcd0c4191a2106c134a87021854baa4d35c01f7
ba", "0x5aeac170c669e1a83dd1c15f22813532d8d762b1", "0x68d804e568cd6d0db6473251ff
521458e4edc095", "0xa092d003a2e08bdb6a24079c2d0c69ff23f63eb0", "0x37e715009ffb12
fcd1051e76edffedb513d830fd", "0xa33281e32fcf19ff284e0381bcfb426f5e7c0f58", "0xb5
0970d977fe010dfb6f6759bb5c832a8ef0017b", "0xffa4658ff14e486a3a57cc7163c153724c9d
1904"]
> web3.personal.unlockAccount("0xbc869c21d631e122d35789942a573241ec04d2e4")
Unlock account 0xbc869c21d631e122d35789942a573241ec04d2e4
Passphrase:
true
> web3.personal.unlockAccount("0xc35fdb9f41a34e998f4d094922e190b4c6fd8e32")
Unlock account 0xc35fdb9f41a34e998f4d094922e190b4c6fd8e32
Passphrase:
true
> ^C
>
```

Posteriormente he intentado ejecutar los scripts de truffle desde la consola.

```
$truffle compile --all
```

Ha funcionado correctamente sin mayores problemas y he intentado la migración ejecutando

```
$truffle migrate --network ganache --reset
```

En este caso no ha funcionado, devuelve el siguiente error:

```
Truffle v5.0.18 (core: 5.0.18)
Node v8.17.0
alastria@alastria-VirtualBox:~/Escritorio/UNIR_TFE$ truffle migrate --network alastria --reset

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Starting migrations...
=====
> Network name:      'alastria'
> Network id:        83584648538
> Block gas limit:   0x29bbc291

1 initial_migration.js
=====

Deploying 'Migrations'
-----
> transaction hash:  0x76e97b0f2b5261c9e15869d4b89e415299102214b4c8cff5a79fb3e6d5515df0
Error: Error: Error: *** Deployment Failed ***

"Migrations" received a generic error from Geth that
can be caused by hitting revert in a contract constructor or running out of gas.
* Returned error: gas required exceeds allowance or always failing transaction.
* Try: + using the '--dry-run' option to reproduce this failure with clearer errors.
      + verifying that your gas is adequate for this deployment.

    at Object.run (/usr/lib/node_modules/truffle/build/webpack:/packages/truffle-migrate/index.js:84:1)
    at <anonymous>
    at process._tickCallback (internal/process/next_tick.js:189:7)
Truffle v5.0.18 (core: 5.0.18)
Node v8.17.0
```

He estado jugueteando con el nivel de gas sin éxito. Finalmente he optado por utilizar la opción "--dry-run" como indica, el resultado ha sido:

```
Truffle v5.0.18 (core: 5.0.18)
Node v8.17.0
alastria@alastria-VirtualBox:~/Escritorio/UNIR_TFE$ truffle migrate --network alastria --reset --dry-run

Compiling your contracts...
=====
> Everything is up to date, there is nothing to compile.

Migrations dry-run (simulation)
=====
> Network name:      'alastria-fork'
> Network id:        83584648538
> Block gas limit:   0x29b926d6

1 initial_migration.js
=====

Deploying 'Migrations'
-----
Error: Error: Error: *** Deployment Failed ***

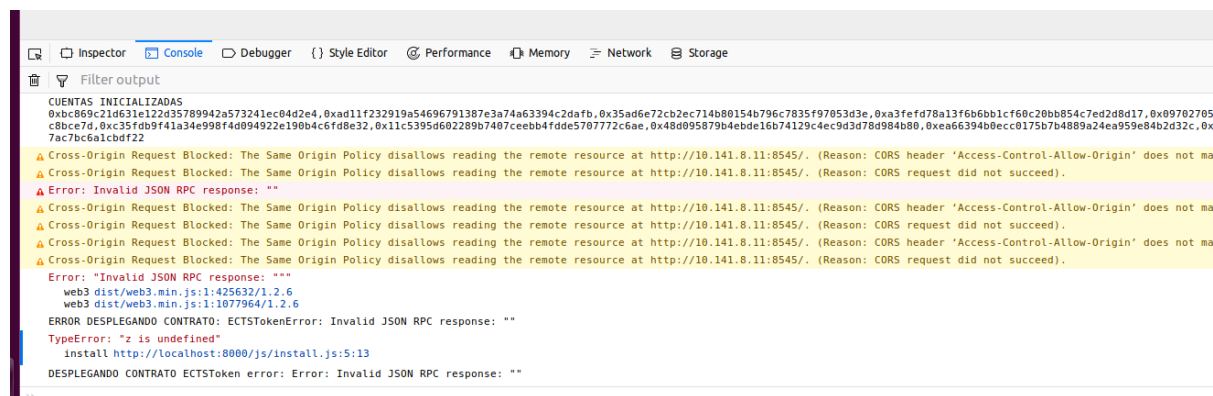
"Migrations" could not deploy due to insufficient funds
* Account:  0xc35fdb9f41a34e998f4d094922e190b4c6fd8e32
* Balance:  0 wei
* Message:  sender doesn't have enough funds to send tx. The upfront cost is: 2097150000000000 and the sender's account only has:
0
* Try:
  + Using an adequately funded account
  + If you are using a local Geth node, verify that your node is synced.

