

UNIR  
Experto Universitario en Desarrollo  
de Aplicaciones Blockchain

# Títulos Académicos Tokenizados

## TFE

Versión: v1.1  
Autor: Javier Montesinos Morcillo  
Última actualización: 16 de marzo de 2020

Tutor: José Luis Nieto



## Historial de Cambios

Fecha	Versión	Autor	Breve Descripción
15/03/2020	v1.0	Javier Montesinos	Creación del documento.
16/03/2020	v1.1	Javier Montesinos	Revisión antes de su entrega.

A mi mujer Ketí y a mis hijos Javier y Raúl,  
por la paciencia infinita que han tenido conmigo durante el tiempo  
que ha durado este Experto apoyándome en todo momento y con mayor énfasis  
en sus últimas semanas para la elaboración de este TFE.

Gracias por vuestro apoyo incondicional...

# Índice

<b>1. Introducción</b>	<b>5</b>
<b>2. Estado del Arte</b>	<b>5</b>
2.1. Premisas sobre Títulos Académicos	5
2.2. Alcance del TFE	6
2.3. RGPD	9
2.4. Más allá del TFE	9
<b>3. Tecnologías</b>	<b>10</b>
<b>4. Funcionalidad / Casos de uso</b>	<b>14</b>
4.1. Inicializar el sistema	15
4.2. Acceso como universidad, profesor o alumno	17
4.3. Cálculo de tokens necesarios para matricular en una asignatura	18
4.4. Comprar Tokens ECTS	18
4.5. Matricular en Asignatura	20
4.6. Evaluar Asignatura no aprobada por parte de un profesor	22
4.7. Evaluar Asignatura aprobada por parte de un profesor	23
4.8. Trasladar matrícula a otra Universidad	24
4.9. Diagramas de secuencia	26
<b>5. Arquitectura / Modelo del sistema</b>	<b>28</b>
5.1. Controlador	28
5.2. Modelo	28
5.3. Vista	29
<b>6. Despliegue</b>	<b>30</b>
6.1. Desarrollo (Ganache)	30
6.1.1. Instalar ganache-cli	30
6.1.2. Lanzar ganache-cli	30
6.1.4. Desplegar los Smart Contracts en la red	31
6.1.5. Iniciar la aplicación web	33
6.1.5.1. SimpleHTTPServer	33
6.1.5.2. Angular CLI	33
6.1.5.3. Versiones Angular CLI	34
6.1.6. Acceder a la aplicación web	34
6.2. Preproducción (Testnet) / Producción (Alastria Nodo UNIR)	35
<b>7. Testing</b>	<b>38</b>
<b>8. Conclusiones</b>	<b>39</b>
<b>9. Bibliografía</b>	<b>40</b>

# 1. Introducción

El objetivo de este documento es recoger los aspectos de diseño, desarrollo, testing y despliegue del Trabajo Final de Experto (en adelante TFE) del curso Experto Universitario en Desarrollo de Aplicaciones Blockchain en su promoción de Octubre/19 a Marzo/20.

El objetivo del TFE ha sido aplicar los conocimientos adquiridos a lo largo del curso para desarrollar un sistema basado en Blockchain que permita tokenizar el sistema de títulos universitarios de forma que los propios alumnos puedan:

- Matricularse en asignaturas mediante la adquisición de tokens ERC20.
- Disponer de las asignaturas como tokens ERC721 que puedan acreditar que han completado una asignatura concreta así como en que universidad lo han realizado y con que nota entre otra información.

Como se verá a lo largo de este documento el sistema desarrollado consta de 2 elementos diferenciados:

- Backend, basado en código Solidity desarrollado con Remix, Truffle, Ganache que usa como base la librería Open Zeppelin que se despliega sobre la red Blockchain.
- Frontend, basado en una aplicación angular (en su versión 8) que utiliza web3.js para interactuar con el backend disponible en la red blockchain y LocalStorage para simular la persistencia del aplicativo.

## 2. Estado del Arte

Para abordar el desarrollo de este TFE se ha analizado el sistema de créditos y títulos de grado de ámbito nacional, no obstante la solución propuesta bien podría ser de aplicación a los títulos emitidos por cualquier otra universidad tanto Europea como en el resto del mundo con el ajuste y configuración que pudiera ser necesario.

### 2.1. Premisas sobre Títulos Académicos

Los créditos de los que consta un grado suelen ser en líneas generales de 60 por curso, por lo tanto en un grado de 4 cursos el alumno debería completar 240 créditos.

Consideraremos que para una misma asignatura en función de su grado de Experimentalidad (que inicialmente podemos considerar entre 1 y 4) un crédito en una misma universidad pueda tener un precio diferente según el grado en el que nos matriculamos o la asignatura que consideremos.

Por otro lado cuando un alumno vuelve a matricularse de una asignatura por segunda y sucesivas veces, esa matrícula también suele ser más cara que las anteriores y ese precio varía en función de la universidad en la que nos matriculamos.

Por último buscamos identificar un sistema en el que cada universidad pueda ofertar sus títulos y por lo tanto sus asignaturas según el precio que tenga establecido en base a sus recursos, profesorado....

Según los puntos anteriores parece lógico pensar en un sistema en el que los tokens adquiridos en una universidad donde son más económicos que en otras no se puedan utilizar para matricularse en otra universidad(universidades) cuyos tokens son más caros.

Así mismo se plantea un sistema en el que una vez el Estado haya registrado a las diferentes universidades que tienen acceso, estas puedan de una forma sencilla establecer un precio distinto para:

- Cada grado de experimentalidad de las asignaturas / grados contemplados en el sistema.
- Cada año de matrícula realizado por un alumno en una asignatura.
- Precio para cada Token ERC20 que permita a los alumnos matricularse en asignaturas y títulos.

## 2.2. Alcance del TFE

Se plantea el uso de Blockchain como Backend de un sistema en el que se utilicen tokens basados en el estándar ERC20 para la matriculación de los alumnos en asignaturas y del estándar ERC721 para la identificación de los alumnos en las diferentes asignaturas de las que constará el sistema proporcionado.

La creación de estos tokens permitiría una operativa rápida, sencilla, segura y de confianza entre estado, universidades, profesores y alumnos.

Para representar los tokens ERC20 (en adelante tokens ECTS) que los alumnos deberán adquirir para su matriculación se ha creado basado en el ERC20 de Open Zepellin una implementación particular que incluye un libro de registro similar al utilizado en el estándar para la gestión de los allowances de cada propietario pero en este caso aporta el balance en tokens ECTS que cada alumno tiene para cada una de las universidades. De tal forma que se ha ampliado el método `_transfer` para recoger esta operativa.

Esta implementación está recogida en los smart contracts: `ECTSERC20.sol` (implementación personalizada del ERC20) y `ECTSToken.sol` (implementación particular del token ECTSToken)

Para recoger cada asignatura que estará disponible en el sistema (`ERC721.sol` y `ERC721Metadata.sol`) se plantea un token ERC721 personalizado. Se ha añadido como información adicional al token ERC721 aquella que se considera necesaria desde el punto de vista de la matriculación de los alumnos y que está recogida como struct en el smart contract `AsignaturaToken.sol`.

```
struct Asignatura {  
    address universidad;  
    address profesor;  
    address alumno;  
    string cursoAcademico;  
    uint256 anioMatricula;  
    uint256 nota;  
    bool aprobado;  
    bool evaluado;  
    bool valida;  
}
```

Adicionalmente también se recoge la información necesaria en relación a las matrículas de los diferentes alumnos, sus años de matrícula, universidades y profesores que pueden impartir la asignaturas o los créditos de la asignatura, su experimentalidad o nota mínima para su aprobado.

```
uint256 private _creditos;  
uint256 private _experimentabilidad;  
uint256 internal _notaMinimaAprobado = 500;  
mapping (uint256 => Asignatura) private _matriculas;  
mapping (address => uint256) private _aniosMatricula;  
mapping (address => address) private _universidadesProfesores;
```

Por último se considera la creación de un smart contract que representa al Estado (Estado.sol) y que es el encargado de registrar Universidades, Profesores y Alumnos on-chain para su interacción con el sistema así como del despliegue en la red de los smart contracts ERC721 que representan a las diferentes asignaturas y que se crean bajo demanda por parte de las universidades al Estado.

En este smart contract las universidades tiene la posibilidad de configurar el precio en weis de cada ECTS necesario para matricularse en sus asignaturas así como de los factores de corrección necesarios para ponderar el número de ECTS necesarios para matricularse en una asignatura un alumno basado en la experimentalidad de esta y del año de matrícula del alumno. Por simplicidad se establecen 4 posibilidad para cada una de las perspectivas:

- Experimentalidad: Grado 1, Grado 2, Grado 3 y Grado 4 o superior
- Año de matrícula: Año 1, Año 2, Año 3 y Año 4 o sucesivo.

A continuación se muestra mediante un ejemplo el sistema planteado para adquirir tokens y matricularnos en una asignatura.

Supongamos una asignatura “Cálculo Numérico de 1” que imparte la Universidad UNEX en el Grado de Matemáticas que consta de 12 créditos y cuyo grado de Experimentalidad es 1 y en la que se va a matricular por 3º año el alumno Antonio Nogales Torres.

En el momento de redactar el documento el precio de 1 Ether es de 110,81€ por lo tanto si consideramos que el precio del crédito en UNEX pueda ser 12 euros -> el precio de cada ECTS sería de 0,11 ETH o lo que es lo mismo 110000000000000000 Weis.

El Token ECTS utiliza 4 decimales para contemplar los cálculos realizados por las universidades para su ponderación así pues realmente si necesitamos 1 crédito -> necesitaremos disponer de 10.000 tokens ECTS, por esto mismo el valor mínimo de un token ECTS valdrá:  $110000000000000000 / 10000 = 11000000000000$  weis.

Inicialmente consideramos que UNEX tiene configurados los siguientes factores de corrección de los ECTS necesarios:

- Grados de experimentalidad:
  - Grado 1: 100 (simularía NO incrementar el precio)
  - Grado 2: 110 (simularía incrementar un 10% el precio)
  - Grado 3: 120 (simularía incrementar un 20% el precio)
  - Grado 4: 130 (simularía incrementar un 30% el precio)
- Años de matrícula:
  - Año 1: 100
  - Año 2: 110
  - Año 3: 120
  - Año 4 y superior: 130

Para calcular los tokens ECTS necesarios para que Antonio Nogales pueda matricularse en Cálculo Numérico de 1 necesitaría:

$$\begin{aligned} \text{Número de créditos} \times \text{Factor Grado de Exp.} \times \text{Factor 3º año de matrícula} &= \\ 12 \times 100 \times 120 &= 144000 \text{ i.e. } 14,4 \text{ tokens ECTS} \end{aligned}$$

Basado en el precio que UNEX tiene establecido para cada ECTS el alumno debería pagar por esos tokens:

$$144000 \times 110000000000000000 \text{ weis} = 1,584 \times 10^{18} \text{ weis} = 1,584 \text{ ETH.}$$

Una vez que el alumno solicita al estado que desea adquirir los 14,4 tokens sobre el contrato Estado aportando los weis necesarios el estado debe transferir de la universidad al alumno los tokens adquiridos y añade al balance de la universidad los weis entregados por el alumno.

Para realizar esta operativa de una forma sencilla en la implementación realizada de ERC20 se ha contemplado una modificación para que en el caso de ser el estado el msg.sender de la transferencia no se verifique la aprobación necesaria para realizar la transferencia de tokens y que la operativa sea más simple ya que de lo contrario se debería generar una



petición de traspaso de tokens con una bolsa de los weis facilitados de forma que cuando la universidad los transfiera al alumno el estado traslade los weis a la universidad con esto se consigue que en una sola transacción se complete el proceso.

