



4 DE JULIO DE 2022

# MEMORIA TRABAJO FINAL CURSO BLOCKCHAIN - UNIR

EXPERTO UNIVERSITARIO EN DESARROLLO BLOCKCHAIN

ANTON POLENYAKA

[HTTPS://WWW.LINKEDIN.COM/IN/ANTONPOLENYAKA/](https://www.linkedin.com/in/antonpolenyaka/)

[HTTPS://GITHUB.COM/ANTONPOLENYAKA](https://github.com/antonpolenyaka)

Versión del documento: 1.0



## Contenido

<b>Introducción</b>	3
Objetivos	3
Detalle del caso de uso	3
Justificación de la tecnología Blockchain en este caso de uso	4
<b>Desarrollo de la solución propuesta</b>	6
Visión global/abstracta/conceptual del proyecto a desarrollar	6
Fases del desarrollo	6
Diagrama casos de uso	7
Descripción del entorno de desarrollo utilizado	8
Arquitectura del sistema	9
Actores implicados	9
Componentes del sistema (diagrama de clases)	10
Smart Contracts	10
Diagrama de clases	13
Diagramas de secuencias	14
<b>Despliegue del sistema</b>	16
<b>Pruebas de validación</b>	21
Manual con llamadas desde Remix IDE	21
Pruebas automatizadas con “truffle test” y ficheros /test/*.sol	21
Pruebas automatizadas con “truffle test” y ficheros /test/*.js	23
Información general sobre los contratos mediante Surya	24
<b>Funcionamiento de la aplicación</b>	27
Parte general: Compra/Venta de tokens FDT (ERC-20)	28
Como realizar la compra de los tokens FDT	29
Como realizar la venta de los tokens FDT	30
Como realizar verificación de numero de tokens en cualquier cuenta FDT	32
Parte propietario de parcela	33
Como realizar alta de una nueva parcela	34
Como solicitar el servicio de fumigación	35
Parte propietario del dron (o empresa gestora)	38
Como realizar alta de nuevo dron	39
Como realizar tareas de fumigación	40
<b>Conclusiones</b>	43



# Introducción

## Objetivos

El objetivo es desarrollo de una aplicación DApp basado en un Blockchain, mediante cual se aplican los conocimientos que tuve oportunidad de adquirir en el “Curso Experto Universitario en Desarrollo de Aplicaciones Blockchain en UNIR”.

Este proyecto podría llegar a ser un MVP (producto mínimo viable), pero le faltaría algunos detalles que en mi opinión serian necesarios para demostrar una tokenomica de mayor interés. Aún así las funcionalidades básicas del sistema están cubiertas y por ello pueden ser mostrados a los posibles inversores para su valoración de la propuesta.

A nivel general este proyecto lo que pretende hacer es lo siguiente:

- Unir dos tipos de clientes en la misma plataforma (propietarios de parcelas y empresas gestoras/propietarias de drones)
- Proporcionar un sistema de pago seguro entre dos tipos de clientes
- Proporcionar transparencia y confianza entre los usuarios del sistema
- Funcionamiento independiente sin una entidad central

## Detalle del caso de uso

Una empresa ha desarrollado un sistema de fumigación con drones y nos ha solicitado que desarrollemos una solución basada en la Blockchain de Alastria para su uso.

Las características propias de los drones son:

- Un identificador único y ascendente, comenzando en 1 y que no puede repetirse.
- La empresa que lo gestiona, que será la única que pueda mandar acciones al dron.
- Altura máxima y mínima de vuelo.
- Una lista de pesticidas que puede suministrar. Los pesticidas existentes son cinco y sus nombres son: Pesticida A, Pesticida B, Pesticida C, Pesticida D y Pesticida E.
- Coste.

La operación de fumigación es inmediata y debe lanzar un evento de parcela fumigada con el ID de la parcela.

Las características de las parcelas son:

- Un identificador único y ascendente, que comienza en 1 y que no puede repetirse.
- Un propietario.
- Altura máxima y mínima de vuelo permitida.
- Pesticida aceptado, que va a ser uno de la lista de pesticidas descrita anteriormente.

Otras operaciones que debe suministrar la plataforma son:

- Contratar un dron a la empresa para desinfectar una parcela con un pesticida determinado.
- Pago de la operación realizada desde la cuenta del propietario a la de la empresa.

Para la gestión de pagos se debe crear un token propio basado en el estándar ERC 20. Además, los drones y las parcelas pueden gestionarse mediante tokens no fungibles basados en el estándar ERC 721.

La empresa solicita tener una interfaz web que le permita registrar los drones y asignarles trabajos. A su vez, también se debe proporcionar una interfaz web para que los propietarios de las parcelas las puedan registrar

y tengan la posibilidad de contratar un dron con las características que requiere su parcela y que pueda desplazarse hasta la misma.

### Justificación de la tecnología Blockchain en este caso de uso

La justificación de la tecnología en este caso la obligación por parte de UNIR realizar lo en Blockchain y por ello no es cuestionable esta situación. Aun así, podemos entender que por motivos pedagógicos teníamos que dar nuestra opinión porque en este caso sería una buena opción utilizar tecnología Blockchain.

En **primer** lugar, podemos basarse en una de los diagramas que nos enseñaron en el curso:

[https://www.researchgate.net/publication/314213262\\_A\\_Taxonomy\\_of\\_Blockchain-Based\\_Systems\\_for\\_Architecture\\_Design](https://www.researchgate.net/publication/314213262_A_Taxonomy_of_Blockchain-Based_Systems_for_Architecture_Design)

Siguiendo la lógica de este diagrama podemos decir, que no tenemos ninguna entidad de confianza, ya que todos los clientes son iguales entre si y no puede uno tener mas poder que otros clientes de la plataforma y por ello tomar decisiones de funcionamiento de la plataforma. Además, tampoco podemos confiar, que esta entidad siga siempre, las empresas pueden cerrarse por diferentes motivos y no podemos confiar una plataforma plenamente en una entidad.

Siendo una plataforma de uso abierto on-chain, no es necesario montar un nuevo Blockchain para ello, pudiendo elegir un Blockchain como Alastria o Blockchain públicos que cubren las necesidades de este proyecto.

Incluso podría ser un sistema donde se fomenta anonimato dependiendo del tipo de cultivos en las parcelas, lo que nos podría ofrecer un pago entre los propietarios de parcelas y los propietarios de drones sin conocer se en persona, etc.

En **segundo** lugar, podemos decir con seguridad, que tema Blockchain es un tema interesante y esta de moda. Tal como se describe en la propuesta del trabajo a realizar se utilizarán los tokens de utilidad ERC20 y tokens NFT basados en ERC721. Hoy en día solo el hecho que una empresa utiliza para sus fines Blockchain y NFT podría tener una subida de precios de acciones, publicidad gratuita en los medios de comunicación, potencial interés de clientes de usar algo nuevo y otros tipos de beneficios que no son estrictamente ligados a la tecnología, sino a su popularidad.