    at Object.run (/usr/lib/node_modules/truffle/build/webpack:/packages/truffle-migrate/index.js:84:1)
    at <anonymous>
    at process._tickCallback (internal/process/next_tick.js:189:7)
Truffle v5.0.18 (core: 5.0.18)
```

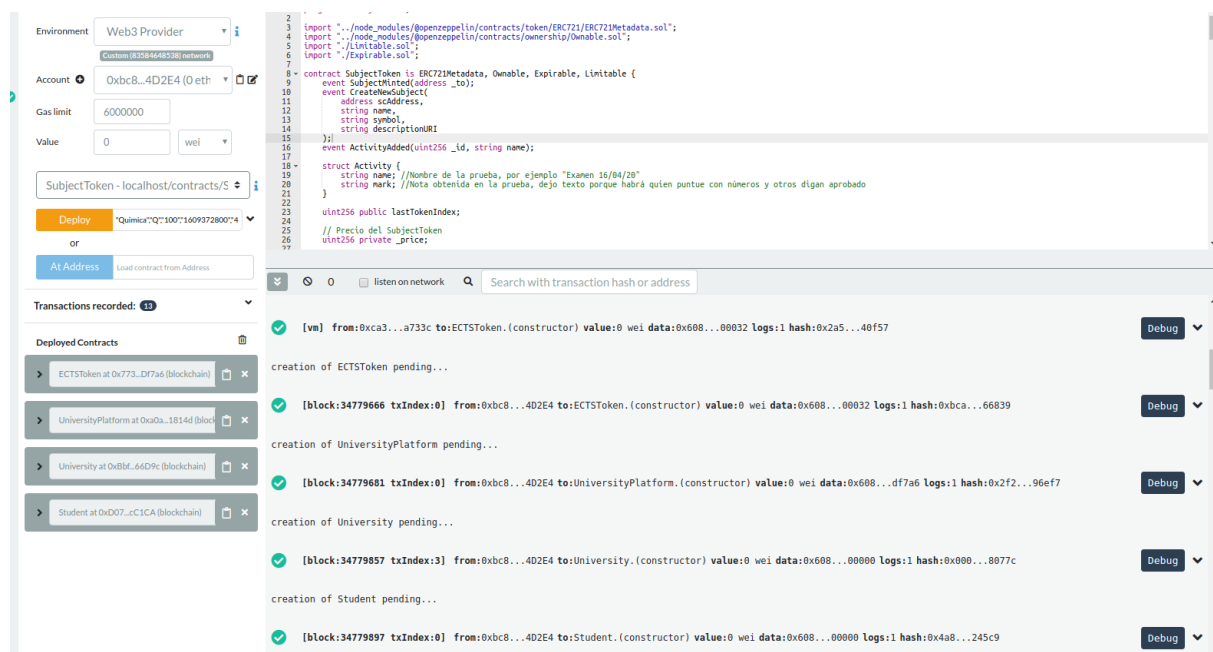
Parece indicar que la cuenta no tiene ether suficiente para hacer el despliegue.

He probado con otras cuentas y el resultado siempre ha sido el mismo, por lo que entiendo que se debe al uso de la opción "--dry-run".

He intentado otro camino, ya que la aplicación autodespliega, he intentado ejecutarla configurando el end-point de la UNIR, <http://10.141.8.11:8545>, pero el resultado ha sido un problema de CORS.

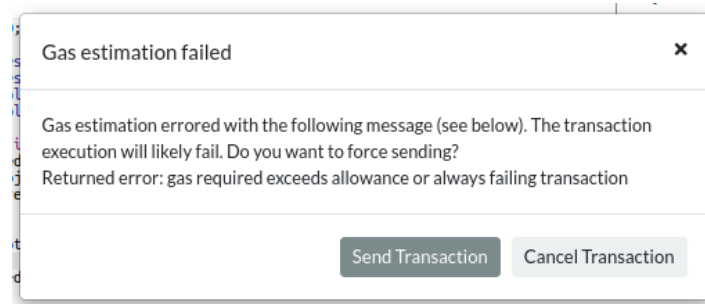


Finalmente he intentado realizar el despliegue manualmente desde remix contra <http://10.141.8.11:8545> y en este caso se han desplegado correctamente los siguientes SC:



No se aprecia pero las transacciones han sido correctas para ECTSToken.sol, UniversityPlatform.sol, University.sol y Student.sol.

El último de los SC que se ha desarrollado ha fallado indicando un problema de GAS, "*gas required exceeds allowance or always failing transaction*".



He estado realizando diversas pruebas con distintos parámetros de gas, pero finalmente no he podido encontrar la solución.

SubjectToken - localhost/contracts:

Deploy

name: "Química"

symbol: "Q"

limitmint: "100"

expirationtime: "1609372800"

price: "4"

descriptionURI: "http://www.unir.net"

transact

or

At Address

Load contract from Address

Transactions recorded:

Deployed Contracts

> ECTSToken at 0x773...Df7a6 (Blockchain)

> UniversityPlatform at 0xa0a...1814d (Blockchain)

> University at 0xbBf...66D9c (Blockchain)

18
19
20
21
22
23
24
25
26
27