De la misma forma sobre la implementación particular del ERC721 se ha verificado que si el msg.sender que transfiere tokens es el estado no se verifique la aprobación necesaria.

El balance en ambos casos es del alumno pero los tokens son transferidos por el smart contract (Estado.sol) intermedio que representa al Estado.

En el caso de las asignaturas ERC721 creadas para representar la matrícula de un alumno en una asignatura inicialmente en el momento de matricularse el alumno se asignan a la universidad recogiendo la información necesaria del alumno. Tan solo en el momento en el que el profesor autorizado evalúa la nota definitiva del alumno se traspasa al token al alumno si la nota es superior al valor configurado como nota mínima para aprobar sobre el SC Asignatura correspondiente.

Para recoger la posibilidad de evaluar notas con decimales se plantea un sistema basado en dos decimales de forma que una nota registrada de 1000 corresponde con un 10.

## 2.3. RGPD

El planteamiento inicial era recoger en las estructuras que representan a universidades, alumnos y profesores on-chain, datos como nombre, apellidos, email... por RGPD y dado el carácter público de los datos on-chain se opta por registrar exclusivamente las direcciones asignadas a cada uno de ellos de forma que es en la BD de la aplicación (como es lógico) en donde se recoge esta información.

Para la persistencia utilizada en el proyecto por simplicidad como se verá más adelante en este documento se ha optado por LocalStorage del navegador.

## 2.4. Más allá del TFE

Como evolución de este TFE se plantea la posibilidad de contemplar los Grados on-chain y disponer de identidades digitales asociadas al alumno, profesor y universidad de forma que:

- Cuando el profesor evalúa con una nota aprobada una asignatura para un alumno, además de transferir al alumno el Token ERC721 correspondiente, se emita un CLAIM desde la identidad del profesor a la del alumno alegando su aprobación.
- Cuando un profesor evalúa con una nota aprobada sobre una asignatura y esta es la última asignatura de un grado pendiente para el alumno se emite una petición de emisión de título a la universidad para que utilizando su identidad digital emite una CLAIM alegando que el alumno ha obtenido el grado.

### 3. Tecnologías

A continuación se recogen las tecnologías utilizadas en el desarrollo de este TFE así como una breve descripción de cada una de ellas y de el uso concreto realizado en el proyecto.



#### **Solidity**

Solidity es un lenguaje de alto nivel orientado a contratos. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la Máquina Virtual de Ethereum (EVM). Ha sido el lenguaje utilizado para desarrollar los smart contracts del proyecto Estado.sol, ECTSToken.sol....

#### **Remix**

Remix es un IDE de código abierto que ayuda a escribir contratos de Solidity directamente desde el navegador. Está escrito en JavaScript, y admite el uso en el navegador y localmente. Contempla entre otros; pruebas, depuración e implementación de contratos inteligentes así como la conexión con redes de prueba, Ganache, Testnets... Inicialmente los smart contracts del TFE se han ido creando y probando mediante Remix, para pasar posteriormente a Truffle



#### **Truffle**

Truffle es un entorno de desarrollo para Ethereum, su objetivo principal es facilitar las tareas como desarrollador sobre Ethereum. Con Truffle, se dispone de forma sencilla de compilación, despliegue y testing de contratos inteligentes en diferentes redes y mediante configuraciones concretas necesarias para cada una de ellas. Se utiliza en este TFE para el desarrollo de todos los smart contracts así como para su despliegue y su testing.



## Ganache

Ganache es una sencilla herramienta para disponer principalmente en entornos de desarrollo de una cadena de bloques personal para el desarrollo de Ethereum que se puede usar para implementar contratos, desarrollar sus aplicaciones y ejecutar pruebas. Está disponible tanto como una aplicación de escritorio como una herramienta de línea de comandos.



En este TFE se ha utilizado la versión de línea de comandos por su simplicidad y rapidez de inicio, configuración...



## Testnet Alastria (Quorum)

Basado en Ethereum, Quorum es una plataforma blockchain de código abierto que combina la innovación de la comunidad pública de Ethereum con mejoras para satisfacer las necesidades de la empresa permitiendo entre otras características transacciones privadas.

La red Testnet de Alastria está basada en esta red y es la que se ha utilizado para verificar el despliegue en un entorno productivo basado en Quorum del TFE.

## web3.JS

web3.js es una colección de bibliotecas Javascript que permiten interactuar con un nodo ethereum local o remoto, utilizando una conexión HTTP o IPC.

Se ha utilizada esta librería para que el frontend del proyecto basado en una aplicación web interactúe con el backend basado en blockchain.





### LocalStorage

El objeto Storage (Local y Session) de los navegadores (API de almacenamiento web) nos permite almacenar datos de manera local en ellos sin necesidad de realizar conexión alguna a una base de datos.

Por simplicidad se ha utilizado este sistema de almacenamiento para persistir la información del TFE que la aplicación web necesita, como por ejemplo los datos de universidades, alumnos, profesores o asignaturas y sus correspondientes cuentas 0x... de acceso.

**Si bien no es un sistema seguro para almacenar los datos siguiendo el modelo del prototipo [propuesto](#) por Origin Protocolos se ha optado por esta opción por su simplicidad. En un entorno productivo real esta información se debería almacenar en una BD como pueda ser PostgreSQL a la que se debería acceder mediante un API Rest que podría estar desarrollada en Node, Java o PHP.**

### Github

GitHub es un servicio de alojamiento de repositorios Git para el control de código fuente. Si bien Git es una herramienta de línea de comandos, GitHub proporciona una interfaz gráfica basada en la Web. También proporciona control de acceso y varias funciones de colaboración, como wikis y herramientas básicas de administración de tareas para cada proyecto.



Se han utilizado dos repositorios en el TFE uno para el backend blockchain: [unir-tfe-open-zeppelin](#) y otro para la aplicación web: [unit-tfe-dapp](#)



### Visual Studio Code

Visual Studio Code es un editor de código que se puede usar con una amplia variedad de lenguajes de programación, incluidos Solidity, JavaScript, HTML, Node.js... Se ha utilizado este IDE para el desarrollo tanto de la parte de backend basada en blockchain como de la aplicación web.

## Angular

Angular es un framework para el desarrollo de aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página.

Se ha utilizado para el desarrollo de toda la parte correspondiente en el proyecto a la Dapp o aplicación web.



## Bootstrap

Bootstrap es una biblioteca de código abierto para diseño de sitios y aplicaciones web.

Se ha utilizada para maquetado y estilo del diseño del front de la aplicación web.

## 4. Funcionalidad / Casos de uso

En este apartado se recoge una guía de la aplicación desarrollada sobre los principales casos de uso contemplados.

A continuación se muestra un diagrama UML de todos los casos de uso que implementa el desarrollo sobre el proyecto backend Blockchain que permitirían a las diferentes entidades del sistema su interacción y parametrización como es el caso del Estado y las Universidades. No obstante tan solo se han desarrollado sobre la aplicación web aquellos caso de uso que intervienen de forma directa en el proceso de matriculación de un alumno en una asignatura y su posterior traslado de expediente una vez aprobada a otra Universidad.

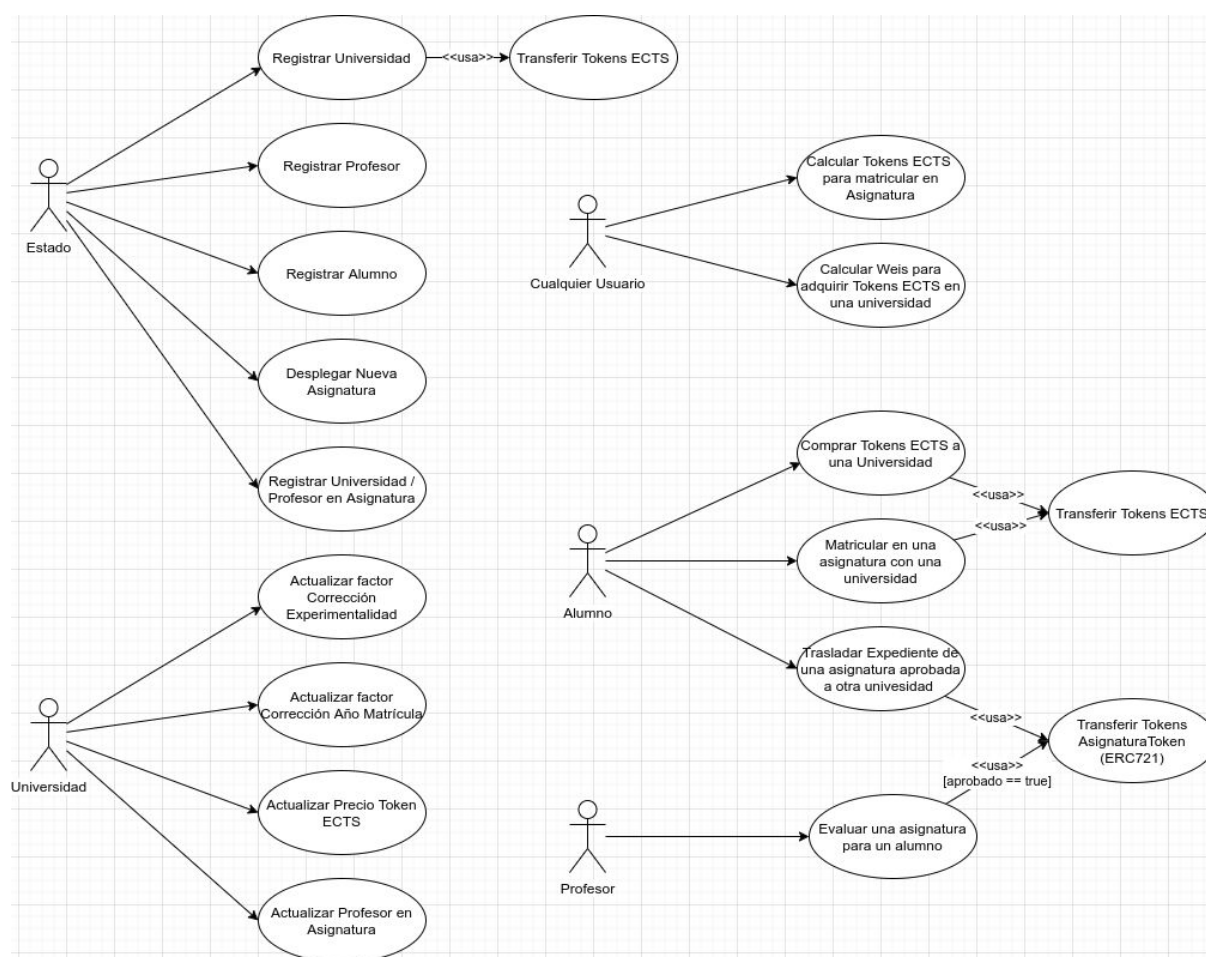
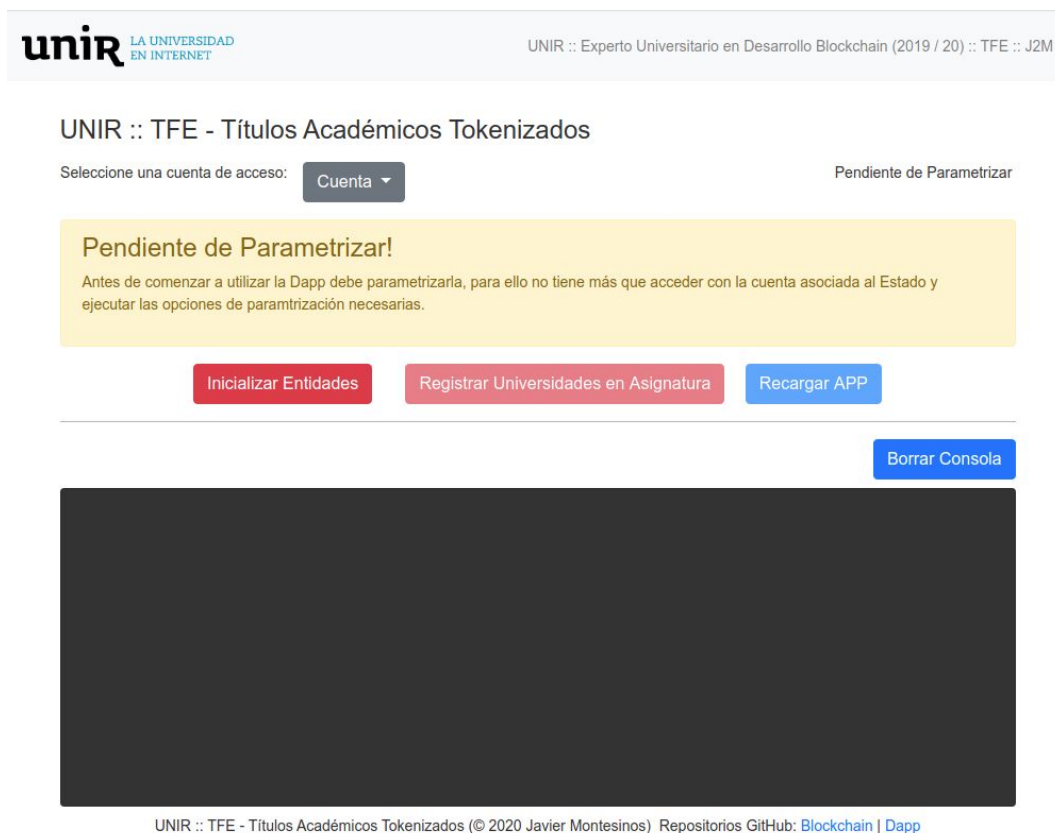