**Tercer** punto a favor, es una mayor seguridad anti ataques de hackers. Si los Smart Contracts están bien hechos y se selecciona un Blockchain confiable, robusto, etc. La probabilidad de que el sistema sea atacado por los hackers y que logran sus metas es menor por no decir nulos. Es evidente que errores en los Smart Contracts pueden tener vulnerabilidades, pero las aplicaciones normales también, así que puestos a elegir un servidor centralizado tiene más números de ser atacado con éxito que Blockchain.

**Cuarto** punto a favor es la descentralización que nos ofrece no solo utilizar redes p2p para la defensa de los hackers, sino también en una parte tener confianza que una entidad no podrá cortar el suministro a una parte de usuarios como sucede con empresas como Google, Amazon, Microsoft, etc. Teniendo soluciones en cloud se corre el máximo riesgo a nivel de negocio, seguridad de país e independencia tecnológica. En últimos años las empresas estadounidenses en su mayoría han ido promoviendo la migración a cloud como si fuera lo mejor que podría pasar a una empresa, institución pública, etc. La parte negativa que se oculta es una mayor dependencia de estas empresas y los gobiernos que tienen influencia sobre ellas. Ya se han visto casos de desconexiones de servicio de millones de usuarios, miles de proyectos de noche a la mañana con todo el perjuicio económico y quiebra de los negocios que esta centralización puede producir en los países enteros como Yugoslavia, Iraq, Irán, Afganistán, Yemen, Palestina, Venezuela, Livia, Rusia, China, Bielorrusia, etc. La solución Blockchain parece que nos da algo mas de confianza, dependiendo de la forma como se monta, tener

mayor control y confianza en que un gobierno o una empresa no podrá quebrar nuestra plataforma cuando ellos lo desean. Desde mi punto de vista es uno de los mayores puntos a favor que tiene el uso de Blockchain frente a soluciones centralizadas.

Aun que no es el caso, como **quinto** punto a favor podría ser una autofinanciación de este proyecto mediante salidas al mercado nuevas como ICO/IDO/etc. Lo que ayudaría a financiar sobre todo en la temprana etapa del proyecto evitando complicaciones de financiaciones tradicionales.

**Sexto** punto a favor de utilizar Blockchain, es que es totalmente auditable debido a su transparencia, lo que ayuda a crear la confianza entre los usuarios no solo entre ellos, sino también al sistema. Teniendo en cuenta que se habla no solo de prestación de servicio, sino también de reserva de dinero entre la solicitud del trabajo y el pago por el trabajo realizado.

**Séptimo** punto a favor es reducción de costes indirectos. En prácticamente todos los proyectos hay costes directos e indirectos. En el caso de uso de Blockchain, dependiendo cual se elige, se puede decir que, si un cliente no utiliza, no paga nada. Y si utiliza, solo paga lo imprescindible que es para realizar una tarea de fumigación de su parcela y si luego durante unos meses por ejemplo de invierno no necesita este servicio, no se le exige nada, ya que solo paga en el momento que necesita el servicio y solo por el servicio, todos los costes indirectos que puede haber están incluidos directamente en el uso de Blockchain en el momento preciso.

**Octavo** punto a favor es una posibilidad de tener un proyecto OpenSource, no solo del código, sino del uso. Los programadores independientes podrían crear aplicaciones web alternativas, aplicaciones de móviles, dispositivos integrados con el sistema, sin necesidad de permiso de una entidad central. Esto los llevaría a estar integrados con el sistema, montar su negocio encima de la plataforma basada en Blockchain sin miedo de pérdida de acceso y sin privilegios siendo un mercado totalmente libre.

## Desarrollo de la solución propuesta

En los siguientes apartados se describe en detalle la solución propuesta del proyecto para cubrir los casos de uso descritos en anteriormente en la propuesta del TFE.

**El código del proyecto desarrollado se encuentra en:**

<https://github.com/antonpolenyaka/TFE-curso-UNIR-ExpertoBlockchain.git>

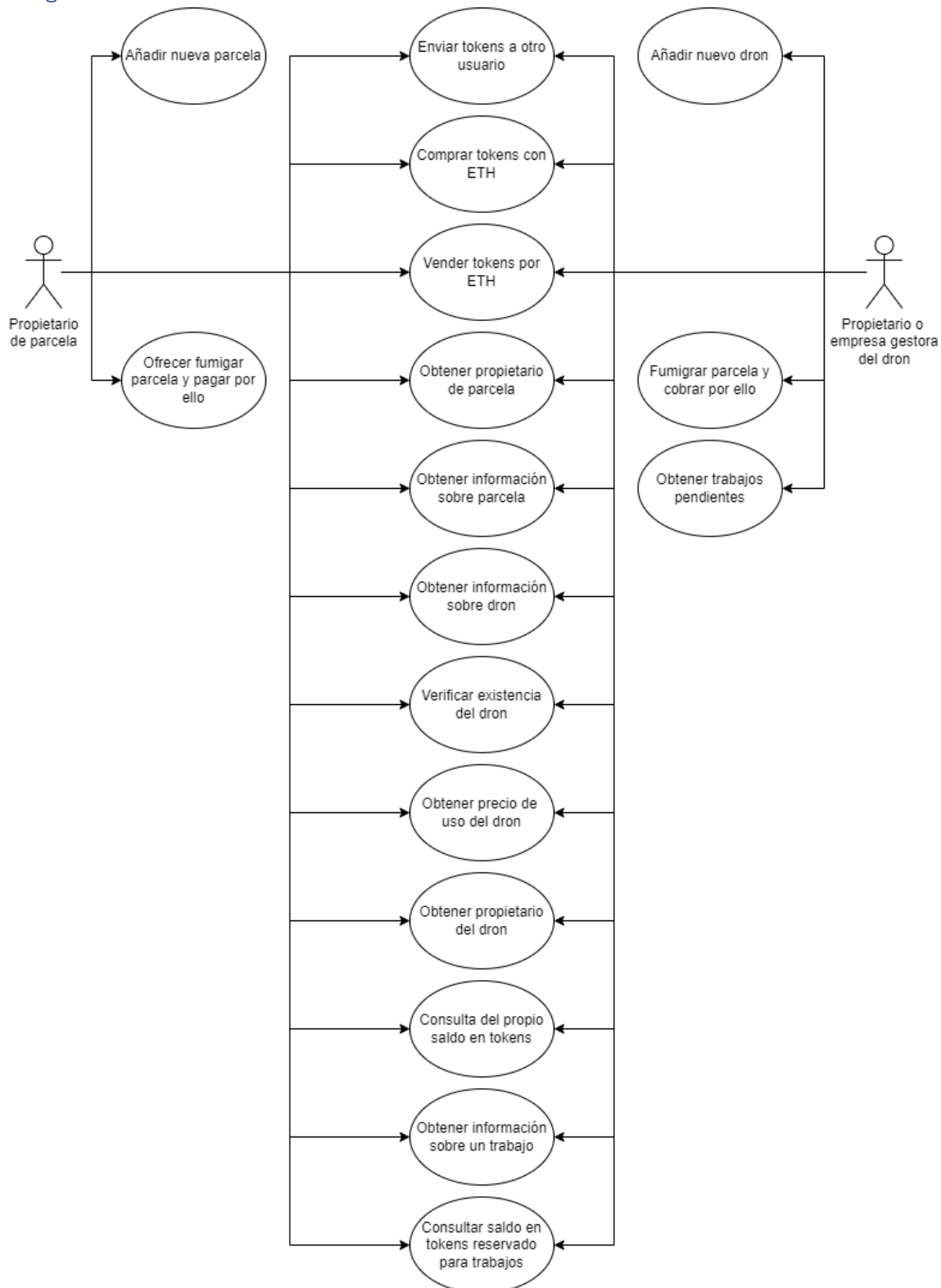
### Visión global/abstracta/conceptual del proyecto a desarrollar