```
struct Activity {  
    string name; //Nombre de la prueba, por ejemplo "Examen 16/04/20"  
    string mark; //Nota obtenida en la prueba, dejo texto porque habrá quien puntue con números y otros digan aprobado  
}  
  
uint256 public lastTokenIndex;  
// Precio del SubjectToken  
uint256 private _price;
```

0 ☐ listen on network Search with transaction hash or address

[block:34780059 txIndex:0] from:0xb8...4D2E4 to:SubjectToken.(constructor) value:0 wei data:0x608...00000 logs:0 hash:0xbbf...c9244

status

false Transaction mined but execution failed

transaction hash

0xbbf0d1747fe353a6a30755dd52e3f594eaa853cc1d743d6a854b0fe8a828c9244

from

0xb8c869c21d631e122d35789942a573241ec94D2E4

to

SubjectToken.(constructor)

gas

2000000 gas

transaction cost

2000000 gas

hash

0xbbf0d1747fe353a6a30755dd52e3f594eaa853cc1d743d6a854b0fe8a828c9244

input

0x608...00000

decoded input

{
 "string name": "Química",
 "string symbol": "Q",
 "uint256 limitmint": {
 "_hex": "0x64"
 },
 "uint256 expirationtime": {
 "_hex": "0x3fed1480"
 },
 "uint256 price": {
 "_hex": "0x94"
 },
 "string descriptionURI": "http://www.unir.net"
}
}

decoded output

-

logs

[]

value

0 wei

10 Conclusiones

Las cadenas de bloques son una de las tecnologías que ha cobrado mayor popularidad en los últimos años, sin embargo, muchos de los proyectos que utilizan cadenas de bloques son iniciados de manera apresurada y sin detenerse a pensar si es realmente necesario el hacer uso o no de dicha tecnología.

La tecnología dista todavía de estar madura, prueba de ello es la enorme sensibilidad a las versiones y combinación de las mismas que el sistema presenta. Por lo que a mi proyecto respecta, intenté salir a la última, y al final he tenido muchos problemas de compatibilidad en las diferentes redes.

Blockchain es una realidad, y puede complementar y mejorar muchas de las aplicaciones que en la actualidad se están construyendo.

Existen herramientas y un framework de desarrollo sólido para desarrollar sobre esta tecnología.

Respecto al TFE, creo que ha sido la experiencia más dura y al tiempo formativa que se ha desarrollado en el curso. Si el resto del curso pecaba de demasiado teórico, el TFE ha sido un duro golpe contra la realidad. Constructivamente me atrevería a recomendar más tiempo para su desarrollo en futuras ediciones y evitar los plazos tan estrictos de tiempo, creo que casi todo el alumnado nos encontramos trabajando y robarle tiempo a la familia, a las noches y al tiempo libre no es sencillo, y es una lastima que no podamos acabar de concretar el proyecto, no obstante estoy muy contento con lo que he aprendido con él.

Muchísimas gracias por todo.

11 Referencias

- S. Nakamoto, "Bitcoin open source implementation of P2P currency". <http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source>
- FTP016: Vitalik Buterin – What is Ethereum and How to Build a Decentralized Future. <https://futurethinkers.org/vitalik-buterin-ethereum-decentralized-future/>
- Quorum Blockchain – Guía Definitiva. <https://101blockchains.com/es/quorum-blockchain-guia/>
- Los créditos ECTS. – Universidad de las Palmas de Gran Canaria - <https://www2.ulpgc.es/index.php?pagina=ECTS&ver=loscreditos>
- ¿Necesitas una blockchain? – 101Blockchain.com - <https://101blockchains.com/es/necesitas-una-solucion-blockchain/>
- Usar o no blockchain. – SG - <https://sg.com.mx/revista/57/usar-o-no-blockchain>
- ERC223 – Dexaran - <https://github.com/Dexaran/ERC223-token-standard>