Diagrama UML de Casos de Uso

## 4.1. Inicializar el sistema

Lo primero que necesitamos es inicializar el sistema para lo cual debemos acceder con la cuenta asociada al estado y ejecutar los 3 sencillos pasos necesarios que se recogen en la imagen siguiente: Inicializar Entidades, Registrar Universidades en Asignaturas y por último Recargar la app.



La inicialización implica:

- Registrar dos universidades en el sistema: UNIR y UNEX
- Registrar un profesor Jose Luis Nieto
- Registrar un alumno Javier Montesinos
- Registrar dos asignaturas en el sistema Desarrollo de Aplicaciones Blockchain y Trabajo Final de Experto con 6 y 8 créditos respectivamente y un grado 4 de experimentalidad.

El precio en weis de 1 Token ECTS (0,0001 ECTS) de todas las asignaturas para ambas universidades se establece en: 6800000000000 weis


De la misma forma para comprobar la diferencia al matricularse en asignaturas con distinto grado de experimentalidad y en años diferentes al 1º se establecen para ambas universidades sus factores de corrección en:

- 100, 110, 120 y 130 para Experimentalidad 1, 2, 3 y 4 respectivamente y
- 100, 110, 120 y 130 para Años de matriculación 1º, 2º, 3º y 4º y posteriores respectivamente.



El registro de universidades implica las siguientes operaciones:

- Registrar a UNIR y UNEx con Jose Luis Nieto ambas como profesor en las dos asignaturas creadas. Esta información es necesaria para poder matricularse de una asignatura en una universidad.


UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

### UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: Cuenta ▾ Pendiente de Parametrizar

**Pendiente de Parametrizar!**

Antes de comenzar a utilizar la Dapp debe parametrizarla, para ello no tiene más que acceder con la cuenta asociada al Estado y ejecutar las opciones de parametrización necesarias.

Inicializar Entidades
Registrar Universidades en Asignatura
Recargar APP

Borrar Consola


```

Universidad registrada para la dirección: 0x951d593463525779200b850CA241Fc5c346FE950
Universidad registrada para la dirección: 0xE0812809290611D12EA7EcC7Cb700282755531BC
Profesor registrado para la dirección: 0x7a55FdcB796BA184fDA57530af3303b5553efC56
Alumno registrado para la dirección: 0xFc83780d8bc6eAE4DEe1f12F7976251D27530fD
Registrada asignatura: Desarrollo de Aplicaciones Blockchain (DAB),
en la dirección: 0x36Fb28597264c9e3c7b5Ed382578853c4149320d
Registrada asignatura: Trabajo Final de Experto (TFE),
en la dirección: 0xc73faC5CdDEB9459c40c905F75754c59A0a725B0

```

UNIR :: TFE - Títulos Académicos Tokenizados (© 2020 Javier Montesinos) Repositorios GitHub: [Blockchain](#) | [Dapp](#)

Por último tan solo nos queda cargar la app, para que se carguen los datos almacenados en LocalStorage.


UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

### UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: Cuenta ▾ Pendiente de Parametrizar

**Pendiente de Parametrizar!**

Antes de comenzar a utilizar la Dapp debe parametrizarla, para ello no tiene más que acceder con la cuenta asociada al Estado y ejecutar las opciones de parametrización necesarias.

Inicializar Entidades
Registrar Universidades en Asignatura
Recargar APP

Borrar Consola

```

Registrada asignatura: Trabajo Final de Experto (TFE),
en la dirección: 0xc73faC5CdDEB9459c40c905F75754c59A0a725B0
Registrada universidad / profesor:
0x951d593463525779200b850CA241Fc5c346FE950 / 0x7a55FdcB796BA184fDA57530af3303b5553efC56
en asignatura: 0x36Fb28597264c9e3c7b5Ed382578853c4149320d
Registrada universidad / profesor:
0xE0812809290611D12EA7EcC7Cb700282755531BC / 0x7a55FdcB796BA184fDA57530af3303b5553efC56
en asignatura: 0x36Fb28597264c9e3c7b5Ed382578853c4149320d
Registrada universidad / profesor:
0x951d593463525779200b850CA241Fc5c346FE950 / 0x7a55FdcB796BA184fDA57530af3303b5553efC56
en asignatura: 0xc73faC5CdDEB9459c40c905F75754c59A0a725B0
Registrada universidad / profesor:
0xE0812809290611D12EA7EcC7Cb700282755531BC / 0x7a55FdcB796BA184fDA57530af3303b5553efC56
en asignatura: 0xc73faC5CdDEB9459c40c905F75754c59A0a725B0

```

UNIR :: TFE - Títulos Académicos Tokenizados (© 2020 Javier Montesinos) Repositorios GitHub: [Blockchain](#) | [Dapp](#)



En la siguiente imagen se puede verificar los datos almacenados en LocalStorage:

Application	Filter	
Manifest	Key	Value
Service Workers	estado	"0x5C4756bb912Dea209B94587D4d761aCE5d321054"
Clear storage	universidades	[{"address":"0x951d59346352577920DbB5DCA241Fc5c346FE950"...
	profesores	[{"address":"0x7a55FdcB796BA184fDA57530af3303b5553efC56",...]
	alumnos	[{"address":"0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD"...
	entidades-registradas	true
LocalStorage	asignaturas	[{"address":"0x36Fb28597264c9e3c7b5Ed382578B53c414932Dd",...]
http://localhost:4200	universidades-registradas	true
Session Storage	app-recargada	true
IndexedDB		
Web SQL		
Cool		
Web SQL		

## 4.2. Acceso como universidad, profesor o alumno

Mediante el campo Dropdown “Cuenta” podemos simular el acceso del estado, una universidad, un profesor o un alumno al sistema tal como se muestra en la siguiente imagen de forma que se identifica el perfil seleccionado, el usuario así como su balance en ETH y ECTS.

LA UNIVERSIDAD EN INTERNET
 UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

### UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: **Cuenta** Bienvenid@ Estado ( 0x5C4756bb912Dea209B94587D4d761aCE5d321054 )  
118.800.000.000 ECTS ( 99,75 ETH )

Calcular ECTS

Calcular Tokens ECTS

Mediante esta funcionalidad puede calcular el balance de ECTS basado en el año de experimentabilidad y las asignaturas disponibles para verificar.

Asignaturas :

Universidad :

0x5C4756bb912Dea209B94587D4d761aCE5d321054  
0x951d59346352577920DbB5DCA241Fc5c346FE950  
0x7a55FdcB796BA184fDA57530af3303b5553efC56  
0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD  
0xE081280929D611D12EA7EcC7Cb700282755531BC

Desarrollo de Aplicaciones Blockchain ( DAB ) - Exp: 4 - Créditos: 6

UNIR

Las cuentas disponibles en el sistema están asignadas a los siguientes usuarios:


- Estado = '0x5C4756bb912Dea209B94587D4d761aCE5d321054';
- Universidad UNIR = '0x951d59346352577920DbB5DCA241Fc5c346FE950';
- Profesor - Jose Luis Nieto = '0x7a55FdcB796BA184fDA57530af3303b5553efC56';
- Alumno - Javier Montesinos = '0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD';
- Universidad UNEX = '0xE081280929D611D12EA7EcC7Cb700282755531BC';

La funcionalidad que puede ejecutar cada usuario en función de su rol es la siguiente:

- Funcionalidad pública: Calcular ECTS
- Funcionalidad para alumnos: Comprar ECTS, Matricular en Asignatura, Trasladar Asignatura aprobada
- Funcionalidad para profesores: Evaluar Asignatura

### 4.3. Cálculo de tokens necesarios para matricular en una asignatura

Seleccionando la asignatura podemos identificar el tipo de experimentalidad que tiene así como los créditos. Como verificamos que las dos asignaturas existentes suman 14 créditos y comparten grado de experimentalidad así como el año de matrícula para el alumno, se puede calcular los ECTS necesarios para adquirir esos 14 créditos


LA UNIVERSIDAD  
EN INTERNET
UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Selecione una cuenta de acceso: Cuenta ▾
Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD)  
0 ECTS ( 100,00 ETH)

Calcular ECTS
Comprar ECTS
Matricular
Trasladar


#### Calcular Tokens ECTS

Mediante esta funcionalidad puede calcular los tokens ECTS necesarios para matricularse en una asignatura en una universidad concreta basado en el año de experimentabilidad de la asignatura o en el año de matriculación. Para ello se muestra en el desplegable de asignaturas las asignaturas disponibles para verificar su experimentabilidad y los créditos necesarios.

Asignaturas :	Desarrollo de Aplicaciones Blockchain ( DAB ) - Exp: 4 - Créditos: 6 ▾
Universidad :	UNIR ▾
Experimentabilidad :	Exp. Grado >= 4 ▾
Año de matrícula :	1º Año ▾
Créditos :	14
	<span>Calcular ECTS</span>
Tokens ECTS Necesarios :	18,2

### 4.4. Comprar Tokens ECTS

A continuación se muestran tanto el balance tanto de UNIR como del alumno Javier Montesinos tanto en ECTS como en ETH para poder compararlos una vez el alumno adquiera los ECTS.