Dada la complejidad del proyecto y para demostrar los conocimientos adquiridos, en los siguientes apartados se explicarán partes de proyecto a nivel conceptual con o sin uso de diagramas UML como parte de ingeniería de software desarrollado.

### Fases del desarrollo

Siendo un proyecto de aplicación acortada a demostración de conocimiento, se queda fuera del alcance de este proyecto el desarrollo real que puede suponer un proyecto real con unas necesidades reales de día a día. En esta parte solo se consiguen unos objetivos concretos, que en futuro podrían ser mejorados, pero como un TFE/MVP tendría que ser más o menos suficiente.

## Diagrama casos de uso





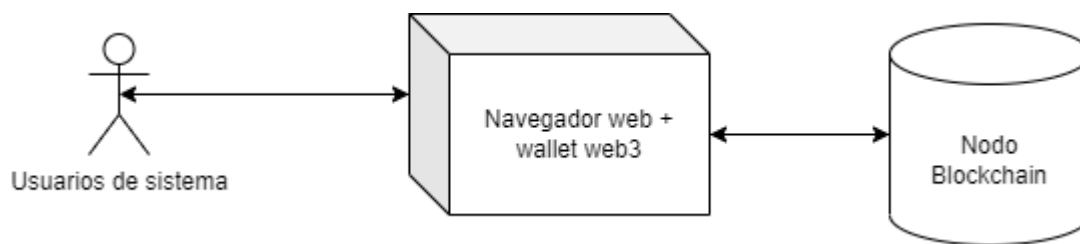
## Descripción del entorno de desarrollo utilizado

Para el desarrollo del proyecto han sido utilizado siguientes herramientas:

1. Herramientas
  - a. Remix IDE v0.24.1
  - b. Visual Studio Code con extensiones
    - i. Solidity de Juan Blanco v0.0.139
    - ii. Solidity Visual Developer de tintinweb v0.1.2
    - iii. Solidity Language & Theme (only) de tintinweb v0.0.6
  - c. Ganache v.2.5.4
  - d. NodeJS v18.4.0
  - e. Truffle v5.5.19
  - f. Surya v0.4.6
  - g. Biblioteca OpenZeppelin actualizada a 04/07/2022
  - h. Web3 v1.7.4
  - i. ReactJS v16.8.4
  - j. Solidity v0.8.15
  - k. Y otros paquetes indicados en package.json del proyecto
2. Testing
  - a. Truffle test v5.5.19
  - b. Surya v0.4.6
3. Herramientas del navegador
  - a. Chrome v103.0.5060.66
  - b. Extensión Metamask v10.16.1
4. Herramientas frontend
  - a. ReactJS v16.8.4
  - b. JavaScript
  - c. HTML5
  - d. CSS

## Arquitectura del sistema

En este proyecto la arquitectura es bastante simple y se representa en el siguiente diagrama.



## Actores implicados

En principio hay dos actores que interactúan con el sistema:

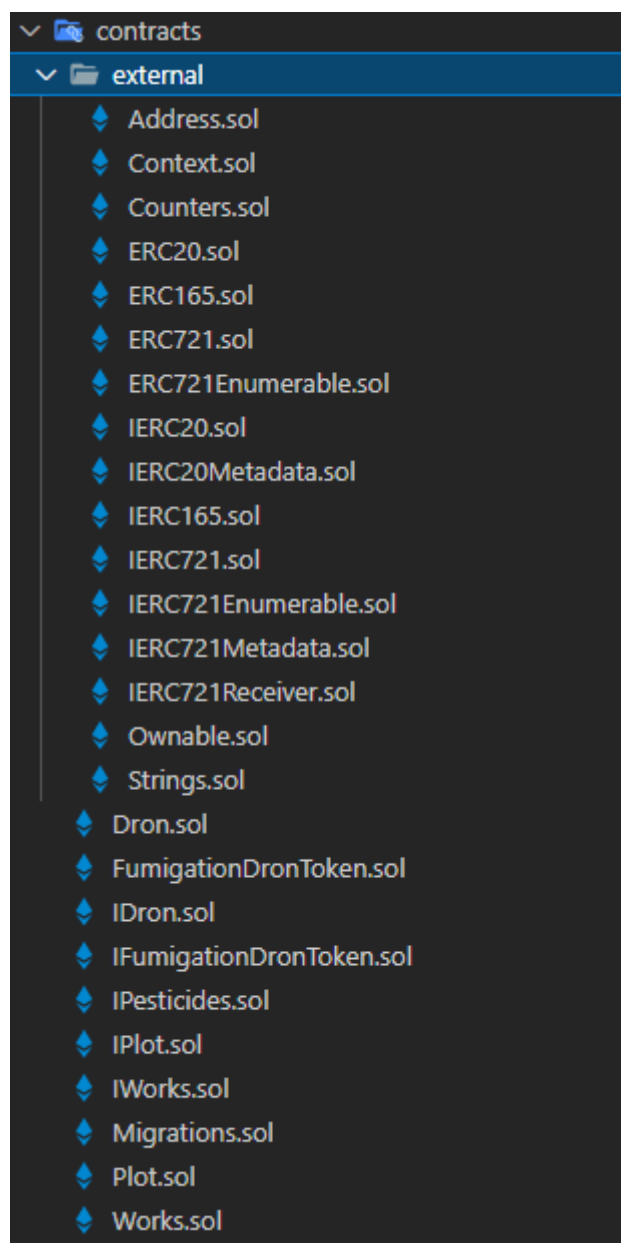
1. Empresa que gestiona los drones / propietario de drones que suministran el servicio de fumigar las parcelas.
2. Propietarios de parcelas que solicitan el servicio de fumigar las parcelas.

## Componentes del sistema (diagrama de clases)

### Smart Contracts

Para el desarrollo del proyecto se han utilizado con Smart Contracts escritos en Solidity. Para facilitar una rápida visión de funcionalidad cada uno de ellos además tiene una interface, que podría ser ampliada su utilidad si sistema creciera y hubiera diferentes variables de implementación de los contratos. Pero en estado actual es mera muestra de conocimiento del lenguaje Solidity.

La estructura es la siguiente:



En la cual “Migrations.sol” es el contrato auxiliar de Truffle y los contratos situados en subcarpeta “external” son copiados desde repositorio OpenZeppelin (podrían ser importados con los @imports directamente a los contratos y descargados automáticamente, pero por la desconfianza a terceros han sido incorporados todos y cada uno ya que son necesarios en este proyecto).

Al tener la copia de código fuente y también subido este código al repositorio en GitHub, no se aporta la copia de código fuente en este documento, ya que se considera innecesario.

A continuación, se detallan los principales contratos desarrollados para llevar a cabo el funcionamiento del sistema.

### **IPesticides.sol**

Permite tener enum con la lista de las pesticidas que pueden ser utilizadas en el sistema.

### **FumigationDronToken.sol & IFumigationDronToken.sol**

Es el contrato de token basado en ERC20. Permite las funcionalidades básicas y además se le añade:

- Que tiene un precio en ETH (o criptomoneda nativa del Blockchain basado en EVM).
- Poder comprar el token entregando ETH al contrato ERC20
- Poder vender el token devolviendo ETH al vendedor según el precio del token
- Transferir tokens entre dos wallets
- Consultar balance propio en tokens
- Emitir evento cuando alguien compra tokens
- Emitir evento cuando alguien vende tokens

Como un detalle especial, no se hace emisión de miles y millones de los tokens y luego se procede su venta. En lugar de ello al principio de existencia del contrato no existen los tokens, no los tiene nadie. Cuando un propietario de una parcela necesita el servicio de fumigación, los puede comprar a cambio de Ethers, los que se quedan en balance del contrato, en este momento contrato los crea (hace “mint” de lo tokens). El propietario del dron una vez realiza el servicio recibe los tokens, los puede cambiar por Ethers directamente en el contrato según el precio fijo establecido o vender los él algún CEX/DEX. De esta manera todos los tokens están respaldados por ETH y solo se crean cuando se necesitan o se destruyen, cuando los cambian por Ethers en el contrato mediante la funcionalidad “burn”.

### **Dron.sol & IDron.sol**

Es el contrato basado en ERC721 que permite manejar los drones y realizar los trabajos:

- Dar de alta dron
- Consultar propietario del dron
- Consultar si existe un dron según el id
- Consultar precio de uso de un dron
- Consultar información sobre dron
- Emitir evento cuando se da de alta un nuevo dron

### **Plot.sol & IPlot.sol**

Es el contrato basado en ERC721 que permite manejar las parcelas y encargar los trabajos:

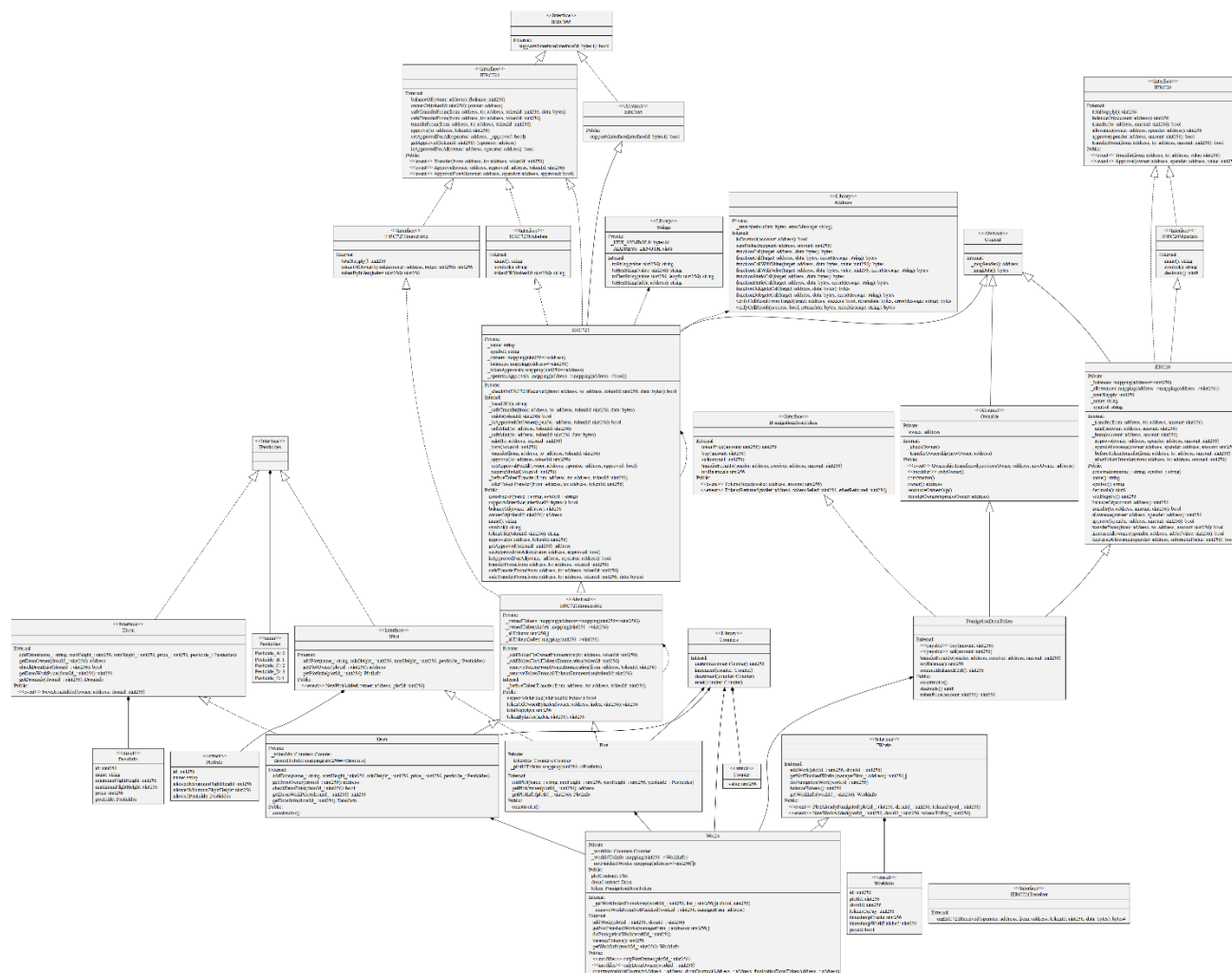
- Dar de alta parcela
- Consultar propietario de parcela
- Consultar información sobre parcela
- Emitir evento cuando se da de alta nueva parcela

### **Works.sol & IWorks.sol**

Es el contrato que tiene la parte de la lógica del negocio que hace el vínculo entre los propietarios de las parcelas y de los drones. Unos encargan los trabajos, el contrato guarda los tokens hasta que trabajo no sea finalizado, luego entrega estos tokens al propietario del dron que realizo el trabajo.