LA UNIVERSIDAD  
EN INTERNET
UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Selecione una cuenta de acceso: Cuenta ▾
Bienvenid@ Estado ( 0x5C4756bb912Dea209B94587D4d761aCE5d321054)  
118.800.000.000 ECTS ( 99,75 ETH)

Calcular ECTS

#### Calcular Tokens ECTS

Mediante esta funcionalidad puede calcular los tokens ECTS necesarios para matricularse en una asignatura en una universidad concreta basado en el año de experimentabilidad de la asignatura o en el año de matriculación. Para ello se muestra en el desplegable de asignaturas las asignaturas disponibles para verificar su experimentabilidad y los créditos necesarios.

## UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso:

Cuenta ▾

Bienvenid@ UNIR ( 0x951d59346352577920DbB5DCA241Fc5c346FE950 )  
600.000.000 ECTS ( 100,00 ETH)

Calcular ECTS

Un vez hemos calculado los tokens ECTS necesarios, el alumno puede acceder a la opción para Comprar ECTS y adquirirlos, informando los tokens que se desea adquirir y la universidad en la que se desea matricular puede Calcular los weis (ETH) necesarios para adquirir estos tokens. Una vez se conoce el precio, el alumno puede proceder a adquirir los tokens para la cual se solicitará una confirmación por su parte.

## UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso:

Cuenta ▾

Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD )  
0 ECTS ( 100,00 ETH)

Calcular ECTS

Comprar ECTS

Matricular

Trasladar

### Comprar Tokens ECTS

Mediante esta funcionalidad puede calcular los weis necesarios para adquirir un determinado número de Tokens en una universidad

Universidad :

UNIR ▾

Tokens ECTS :

18,2

Calcular Weis

Comprar ECTS

Weis Necesarios :

12376000000000000000

### Cálculo de Tokens

**unir** LA UNIVERSIDAD  
EN INTERNET UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

## UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso:

Cuenta ▾

Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD )  
0 ECTS ( 100,00 ETH)

Calcular ECTS Comprar ECTS Matricular Trasladar

### Comprar Tokens ECTS

Mediante esta funcionalidad puede calcular los weis necesarios para adquirir un determinado número de Tokens en una universidad

Universidad : UNIR ▾

Tokens ECTS : 18,2

Weis Necesarios : 12376000000000000000

Calcular Weis Comprar ECTS


Comprar ECTS

¿Está seguro de comprar los créditos indicados? Pulse Aceptar para continuar.

Aceptar Cancelar

### Confirmar compra de Tokens ECTS

En las siguientes imágenes podemos comprobar comparadas con las del inicio de este apartado como ha cambiado el balance de ECTS y de ETH tanto de la UNIR como del alumno tras adquirir los tokens el alumno.


LA UNIVERSIDAD  
EN INTERNET
UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Selecione una cuenta de acceso:

Cuenta ▾


Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD)  
18,2 ECTS ( 98,76 ETH)

Calcular ECTS

Comprar ECTS

Matricular

Trasladar


LA UNIVERSIDAD  
EN INTERNET
UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Selecione una cuenta de acceso:

Cuenta ▾

Bienvenid@ UNIR ( 0x951d59346352577920DbB5DCA241Fc5c346FE950)  
599.999.981,8 ECTS ( 101,24 ETH)

Calcular ECTS

## 4.5. Matricular en Asignatura

Una vez el alumno ha adquirido los tokens necesarios puede acceder a la pestaña Matricular que le permitirá matricularse en una asignatura concreta en una universidad. Para saber los tokens disponibles que tiene en la universidad seleccionada puede hacer uso del botón Balance. Esta opción le informará de los tokens que dispone para su uso en matrículas de la universidad seleccionada.

El curso académico lo establece el sistema por defecto en función de las fechas en las que se matricula el alumno.


LA UNIVERSIDAD  
EN INTERNET
UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Selecione una cuenta de acceso:

Cuenta ▾

Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD)  
18,2 ECTS ( 98,76 ETH)

Calcular ECTS

Comprar ECTS

Matricular

Trasladar

Matricular en Asignatura

Mediante esta funcionalidad puede matricularse en una asignatura en una universidad

Universidad :

UNIR ▾

Tokens ECTS Disponibles :

18,2

Balance

Asignaturas :

Desarrollo de Aplicaciones Blockchain ( DAB ) - Exp: 4 - Créditos: 6 ▾

Curso Académico :

2019 / 2020

Matricular en Asignatura

Tras matricularse podemos observar en la siguiente captura como su balance en ECTS ha disminuido.

**unir** LA UNIVERSIDAD  
EN INTERNET

UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: 

Cuenta ▾

Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD)  
10,4 ECTS ( 98,76 ETH)

Calcular ECTS

Comprar ECTS

Matricular

Trasladar

Nuevamente y tras matricularse en la 2º asignatura ha consumido todos sus tokens ECTS disponibles.

**unir** LA UNIVERSIDAD  
EN INTERNET

UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: 

Cuenta ▾

Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD)  
0 ECTS ( 98,76 ETH)

Calcular ECTS

Comprar ECTS

Matricular

Trasladar

De la misma forma vemos que los tokens han vuelto a pasar a la UNIR para su interacción con otros alumnos.

**unir** LA UNIVERSIDAD  
EN INTERNET

UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: 

Cuenta ▾

Bienvenid@ UNIR ( 0x951d59346352577920DbB5DCA241Fc5c346FE950)  
600.000.000 ECTS ( 101,24 ETH)

Calcular ECTS



Como podemos ver en la siguiente imagen además de registrarse la información on-chain, también se ha recogido en LocalStorage.

Key	Value
estado	"0x5C4756bb912Dea209B94587D4d761aCE5d321054"
universidades	[{"address": "0x951d59346352577920DbB5DCA241Fc5c346FE950"...
profesores	[{"address": "0x7a55FdcB796BA184fDA57530af3303b5553efC56"...
alumnos	[{"address": "0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD"...
entidades-registradas	true
asignaturas	[{"address": "0x36Fb28597264c9e3c7b5Ed382578B53c414932Dd"...
universidades-registradas	true
app-recargada	true
matriculas	[{"universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE...

```

{
  "universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950", ...
}
[
  {
    "universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950", ...
    "universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950"
    "profesor": "0x7a55FdcB796BA184fDA57530af3303b5553efC56"
    "asignatura": "0x36Fb28597264c9e3c7b5Ed382578B53c414932Dd"
    "alumno": "0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD"
    "id": "1"
  },
  {
    "universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950", ...
    "universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950"
    "profesor": "0x7a55FdcB796BA184fDA57530af3303b5553efC56"
    "asignatura": "0xc73faC5CddEB9459c40c905F75754c59A0a725B0"
    "alumno": "0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD"
    "id": "1"
  }
]

```

## 4.6. Evaluar Asignatura no aprobada por parte de un profesor

Simulando el acceso de un profesor a la plataforma vemos como tiene disponible la opción de Evaluar, para ello no tiene más que seleccionar la asignatura el alumno y hacer click en Obtener Token ID para obtener del LocalStorage el id asociado al token que recoge la matrícula del alumno en la asignatura. Se ha optado por este modo de interacción por simplicidad pero lo ideal sería que el alumno acceda a una lista con los alumnos pendientes de evaluar de una asignatura.

Una vez informada la nota, para el caso del ejemplo suspensa “por los pelos” con un 4,95, el sistema registra on-chain la información para sucesivas matrículas del alumno en esta asignatura.

UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso: Cuenta Bienvenid@ Jose Luis ( 0x7a55FdcB796BA184fDA57530af3303b5553efC56 )  
0 ECTS ( 100,00 ETH )

Calcular ECTS Evaluar

**Evaluar Asignatura**  
Mediante esta funcionalidad podrá registrar las notas finales sobre una matrícula de los alumnos matriculados en una asignatura

Asignaturas : Desarrollo de Aplicaciones Blockchain ( DAB ) - Exp: 4 - Créditos: 6

Alumno : Javier Montesinos Morcillo Obtener Token ID

Token Id : 1

Nota : 4,95 (0,00 - 10,00) Evaluar Asignatura

Adicionalmente se registra la información en LocalStorage.

Key	Value
universidades-registradas	true
matriculas	[{"universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE..."}]
app-recargada	true
entidades-registradas	true
matriculas-evaluadas	[{"universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE...", "universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950", "profesor": "0x7a55FdcB796BA184fDA57530af3303b5553efC56", "asignatura": "0x36Fb28597264c9e3c7b5Ed382578B53c414932Dd", "alumno": "0x951d59346352577920DbB5DCA241Fc5c346FE950", "id": "1", "nota": 495}]]
alumnos	[{"address": "0x951d59346352577920DbB5DCA241Fc5c346FE950"...}]
universidades	[{"address": "0x951d59346352577920DbB5DCA241Fc5c346FE950"...}]
asignaturas	[{"address": "0x36Fb28597264c9e3c7b5Ed382578B53c414932Dd", "...}]
estado	"0x5C4756bb912Dea209B94587D4d761aCE5d321054"
profesores	[{"address": "0x7a55FdcB796BA184fDA57530af3303b5553efC56", "...}]

## 4.7. Evaluar Asignatura aprobada por parte de un profesor

En el caso de evaluar una asignatura con una nota aprobada esta se registra del mismo modo pero on-chain el token pasa a ser titularidad del alumno de forma que lo puede utilizar para realizar traslados de asignatura en el apartado se verificará como la asignatura no está en posesión del alumno mientras que cuando está aprobada si.

LA UNIVERSIDAD  
EN INTERNET

UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso:

Cuenta ▾

Bienvenid@ Jose Luis ( 0x7a55FdcB796BA184fDA57530af3303b5553efC56)

0 ECTS ( 100,00 ETH)

Calcular ECTS

Evaluar

Evaluar Asignatura

Mediante esta funcionalidad podrá registrar las notas finales sobre una matriculo de los alumnos matriculados en una asignatura

Asignaturas :

Trabajo Final de Experto ( TFE ) - Exp: 4 - Créditos: 8 ▾

Alumno :

Javier Montesinos Morcillo ▾

Obtener Token ID

Token Id :

1

Nota :

10

(0,00 - 10,00)

Evaluar Asignatura

Al igual que en el caso de la asignatura no aprobada esta información se registra también en LocalStorage.

Key	Value
universidades-registradas	true
matriculas	[]
app-recargada	true
entidades-registradas	true
matriculas-evaluadas	[{"universidad": "0x951d59346352577920DbB5DCA241Fc5c346FE950", ...}]
alumnos	[{"address": "0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD", ...}]
universidades	[{"address": "0x951d59346352577920DbB5DCA241Fc5c346FE950", ...}]
asignaturas	[{"address": "0x36Fb28597264c9e3c7b5Ed382578B53c414932Dd", ...}]
estado	"0x5C4756bb912Dea209B94587D4d761aCE5d321054"
profesores	[{"address": "0x7a55FdcB796BA184fDA57530af3303b5553efC56", ...}]

## 4.8. Trasladar matrícula a otra Universidad

Por último en el caso de que un alumno desee trasladar una asignatura a otra universidad debe tener aprobada esta. Como podemos ver en la captura siguiente al obtener el token Id de la asignatura de Desarrollo... suspensa por parte del alumno se retorna cero.

**unir** LA UNIVERSIDAD  
EN INTERNET

UNIR :: Experto Universitario en Desarrollo Blockchain (2019 / 20) :: TFE :: J2M

UNIR :: TFE - Títulos Académicos Tokenizados

Seleccione una cuenta de acceso:
 

Cuenta ▾

Bienvenid@ Javier ( 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD )  
 0 ECTS ( 98,75 ETH )

Calcular ECTS
 Comprar ECTS
 Matricular
 **Trasladar**

Trasladar Matrícula de Asignatura

Mediante esta funcionalidad puede solicitar el traslado de una asignatura aprobada a otra universidad.