- Creación del encargo del trabajo (solo para propietarios de las parcelas)
- Consulta de trabajos pendientes para un propietario/empresa gestora de drones
- Realización de trabajos de fumigación (solo para propietarios de drones)
- Consulta de balance de tokens reservados para pagar los trabajos encargados
- Consulta de información sobre un trabajo en concreto según su identificador

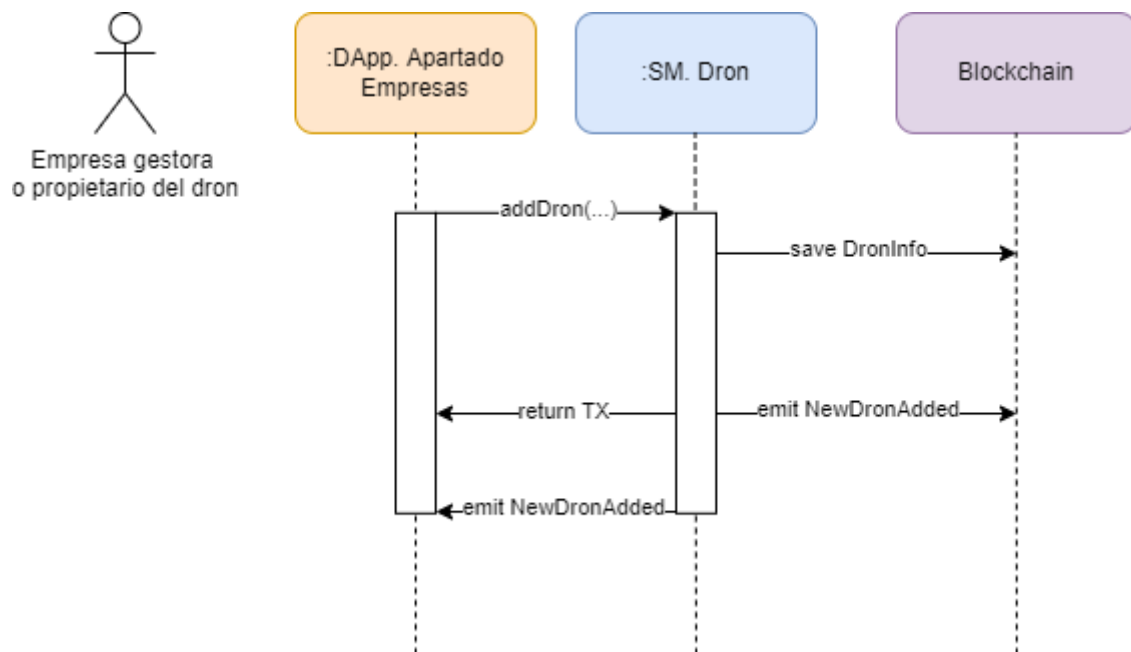
Diagrama generado mediante la herramienta sol2uml.



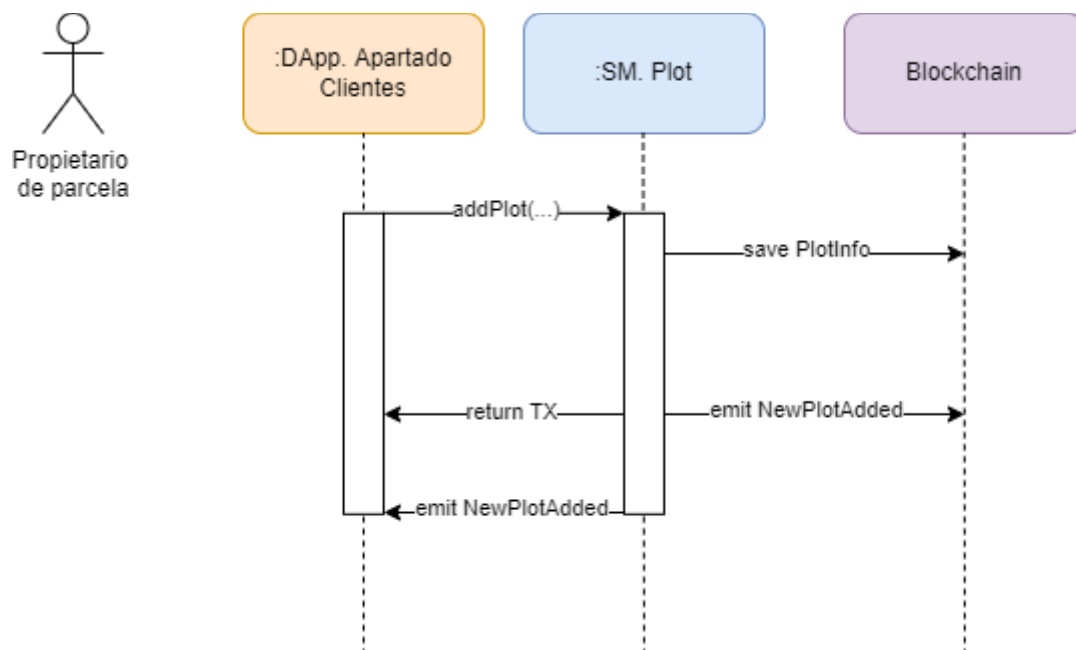
## Diagramas de secuencias

Siendo muchas las funcionalidades de aplicación, pero en mayoría de los casos siendo muy básicas, se exponen varios diagramas que podrían ayudar a entender parte de funcionalidad del proyecto, siendo otras funcionalidades tan básicas que estos diagramas no aportarían ningún valor.

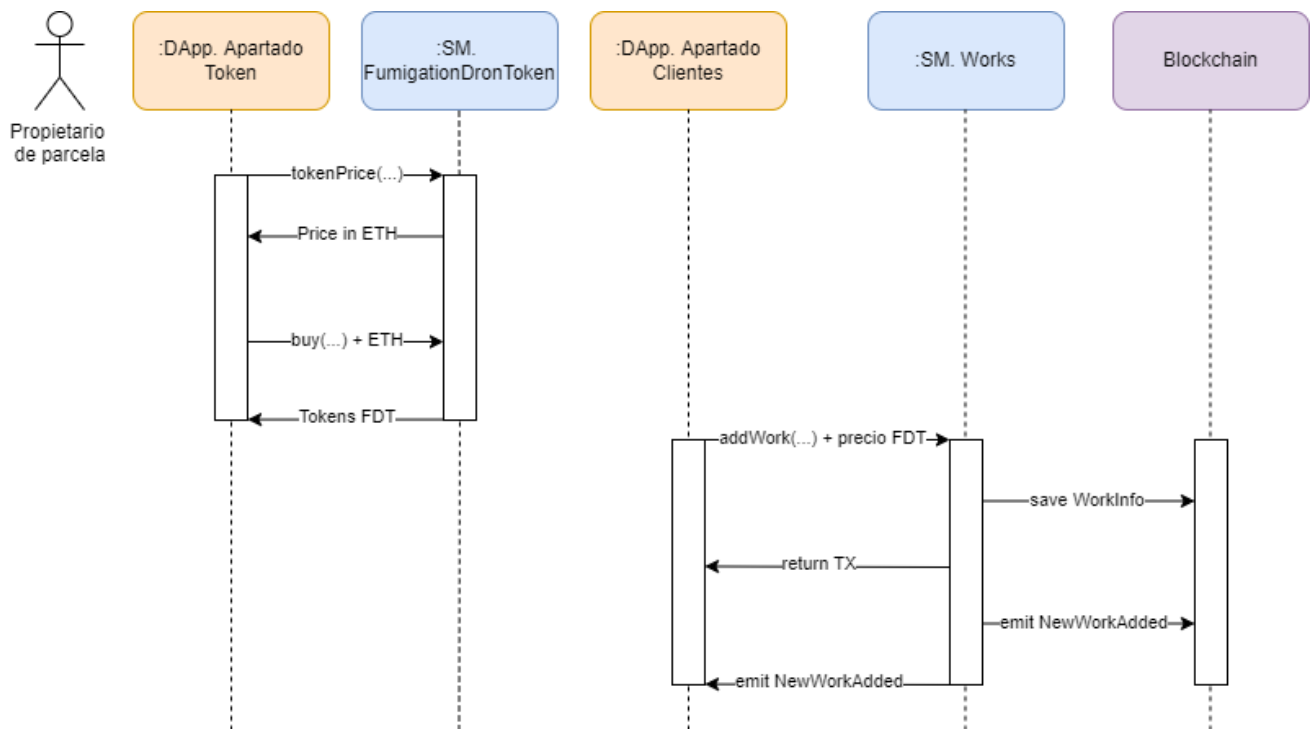
### Secuencia para crear nuevo Dron



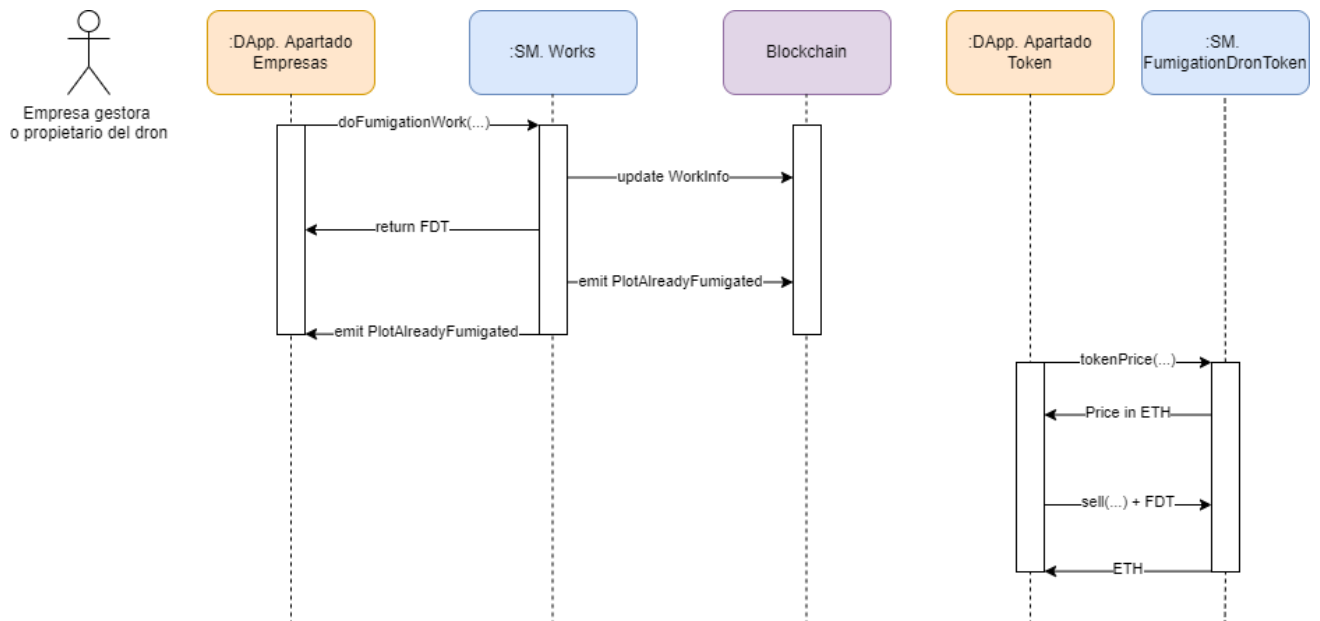
### Secuencia para crear nueva Parcela (Plot)