Asignatura :
 

Desarrollo de Aplicaciones Blockchain ( DAB ) - Exp: 4 - Créditos: 6 ▾

Token Id :
 

0

Obtener Token ID

Universidad Destino :
 

UNIR ▾

Trasladar Asignatura

Sin embargo al recuperar el token id de la asignatura aprobada retorna su valor correcto para permitir el traslado.



**unir** LA UNIVERSIDAD  
EN INTERNET

UNIR :: TFE - Título

Seleccione una cuenta de acceso

[Calcular ECTS](#) [Comprar](#)

**Trasladar Matrícula de Asignatura**  
Mediante esta funcionalidad puede solicitar el traslado de una asignatura aprobada a otra universidad.

Asignatura : Trabajo Final de Experto ( TFE ) - Exp: 4 - Créditos: 8

Token Id : 1 [Obtener Token ID](#)

Universidad Destino : UNIR

[Trasladar Asignatura](#)

**Trasladar Asignatura** ✕

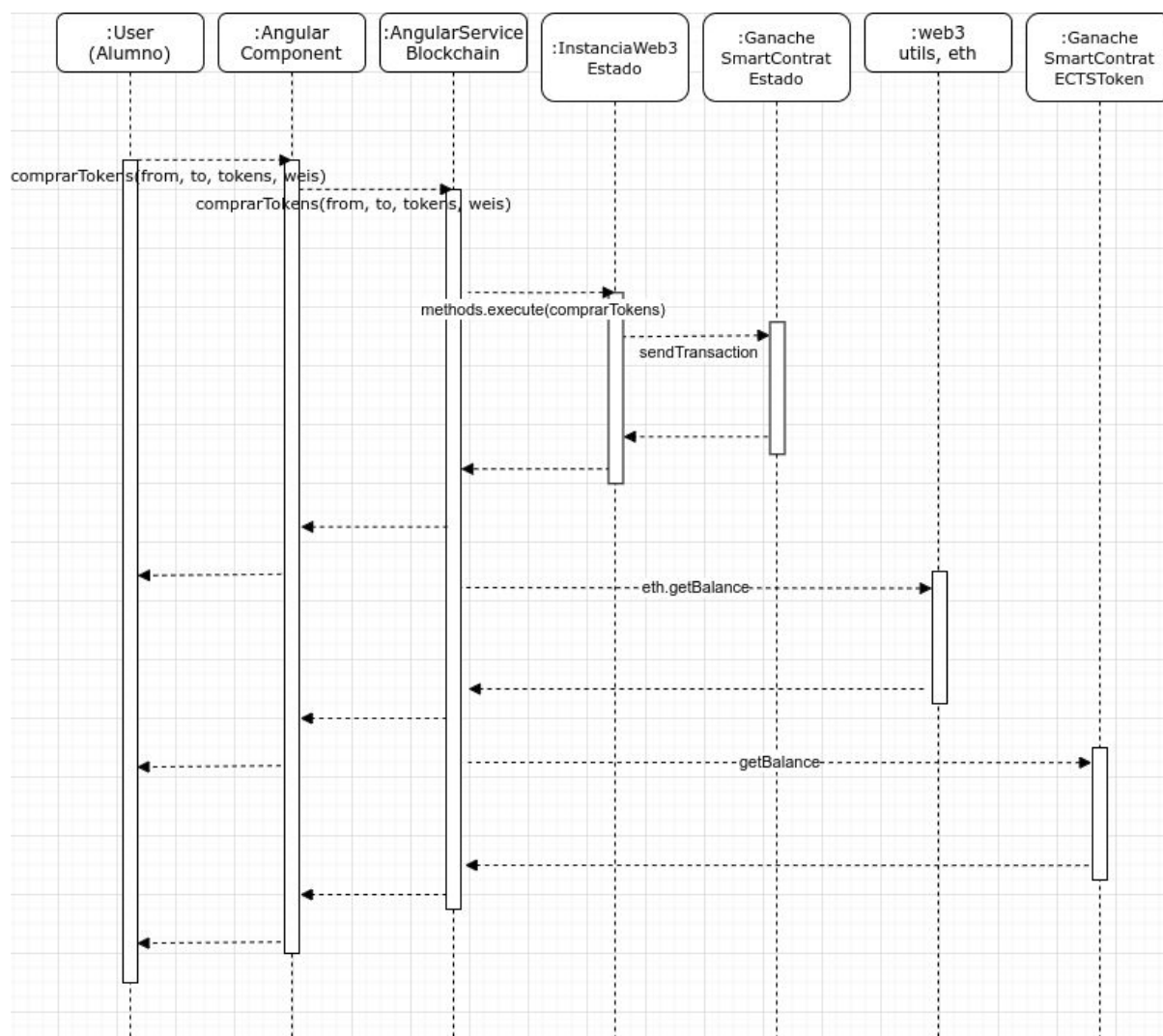
¿Está seguro de trasladar la matrícula de la asignatura seleccionada? Pulse Aceptar para continuar.

[Aceptar](#) [Cancelar](#)

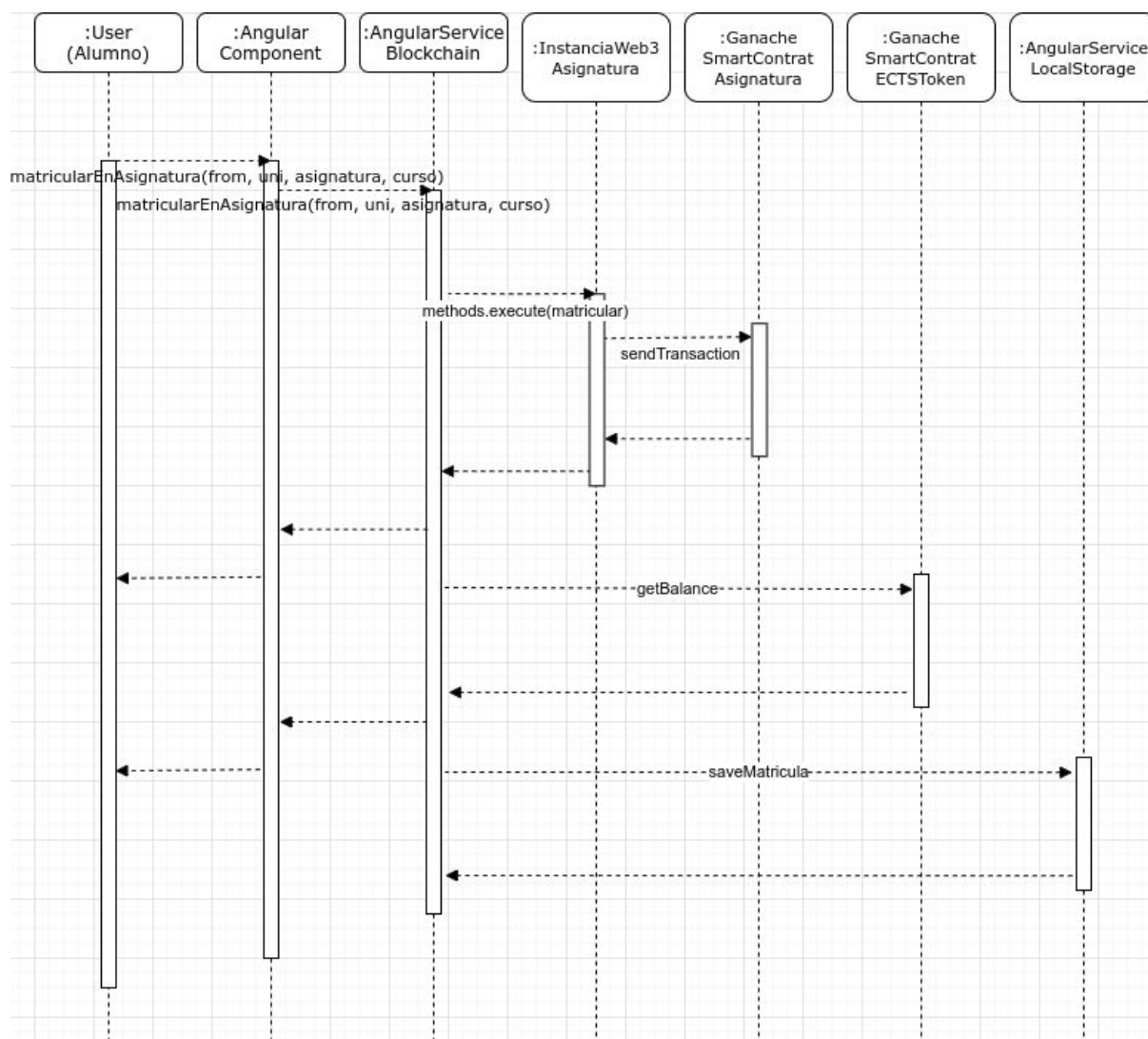
Blockchain (2019 / 20) :: TFE :: J2M  
4DEe1f12F7976251D2753DfD)  
0 ECTS ( 98,75 ETH)

## 4.9. Diagramas de secuencia

Este apartado recoge dos diagramas UML de secuencia de aquellos casos de uso más representativos del desarrollo realizado; **Comprar Tokens** por parte de un Alumno y **Matricularse en una Asignatura** por parte del alumno.



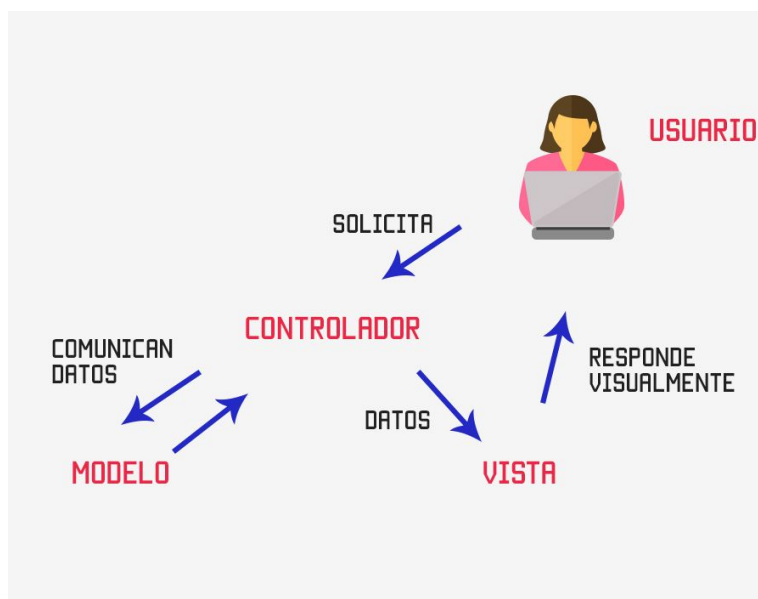
*Caso de uso - Comprar Tokens ECTS - Diagrama de secuencia*



*Caso de uso - Matricular en AsignaturaToken - Diagrama de secuencia*

## 5. Arquitectura / Modelo del sistema

En este desarrollo se ha aplicado el patrón de diseño de arquitectura MVC de forma que puedan quedar claramente diferenciadas e identificadas las capas de la aplicación. En la imagen siguiente se puede observar un esquema sencillo pero claro de lo contemplado por el patrón MVC.