### Secuencia para para contratación de servicio de fumigación

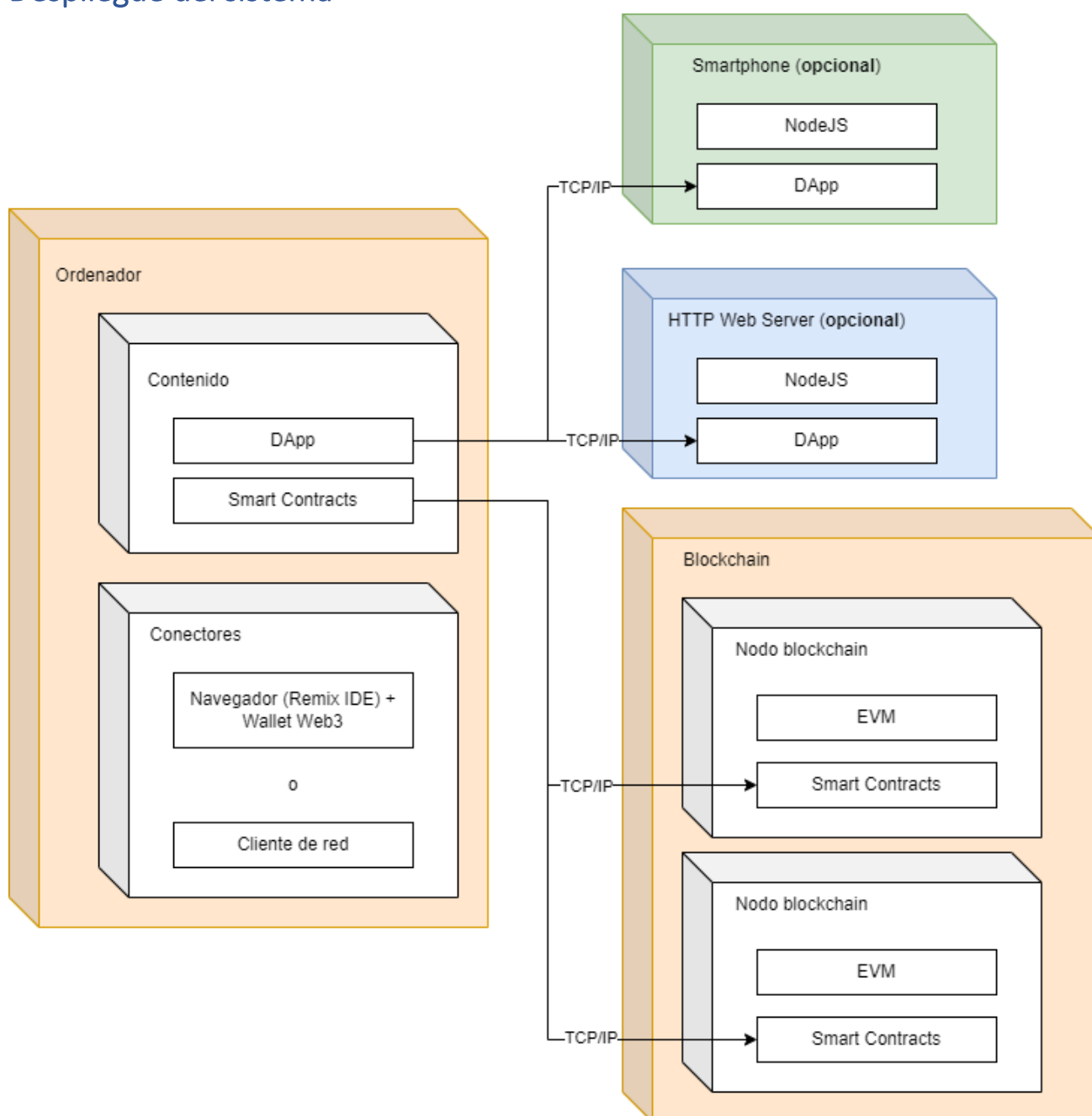


### Secuencia para para realización de servicio de fumigación





## Despliegue del sistema



El código del proyecto desarrollado se encuentra en:

<https://github.com/antonpolenyaka/TFE-curso-UNIR-ExpertoBlockchain.git>

Para realizar el despliegue hay que seguir siguientes pasos (en apartados anteriores se mencionaban todas las herramientas y las versiones):

1. La parte Blockchain
  - a. Instalar NodeJS.
  - b. Crear directorio para proyecto.
  - c. En CI instalar Truffle con npm.
  - d. (opcional) Instalar Ganache si es para probar en red local.

- e. (opcional) Ejecutar red Ganache con un workspace por ejemplo 10 cuentas con 100 ETH cada una.
- f. Bajar con git el contenido del repositorio.
- g. En CI compilar los contratos con “truffle compile”. No debería dar problemas.
- h. Retocar el fichero truffle-config.js para añadir nueva red donde desplegar si hace falta.
- i. En la raíz crear fichero “.secret” con la mem frase de la cuenta que se utilizara para pagar el gas del despliegue.
- j. Desplegar los contratos con “truffle migrate --network **development** --reset”

En lugar de development puede ser indicada otra red por ejemplo bscTestnet, matic.

Se vera resultado similar al siguiente:

```
PS C:\GitHub\TFE-curso-UNIR-ExpertoBlockchain> truffle migrate --network development --reset

Compiling your contracts...
=====
> Compiling .\src\contracts\Dron.sol
> Compiling .\src\contracts\FumigationDronToken.sol
> Compiling .\src\contracts\IDron.sol
> Compiling .\src\contracts\IFumigationDronToken.sol
> Compiling .\src\contracts\IPesticides.sol
> Compiling .\src\contracts\IPlot.sol
> Compiling .\src\contracts\IWorks.sol
> Compiling .\src\contracts\Migrations.sol
> Compiling .\src\contracts\Plot.sol
> Compiling .\src\contracts\Works.sol
> Compiling .\src\contracts\external\Address.sol
> Compiling .\src\contracts\external\Context.sol
> Compiling .\src\contracts\external\Counters.sol
> Compiling .\src\contracts\external\ERC165.sol
> Compiling .\src\contracts\external\ERC20.sol
> Compiling .\src\contracts\external\ERC721.sol
> Compiling .\src\contracts\external\ERC721Enumerable.sol
> Compiling .\src\contracts\external\IERC165.sol
> Compiling .\src\contracts\external\IERC20.sol
> Compiling .\src\contracts\external\IERC20Metadata.sol
> Compiling .\src\contracts\external\IERC721.sol
> Compiling .\src\contracts\external\IERC721Enumerable.sol
> Compiling .\src\contracts\external\IERC721Metadata.sol
> Compiling .\src\contracts\external\IERC721Receiver.sol
> Compiling .\src\contracts\external\Ownable.sol
> Compiling .\src\contracts\external\Strings.sol
> Artifacts written to C:\GitHub\TFE-curso-UNIR-ExpertoBlockchain\src\abis
> Compiled successfully using:
   - solc: 0.8.15+commit.e14f2714.Emscripten.clang

Starting migrations...
=====
> Network name:      'development'
> Network id:       5777
> Block gas limit:  6721975 (0x6691b7)
```

```

1_initial_migration.js
=====

Replacing 'Migrations'
-----
> transaction hash:    0x42e86dddf7492932bb2efb17a420a913150f815dde8421ffd71768aed3f21b13
> Blocks: 0           Seconds: 0
> contract address:    0x447eD53182f1955c51a01bf2938da73199883023
> block number:        1211
> block timestamp:     1656950505
> account:             0x36097e8A6005120891aea8Ef80A5494C1722f751
> balance:             961.61776738
> gas used:            174952 (0x2ab68)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.00349904 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:          0.00349904 ETH

2_dron_migration.js
=====

Replacing 'FumigationDronToken'
-----
> transaction hash:    0xe704d8d1fdb82606ae83cdb5cde5fbb539ecda2ea3f71b7974e97533427406c8
> Blocks: 0           Seconds: 0
> contract address:    0x9e47E1319A4CaC45fddfd703131562703a6afec4
> block number:        1213
> block timestamp:     1656950507
> account:             0x36097e8A6005120891aea8Ef80A5494C1722f751
> balance:             961.59479936
> gas used:            1106090 (0x10e0aa)
> gas price:           20 gwei
> value sent:          0 ETH
> total cost:          0.0221218 ETH