### 5.1. Controlador

Mediante este patrón el usuario interactúa con los diferentes controladores de la aplicación identificados por los componentes de Angular siguientes;

- el principal; app.component.ts
- calcular-ects.component.ts
- comprar-ects.component.ts
- matricular-asignatura.component.ts
- evaluar-asignatura.component.ts o
- trasladar-asignatura.component.ts

### 5.2. Modelo

El controlador interactúa con el modelo para manejar los datos que pueda enviar el usuario y preparar la vista que visualizará este. En este desarrollo el modelo está representado por 2 clases de servicio en el proyecto más el despliegue de los smart contracts en la red blockchain:

- SmartContracts desplegados en la red blockchain:
  - Estado.sol - Representa al estado y todas las operaciones que este puede realizar así como aquellas que puedan realizar otros roles sobre el.

- ECTSToken.sol - Representa el token ERC20 que permite el intercambio entre universidades y alumnos de tokens ERC20 (ECTSToken) por tokens ERC721 (AsignaturaToken)
- AsignaturaToken.sol - Representa cada asignatura disponible en el sistema, de forma que cada token recoge la matrícula de un alumno en la asignatura.
- Servicios en la app Angular:
  - localStorage.service.ts. Recoge todas las operaciones necesarias para persistir y recuperar los datos de la BD seleccionada para el desarrollo, LocalStorage.
  - blockchain.service.ts. Recoge todas las operaciones encargadas de interactuar con los sc desplegados en la red blockchain. Utiliza los siguientes servicios:
    - el servicio localStorage.service.ts para registrar en BD la información necesario.
    - los smartcontracts desplegados en la red blockchain.

### 5.3. Vista

Una vez que el controlador obtiene respuesta del modelo utiliza estos para componer la vista que mostrará al usuario para que este pueda interactuar con la aplicación. Estas vistas estarían contempladas en los siguientes ficheros HTML:

- la principal; app.component.html
- calcular-ects.component.html
- comprar-ects.component.html
- matricular-asignatura.component.html
- evaluar-asignatura.component.html o
- trasladar-asignatura.component.html

## 6. Despliegue

A continuación se recoge como desplegar e inicializar la aplicación para su uso.

### 6.1. Desarrollo (Ganache)

Para desplegar nuestra aplicación sobre Ganache necesitaremos 3 sencillos pasos:

- Iniciar una red blockchain en local utilizando ganache-cli
- Desplegar los smart contracts necesarios; Estado.sol y ECTSToken.sol así como la vinculación existente entre ellos.
- Iniciar el servicio de la aplicación web.

A continuación se recoge de forma detallada cómo ejecutar cada uno de estos pasos.

#### 6.1.1. Instalar ganache-cli

Para instalar ganache-cli no tenemos más que ejecutar el siguiente comando desde una ventana de terminal:

```
$ npm install -g ganache-cli
```

#### 6.1.2. Lanzar ganache-cli

Para lanzar ganache-cli usaremos un **mnemonic** concreto que nos garantizará que las direcciones generadas serán las necesarias según la configuración de la aplicación. Ganache se deberá lanzar en el puerto **8545** que es el puerto por defecto, no obstante se indica de forma concreta en el comando de inicio.

Lanzaremos el servicio con el siguiente comando desde una ventana de terminal:

```
$ ganache-cli -m="belt allow snack gain mom rug wave inflict risk verb  
health notable" -p=8545
```

Una vez lanzado deberemos visualizar algo similar a lo que se muestra en la siguiente imagen, con las cuentas necesarias para el proyecto.

```

root@alastria-ubuntu16: ~
root@alastria-ubuntu16:~# ganache-cli -m="belt allow snack gain mom rug wave inflict risk verb health
notable" -p=8545
Ganache CLI v6.9.1 (ganache-core: 2.10.2)

Available Accounts
=====
(0) 0x5C4756bb912Dea209B94587D4d761aCE5d321054 (100 ETH)
(1) 0x951d59346352577920DbB5DCA241Fc5c346FE950 (100 ETH)
(2) 0x7a55FdcB796BA184fDA57530af3303b5553efC56 (100 ETH)
(3) 0xFc83780Dd8bc6eAE4DEe1f12F7976251D2753DfD (100 ETH)
(4) 0xE081280929D611D12EA7EcC7Cb700282755531BC (100 ETH)
(5) 0xd671FA06396fC7a0337ae37a77c2767ae97598Ba (100 ETH)
(6) 0x4f3832424D8Fb9e290748a455B5C59362cA0aa56 (100 ETH)
(7) 0x984f0e1459db3DD040569c392BA1797811E17C3C (100 ETH)
(8) 0x3b9612f508ebA848c58b3A973B1D46cc224f657b (100 ETH)
(9) 0x81E28ACe8C71E8572e94bA0faa27B8Bf2a59774E (100 ETH)

Private Keys
=====
(0) 0x38280a67e6508b3a6d2b125210a3b8a24ce96f79a5b3d23db57f70fe044fc65e
(1) 0xfc5f55bfccce118eb4a784c4691d4d8b93b97cb1260e44b6e9334b16ada45d122
(2) 0x32b0b7ea372379df09600ca155f87720ecbfb068d777be68b513b54855b07484
(3) 0xdab3128966978cd96ab86a8ad283cd808b276229b9e9c3fddf5474d93095cbf9e
(4) 0x18ead302ddc6ec81ea18af21f9653bd65d5a4d2c78e87aac428ac4fcee8dcba0a
(5) 0x90db21653f0adb90ccc0e8c08ef8b7405e836a0fb74bc3981eb8bb50a9264e7
(6) 0x7b1cc248f92f6bd92fbc8e1543432c855cd189154aca8f17ca61e68882a84e62
(7) 0x0af01f93834495a9fdd783bc5ac09501fc27eca446fd5e89d9cd69ae85198ebd
(8) 0x2abde3585f7c94e555d92fb42068a2131ca3b699eae327ab05e131f29d82d627
(9) 0xd187c1f22221e14ff77f6fea1d686834a2203835925690d0a03ab462197ab8d

HD Wallet
=====
Mnemonic: belt allow snack gain mom rug wave inflict risk verb health notable
Base HD Path: m/44'/60'/0'/0/{account_index}

Gas Price
=====
20000000000

Gas Limit
=====
6721975

Call Gas Limit
=====
9007199254740991

```

#### 6.1.4. Desplegar los Smart Contracts en la red

Para desplegar los contratos en la red Ganache desde la carpeta raíz del código del TFE asociado al backend blockchain debemos instalar primeramente los paquetes npm necesarios, para ello no tenemos más que ejecutar el siguiente comando:

```

$ cd code/unir-tfe-open-zeppelin
$ npm install

```

Para poder ejecutar tanto el despliegue de los smart contracts del proyecto como de los tests de pruebas unitarias necesitamos instalar Truffle, debido a la compatibilidad verificada en el curso con la testnet de alastria y open zeppelin se debe instalar la versión 5.0.18 de truffle, para ello no tenemos más que ejecutar el siguiente comando:

```

$ npm install -g truffle@5.0.18

```



Una vez instalados los paquetes npm del proyecto, Truffle e iniciada la red Ganache podemos desplegar los smart contracts necesarios ejecutando el siguiente comando de Truffle:

```
$ truffle migrate --network development --reset
```

La salida de este comando nos debe informar del despliegue correcto de 3 smart contracts: Migrations, ECTSToken y Estado tal como se puede comprobar en la imagen siguiente:

### Compilación de los smart contracts

```
Compiling your contracts...
=====
> Compiling ./contracts/Estado.sol
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/entidades/Alumnos.sol
> Compiling ./contracts/entidades/Profesores.sol
> Compiling ./contracts/entidades/Universidades.sol
> Compiling ./contracts/token/ERC20/ECTSERC20.sol
> Compiling ./contracts/token/ERC20/ECTSToken.sol
> Compiling ./contracts/token/ERC721/AsignaturaToken.sol
> Compiling ./contracts/token/ERC721/ERC721.sol
> Compiling ./contracts/token/ERC721/ERC721Metadata.sol
> Compiling @openzeppelin/contracts/drafts/Counters.sol
> Compiling @openzeppelin/contracts/introspection/ERC165.sol
> Compiling @openzeppelin/contracts/introspection/IERC165.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/ownership/Ownable.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20Detailed.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Artifacts written to /home/javier/UNIR-Blockchain/05-TFE/unir-tfe-open-zeppelin/build/contracts
> Compiled successfully using:
   - solc: 0.5.16+commit.9c3226ce.Emscripten.clang
```

...

Despliegue del smart contract ECTSToken y del sc Estado.

```
Deploying 'ECTSToken'
-----
> transaction hash: 0x3032496a591bf235a2f4087a478639ca519d35d04b8612994509d1106478fb96
> Blocks: 0
> contract address: 0x4cAB8f9DFAefea8eb69261560a024cf9B3f0b69E
> block number: 3
> block timestamp: 1584286374
> account: 0x5C4756bb912Dea209B94587D4d761aCE5d321054
> balance: 99.96737098
> gas used: 1460215
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.0292043 ETH

Deploying 'Estado'
-----
> transaction hash: 0x71be82d28f51701cf3ba39c4709d72b1f4b7d61590db11fbac3973f3c830a498
> Blocks: 0
> contract address: 0xb482955f78b3a86a1d7275a7fa5bA122250424FF
> block number: 4
> block timestamp: 1584286374
> account: 0x5C4756bb912Dea209B94587D4d761aCE5d321054
> balance: 99.8780812
> gas used: 4464489
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.08928978 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.11849408 ETH

Summary
=====
> Total deployments: 3
> Final cost: 0.12107348 ETH
```



### 6.1.5. Iniciar la aplicación web

Para iniciar la aplicación web tendremos dos posibilidades diferentes.

Si tenemos instalado `angular-cli` podremos lanzar la aplicación web mediante `ng serve` haciendo un build del código facilitado de la aplicación en la carpeta “**unir-tfe-dapp/code**” facilitada en el entregable. Si por el contrario deseamos simplificar el inicio de la app basta con tener instalado `python` y utilizar **SimpleHTTPServer** para servir el contenido desde la carpeta “**unir-tfe-dapp/dist**” facilitada en el entregable.

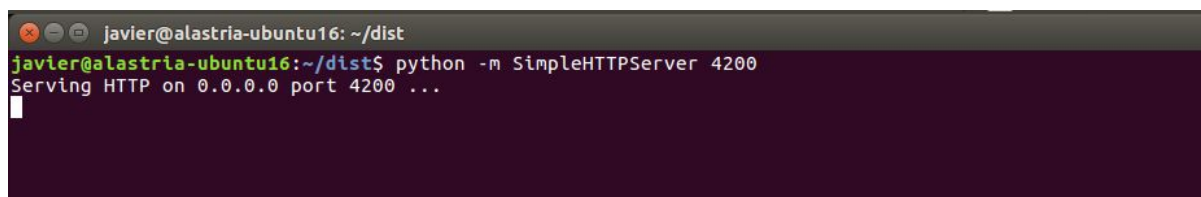
#### 6.1.5.1. SimpleHTTPServer

Usando **SimpleHTTPServer** de `python` iniciaremos el servicio de la aplicación sobre el build entregado con el código del proyecto y ubicado en la carpeta “**unir-tfe-dapp/dist**”.

Para ello no tendremos más que ejecutar en otra ventana de terminal el siguiente comando:

```
$ cd unir-tfe-dapp/dist/  
$ python -m SimpleHTTPServer 4200
```

Deberemos visualizar lo siguiente en nuestro terminal.

A screenshot of a terminal window with a dark background. The prompt is 'javier@alastria-ubuntu16: ~/dist'. The user has entered the command 'python -m SimpleHTTPServer 4200'. The output shows 'Serving HTTP on 0.0.0.0 port 4200 ...' followed by a cursor on a new line.

```
javier@alastria-ubuntu16: ~/dist  
javier@alastria-ubuntu16:~/dist$ python -m SimpleHTTPServer 4200  
Serving HTTP on 0.0.0.0 port 4200 ...  
█
```

#### 6.1.5.2. Angular CLI

La segunda opción para iniciar la aplicación web es hacer un build del código facilitado en la carpeta “**unir-tfe-dapp/code**” del entregable mediante el cliente de angular (**angular-cli**) Para ello no tendremos más que ejecutar los siguientes comandos en una ventana de terminal:

Primeramente instalar los paquetes `npm` necesarios de la aplicación:

```
$ cd unir-tfe-dapp/code/  
$ npm install
```

Por último tan solo nos queda iniciar la aplicación mediante el comando `ng serve`. El parámetro `-o` nos abrirá automáticamente una ventana del navegador.

```
$ ng serve -o
```

“La ventaja de lanzar la aplicación web mediante `ng serve` es que si hacemos cambios en el código estos se refrescarán en la aplicación automáticamente”

### 6.1.5.3. Versiones Angular CLI

```

Angular CLI
Angular CLI: 8.3.24
Node: 13.2.0
OS: linux x64
Angular: 8.2.14
... animations, common, compiler, compiler-cli, core, forms
... language-service, platform-browser, platform-browser-dynamic
... router

Package                                  Version
-----
@angular-devkit/architect                0.803.24
@angular-devkit/build-angular            0.803.24
@angular-devkit/build-optimizer          0.803.24
@angular-devkit/build-webpack            0.803.24
@angular-devkit/core                     8.3.24
@angular-devkit/schematics               8.3.24
@angular/cli                             8.3.24
@ngtools/webpack                         8.3.24
@schematics/angular                     8.3.24
@schematics/update                       0.803.24
rxjs                                     6.4.0
typescript                               3.5.3
webpack                                  4.39.2

```

Las versiones utilizadas para el entorno de desarrollo basadas en **angular-cli** son las que se pueden visualizar en la imagen de la izquierda. *Requisitos previos para instalar Angular CLI es disponer en nuestro equipo de Node.js y de npm package manager.*

Una vez dispongamos de estas dos herramientas instalar Angular CLI es tan sencillo como ejecutar en una ventana de terminal el siguiente comando:

```
$ npm install -g @angular/cli
```

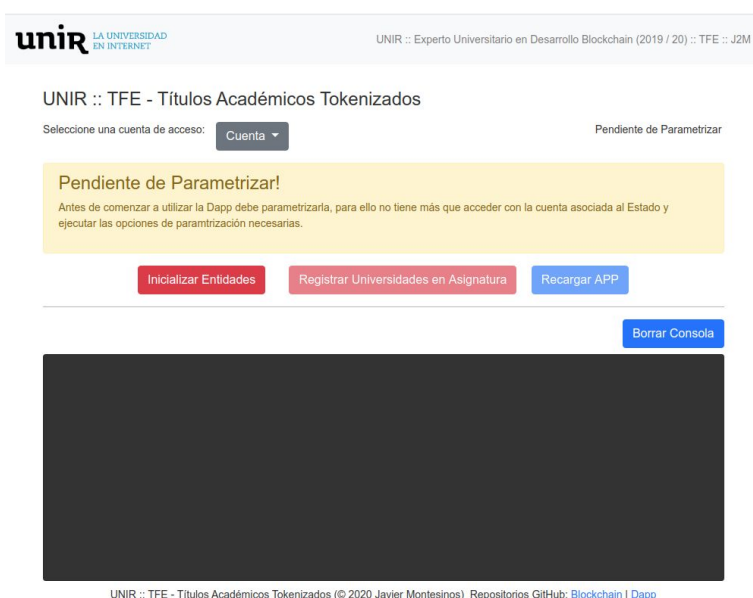
En el siguiente [recurso](#) se muestra de forma detallada cómo instalar

Angular CLI en nuestro equipo de trabajo.

### 6.1.6. Acceder a la aplicación web

Una vez iniciada la aplicación web ya sea utilizando **SimpleHTTPServer** o bien usando **ng serve** de **Angular CLI** podremos acceder a nuestra aplicación web desde cualquier navegador web accediendo a la siguiente url: <http://localhost:4200/>

Deberemos ver el contenido que se muestra en la imagen siguiente.



## 6.2. Preproducción (Testnet) / Producción (Alastria Nodo UNIR)

El despliegue sobre la versión Testnet de Alastria que se ha visto durante el curso ha generado diversas dudas y errores. Tal como se ha comentado por parte de uno de los tutores de los TFE las versiones de Open Zeppelin y Truffle que se debían utilizar para su correcto despliegue en esta red eran la versión 2.3 y la 5.0.18 respectivamente.

Como se puede observar en la captura siguiente del log de git del proyecto unir-tfe-open-zeppelin, dadas las implementaciones particulares de ERC20 y ERC721 contempladas en el TFE se tuvo que abrir una nueva rama en git para implementar estas clases procedentes de la versión 2.3 de Open Zeppelin, dado que la inicial instalada y sobre la que funcionaba correctamente el proyecto era la 2.5. Finalmente se ha entregado el desarrollo sobre la versión 2.3.

```
*   commit 436f60582b29ee75199b3f12303e290daac1615c (HEAD -> master, origin/master)
Merge: ba2efc1 e2c34f6
Author: Javier Montesinos <fj.montesinos@gmail.com>
Date:   Sun Mar 15 10:11:28 2020 +0100

    Merge branch 'open-zeppelin-2.3'

*   commit e2c34f6b0070cc82fe6bf4d963ae3cf90028711f (origin/open-zeppelin-2.3)
Author: Javier Montesinos <fj.montesinos@gmail.com>
Date:   Sun Mar 15 01:26:09 2020 +0100

    Updated. Añadida la asignatura como valor al evento de matricular a un alumno en una asignatura

*   commit 7c12104c563c35b48de6094b71768d2622d8f6f8
Author: Javier Montesinos <fj.montesinos@gmail.com>
Date:   Sun Mar 15 01:24:46 2020 +0100

    Added. Metodo para que un alumno pueda obtener los tokens ERC20 que tiene para una universidad

*   commit b6c496678869642e3bf93f4ed4cc4305de689656
Author: Javier Montesinos <fj.montesinos@gmail.com>
Date:   Sat Mar 14 12:46:02 2020 +0100
```

Para lanzar la Testnet necesitamos ejecutar el siguiente comando en la carpeta test-environment/infrastructure/testnet de nuestra instalación de Tesnet:

```
$ sudo ./bin/start_network.sh clean 1 2
```

A diferencia de lo que ocurre con Ganache en el caso de la Testnet debemos desbloquear la cuenta que incorpora por defecto para poder ejecutar la migración. Para ello nos conectamos a la red utilizando el cliente geth mediante el siguiente comando:

```
$ sudo geth attach ipc:network/general11/geth.ipc
```

Una vez estamos conectados podemos desbloquear la cuenta 0x74d4c56d8dcbb10a567341bfac6da0a8f04dc41d mediante el siguiente comando:

```
[sudo] password for javier:
Welcome to the Geth JavaScript console!

instance: Geth/general1/v1.7.2-stable-94e1e31e/linux-amd64/go1.9.5
coinbase: 0xfc1c5a534591d2da95c2dc191e7a6a290892dbb9
at block: 32 (Wed, 11 Mar 2020 19:54:45 CET)
datadir: /home/javier/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.personal.unlockAccount(web3.eth.accounts[0], 'Passw0rd', 0)
```

Llegados a este punto se ha observado un problema en comparación con lo testeado sobre Ganache en relación al consumo de gas teniendo que aumentar el gas configurado en el fichero truffle-config.js de 0xfffff a 0xffffffff para su correcto despliegue.

Una vez realizado este cambio ya podemos desplegar los smart contracts sobre Tesnet, para ello no tenemos más que ejecutar el comando de despliegue visto anteriormente en el capítulo dedicado a Ganache.

```
$ truffle migrate --network development --reset
```

La salida es similar a la obtenida en el despliegue sobre Ganache pero con un consumo mucho mayor como se puede observar en la siguiente imagen sobre todo en el Smart Contract Estado.

```
2_migration_ects.js
=====

Replacing 'ECTSToken'
-----
> transaction hash: 0xe391e7d95bfa73c7d6eac679bd3c53488495db2b2381c3d12fb4992c48883dae
> Blocks: 0 Seconds: 0
> contract address: 0x44B04Fc0aBfaDAff12269d09CA8611234Ed1ae3F
> block number: 13425
> block timestamp: 1583869028
> account: 0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
> balance: 0
> gas used: 1832022
> gas price: 0 gwei
> value sent: 0 ETH
> total cost: 0 ETH

Replacing 'Estado'
-----
> transaction hash: 0xa35aa5e939b8a0d976beb586a2d7bd85962286366d5f3faf8d39304d643f6380
> Blocks: 0 Seconds: 0
> contract address: 0x4A7C8b3202DC195c6fbBE9a3ef84Cb7C9A8D6d9D
> block number: 13428
> block timestamp: 1583869031
> account: 0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
> balance: 0
> gas used: 6013642
> gas price: 0 gwei
> value sent: 0 ETH
> total cost: 0 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0 ETH
```

Al ejecutar las pruebas de los test se identifica un problema derivado de la llamada entre smart contracts, el desarrollo utiliza esta característica para delegar en otros smart contracts determinadas funcionalidades.



Para descartar que el problema no esté derivado de un tamaño excesivo de los smart contracts desarrollados, del uso de open zeppelin, de las implementaciones realizadas o de complejos cálculos que se pudieran realizar en algunos métodos, se elabora un proyecto en truffle (disponible en github en [unir-sc2sc-testnet](#)) muy sencillo, con dos smart contracts A y B donde uno realiza una sencilla llamada sobre el segundo para obtener un valor, algo muy simple. La ejecución y despliegue sobre Ganache es correcta, sin embargo sobre Testnet se despliega de forma correcta pero en el momento de llamar al método que interactúa con el segundo smart contract se obtiene el error que se puede ver en la siguiente imagen.

```
javier@alastria-ubuntu16:~/sc2sc-alastria$ truffle console --network=alastrialocal
truffle(alastrialocal)> let b = await B.at("0x7684fB450dE19C57e164d588Fe8Ff399ba82E16D")
undefined
truffle(alastrialocal)> (await b.getValorDeA()).toString()
Thrown:
Error: Returned values aren't valid, did it run Out of Gas?
    at XMLHttpRequest._onHttpResponseBodyEnd (/usr/local/lib/node_modules/truffle/build/webpack:/~/xhr2-co
okies/dist/xml-http-request.js:318:1)
    at XMLHttpRequest._setReadyState (/usr/local/lib/node_modules/truffle/build/webpack:/~/xhr2-cooki
es/dist/xml-http-request.js:208:1)
    at XMLHttpRequestEventTarget.dispatchEvent (/usr/local/lib/node_modules/truffle/build/webpack:/~/
xhr2-cookies/dist/xml-http-request-event-target.js:34:1)
    at XMLHttpRequest.request.onreadystatechange (/usr/local/lib/node_modules/truffle/build/webpack:/
~/web3/~web3-providers-http/src/index.js:96:1)
    at /usr/local/lib/node_modules/truffle/build/webpack:/packages/truffle-provider/wrapper.js:112:1
    at /usr/local/lib/node_modules/truffle/build/webpack:/~/web3-eth-contract/~web3-core-requestmana
ger/src/index.js:147:1
    at sendTxCallback (/usr/local/lib/node_modules/truffle/build/webpack:/~/web3-eth-contract/~web3-
core-method/src/index.js:473:1)
    at Method.formatOutput (/usr/local/lib/node_modules/truffle/build/webpack:/~/web3-eth-contract/~
web3-core-method/src/index.js:163:1)
    at Method.outputFormatter (/usr/local/lib/node_modules/truffle/build/webpack:/~/web3-eth-contract
/src/index.js:818:1)
    at Contract._decodeMethodReturn (/usr/local/lib/node_modules/truffle/build/webpack:/~/web3-eth-co
ntract/src/index.js:465:1)
    at ABICoder.decodeParameters (/usr/local/lib/node_modules/truffle/build/webpack:/~/web3-eth-abi/s
rc/index.js:226:1)
truffle(alastrialocal)> (await b.getValorDeB()).toString()
'2'
truffle(alastrialocal)> █
```

Al tratar este tema con el tutor en el TFE, Jose Luis Nieto, este me informa que en la versión de la Testnet utilizada y con la que se han realizado las prácticas de despliegue no se contempla la llamada entre sc y que nos aseguremos de que funciona todo correctamente como mínimo sobre Ganache de forma que argumentemos sobre la memoria el motivo.

## 7. Testing

Como parte del ciclo de vida de cualquier desarrollo en este TFE se deben contemplar pruebas unitarias principalmente sobre el código Solidity. Se han elaborado las pruebas unitarias que garantizan la correcta funcionalidad desarrollada utilizando Truffle.

Estas pruebas se han dividido en 3 bloques claramente diferenciados relacionados con:

1. Registro de entidades on-chain por parte del estado (fichero test/1\_test\_registros.js)
2. Operaciones con el estandar ERC20 que cubran las necesidades para adquirir tokens ECTS (fichero test/2\_test\_erc20\_tokens.js)
3. Operaciones con el estandar ERC721 que cubran las necesidades para interactuar con las asignaturas como son matrícula, evaluación o traslado (fichero test/3\_test\_erc721\_tokens.js)

Para ejecutar los test no tenemos más que ejecutar desde el directorio raíz de la carpeta unir-tfe-open-zeppelin el siguiente comando una vez hemos instalado los paquetes npm necesarios, lanzado Ganache y desplegado los smart contracts sobre ella:

```
$ truffle test --network development
```

La salida de este comando debe ser similar a la que se muestra en la siguiente imagen en la que como se puede observar se pasan todos los test contemplados:

```
Using network 'development'.

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

Contract: Registro de Entidades
  ✓ estado::configuración correcta
  ✓ estado::registrar universidad (83ms)
  ✓ estado::registrar profesor (62ms)
  ✓ estado::registrar alumno (63ms)

Contract: Tokens ERC20
  ✓ estado::configuración correcta
  ✓ alumno::calcular weis x créditos (92ms)
  ✓ alumno::comprar tokens ects (242ms)

Contract: Tokens ERC721
  ✓ estado::configuración correcta (38ms)
  ✓ estado::crear asignatura ERC721 (155ms)
  ✓ estado::registrar universidad - profesor en asignatura (227ms)
  ✓ alumno::matricular en asignatura (439ms)
  ✓ profesor::evaluar nota final de asignatura::suspense (370ms)
  ✓ profesor::evaluar nota final de asignatura::aprobado (373ms)
  ✓ alumno::solicitar traslado de asignatura:aprobado (501ms)
  ✓ alumno::solicitar traslado de asignatura:no aprobado (462ms)

15 passing (5s)
```

## 8. Conclusiones

Tras completar este TFE y llegado el final de este Experto Universitario me gustaría aportar una serie de conclusiones desde mi punto de vista relacionadas con la tecnología Blockchain y con todo lo que gira entorno a ella.

Si bien es una tecnología novedosa y que tras realizar el Experto se reafirma mi convencimiento de que está llamada a revolucionar determinados procesos y mercados tal como los conocemos a día de hoy, hay determinados aspectos que la hacen todavía “inmadura y/o cambiante” de una forma vertiginosa.

El caso de la testnet utilizada para el despliegue de este TFE, es un ejemplo en el que algo a priori básico y contemplado en las clases de programación a lo largo del Experto, como son las llamadas entre contratos, no se contempla en esta versión y esto ha hecho que no se pueda probar sobre ella su desarrollo. Este carácter evolutivo tan rápido hace que pueda ser complejo identificar las versiones necesarias para interconectar funcionalidades de los diferentes componentes necesarios. Algo parecido nos ocurre con las versiones de web3.js que en mi caso y dado que inicié el proyecto con Angular 8 tuve que verificar que versión 1.0 de era compatible y adaptar el desarrollo a lo que esta versión permite.

Por otro lado es importante destacar que tras realizar el experto me queda muy claro que no todos los proyectos deben involucrar el uso de tecnología Blockchain pero que hay casos de uso muy concretos donde su aplicación es de libro y agilizará operaciones y procesos como puedan ser procesos de trazabilidad, certificación o identidad digital.

Deberemos estar atentos a los frameworks que van surgiendo para verificar aquellos que se estabilizan en el mercado y son mejor acogidos por la comunidad mundial, analizando las posibilidades de Open Zeppelin recalé en la documentación del proyecto 0xcert que me pareció muy interesante desde el punto de vista que simplifica la operativa subyacente para realizar determinadas implementaciones de certificación por ejemplo.

Al incorporarme a este Experto había leído algunos libros sobre Blockchain y realizado algún curso menor en desarrollo con Solidity, las dudas que me generaron me hicieron embarcarme en este Experto, una de ellas era la Experiencia de los Usuarios al interactuar con las aplicaciones (Dapp) basadas en blockchain. Después de realizar el curso sigo con dudas en este sentido que he podido comprobar son dudas de la comunidad como se recoge en el artículo: [The Challenge of UX in Ethereum](#) o en el capítulo 7 User Onboarding del libro [Ethereum for Web Developers: Learn to Build Web Applications on top of the Ethereum Blockchain](#)

Por último, creo que tanto la realización de este TFE como del Experto ha ampliado de forma considerable mis conocimientos sobre la tecnología Blockchain para abordar proyectos relacionados y entender que nos puede ofrecer y que no cada una de las redes disponibles en la actualidad. No obstante deberemos estar muy atentos a su evolución.

## 9. Bibliografía

A continuación se recoge el listado de la documentación y recursos consultados para elaborar el TFE:

[1]	Documentación Open Zeppelin sobre Smart Contrats <a href="https://docs.openzeppelin.com/contracts/2.x/">https://docs.openzeppelin.com/contracts/2.x/</a>
[2]	0xcert API <a href="https://docs.0xcert.org/api/">https://docs.0xcert.org/api/</a>
[3]	0xcert Framework <a href="https://docs.0xcert.org/framework/v2/">https://docs.0xcert.org/framework/v2/</a>
[4]	Build a smart contract that transfers ERC20 token from your wallet to other ERC20 compliant wallet- Part 1 <a href="https://medium.com/coinmonks/build-a-smart-contract-that-transfers-erc20-token-from-your-wallet-to-other-addresses-or-erc20-ee8dc35f40f6">https://medium.com/coinmonks/build-a-smart-contract-that-transfers-erc20-token-from-your-wallet-to-other-addresses-or-erc20-ee8dc35f40f6</a>
[5]	Building Ethereum Dapps With ReactJS + Truffle Contract + Web3, A UI For TokenZendr A Smart Contract That Transfers ERC20 Tokens To Other Addresses — Part 2 <a href="https://medium.com/coinmonks/building-ethereum-dapps-with-reactjs-truffle-contract-web3-a-ui-for-tokenzandr-a-smart-bf345478b116">https://medium.com/coinmonks/building-ethereum-dapps-with-reactjs-truffle-contract-web3-a-ui-for-tokenzandr-a-smart-bf345478b116</a>
[6]	Create an Art Marketplace in Ethereum with ERC-721 Tokens in 10 Minutes Using 0xcert <a href="https://medium.com/@merunasgrincalaitis/create-an-art-marketplace-in-ethereum-with-erc-721-tokens-in-10-minutes-using-0xcert-f059fc951904">https://medium.com/@merunasgrincalaitis/create-an-art-marketplace-in-ethereum-with-erc-721-tokens-in-10-minutes-using-0xcert-f059fc951904</a>
[7]	Building an ERC721 non-fungible token smart contract and using metamask to interact with it via a web interface. (PART I) <a href="https://medium.com/coinmonks/building-an-erc721-non-fungible-token-smart-contract-and-using-metamask-to-interact-with-it-via-a-59583f0dd6c1">https://medium.com/coinmonks/building-an-erc721-non-fungible-token-smart-contract-and-using-metamask-to-interact-with-it-via-a-59583f0dd6c1</a>
[8]	Building an ERC721 non-fungible token smart contract and using metamask to interact with it via a web interface. (PART II) <a href="https://medium.com/coinmonks/building-an-erc721-non-fungible-token-smart-contract-and-using-metamask-to-interact-with-it-via-a-3f3a13320572">https://medium.com/coinmonks/building-an-erc721-non-fungible-token-smart-contract-and-using-metamask-to-interact-with-it-via-a-3f3a13320572</a>
[9]	The Challenge of UX in Ethereum <a href="https://medium.com/ecf-review/challenge-of-ux-in-ethereum-122e1a33688d">https://medium.com/ecf-review/challenge-of-ux-in-ethereum-122e1a33688d</a>
[10]	Exploring Non-Fungible Token with Zeppelin Library (ERC721) <a href="https://medium.com/coinmonks/exploring-non-fungible-token-with-zeppelin-library-erc721-399cb180cfaf">https://medium.com/coinmonks/exploring-non-fungible-token-with-zeppelin-library-erc721-399cb180cfaf</a>
[11]	Evolving ERC-721 metadata standards <a href="https://medium.com/blockchain-manchester/evolving-erc-721-metadata-standards-44646c2eb332">https://medium.com/blockchain-manchester/evolving-erc-721-metadata-standards-44646c2eb332</a>
[12]	ERC-721 metadata standards and IPFS



	<a href="https://medium.com/blockchain-manchester/erc-721-metadata-standards-and-ipfs-94b01fea2a89">https://medium.com/blockchain-manchester/erc-721-metadata-standards-and-ipfs-94b01fea2a89</a>
[13]	Klatyn Docs <a href="https://docs.klaytn.com/smart-contract">https://docs.klaytn.com/smart-contract</a>
[14]	Kauri ERC-721 Non-Fungible Token Standard <a href="https://kauri.io/erc-721-non-fungible-token-standard/b382103e9496409c90c495f35940887e/a">https://kauri.io/erc-721-non-fungible-token-standard/b382103e9496409c90c495f35940887e/a</a>
[15]	Ethereum for Architects and Developers <a href="https://www.apress.com/gp/book/9781484240748">https://www.apress.com/gp/book/9781484240748</a>
[16]	Ethereum for Web Developers: Learn to Build Web Applications on top of the Ethereum Blockchain <a href="https://www.amazon.es/Ethereum-Web-Developers-Applications-Blockchain/dp/1484252772">https://www.amazon.es/Ethereum-Web-Developers-Applications-Blockchain/dp/1484252772</a>
[17]	The Blockchain Developer: A Practical Guide for Designing, Implementing, Publishing, Testing, and Securing Distributed Blockchain-based Projects <a href="https://www.amazon.es/Blockchain-Developer-Implementing-Distributed-Blockchain-based/dp/1484248465">https://www.amazon.es/Blockchain-Developer-Implementing-Distributed-Blockchain-based/dp/1484248465</a>