```

```

Replacing 'Dron'
-----
> transaction hash: 0xdfa171f2c2f81b7fcc5a8a961adf307a08ef9fcd9212c36c0ddb5189b9939f94
> Blocks: 0 Seconds: 0
> contract address: 0xd5fA36e342541Dd721B172d3FBEE5cD2eE0fE2Af
> block number: 1214
> block timestamp: 1656950509
> account: 0x36097e8A6005120891aea8Ef80A5494C1722f751
> balance: 961.5589798
> gas used: 1790978 (0x1b5402)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.03581956 ETH

```

```

Replacing 'Plot'
-----
> transaction hash: 0xb5395f7ae144b7ce34c25fe20d498036ee04bc4a1aed3d0e1f9b1baef5cb586d
> Blocks: 0 Seconds: 0
> contract address: 0x847Ffd8fA70645B8430ae6676071f8693e23fD8b
> block number: 1215
> block timestamp: 1656950510
> account: 0x36097e8A6005120891aea8Ef80A5494C1722f751
> balance: 961.52561336
> gas used: 1668322 (0x1974e2)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.03336644 ETH

```

```

Replacing 'Works'
-----
> transaction hash: 0x09a623e7fbfb93c67c564a059930583538e7ed3d55a474d94f1b6ab9d7726229
> Blocks: 0 Seconds: 0
> contract address: 0xF00035bF13D8Ef1880CA5432bf7aDcA61c9D8e0f
> block number: 1216
> block timestamp: 1656950512
> account: 0x36097e8A6005120891aea8Ef80A5494C1722f751
> balance: 961.5023196
> gas used: 1164688 (0x11c590)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.02329376 ETH

```

```

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.11460156 ETH

```

```

Summary
=====
> Total deployments: 5
> Final cost: 0.1181006 ETH

```

```
PS C:\GitHub\TFE-curso-UNIR-ExpertoBlockchain> █
```

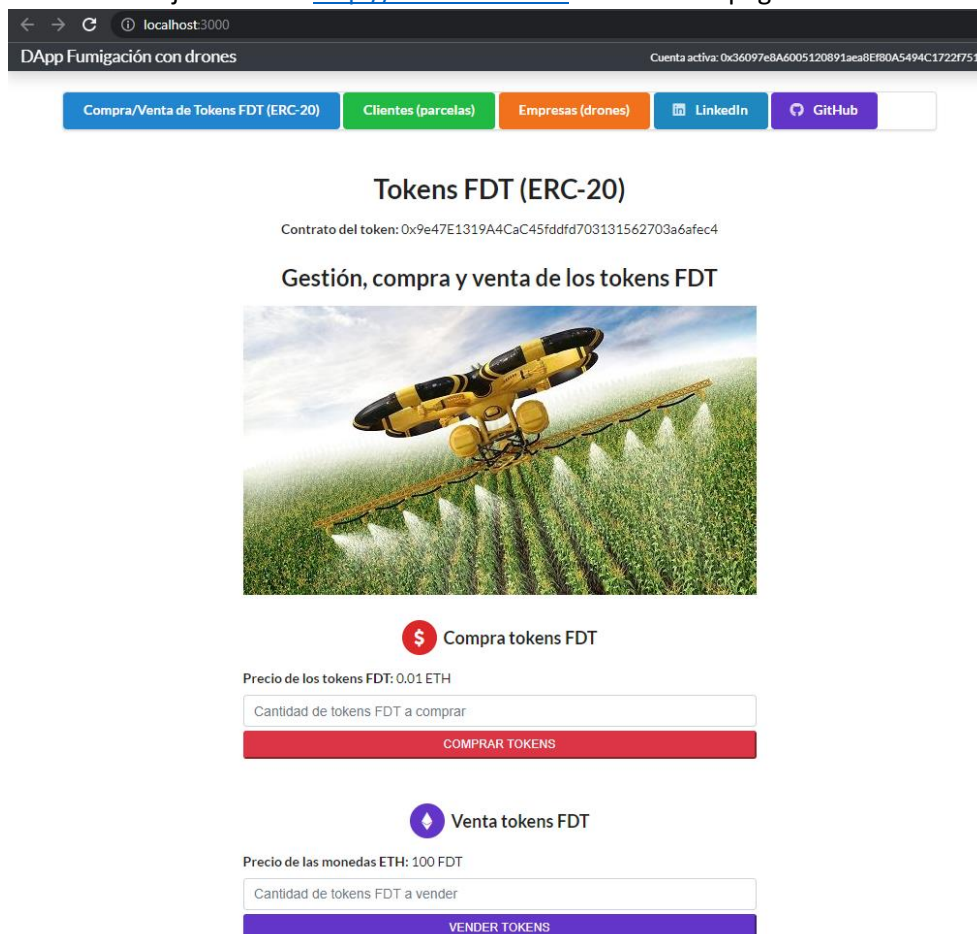
- k. De esta información tras publicación hay que obtener los números de 4 contratos desplegados, por ejemplo, del contrato Dron.sol:

```
Replacing 'Dron'
-----
> transaction hash: 0xdfa171f2c2f81b7fcc5a8a961adf307a08ef9fcd9212c36c0ddb5189b9939f94
> block: 0
> contract address: 0xd5fA36e342541Dd721B172d3FBEE5cD2eE0fE2Af
> block number: 1214
> block timestamp: 1656950509
> account: 0x36097e8A6005120891aea8Ef80A5494C1722f751
> balance: 961.5589798
> gas used: 1790978 (0x1b5402)
> gas price: 20 gwei
> value sent: 0 ETH
> total cost: 0.03581956 ETH
```

- l. Una vez apuntados las direcciones de contratos hay que también apuntar el identificador de la red, por ejemplo, ganache 5777, Ethereum 1, etc.
- m. Editar fichero situado en “\TFE-curso-UNIR-ExpertoBlockchain\src\components\config.js” indicando el numero de la red y las direcciones de contratos.

```
src > components > config.js > ...
1 export const FumigationDronTokenAddress = "0x8402f58d922343e73CA6De6518c26cF65bDe101A";
2 export const DronAddress = "0xBc94ad08A14509a9d496aa51b6E1888dAeb4c9C3";
3 export const PlotAddress = "0xF8D205E8b8dF0b7E2956838d9ddB418444E3ff7D";
4 export const WorksAddress = "0xE5F30ca47bA90226FA4767d0166Ebc12dfB40a31";
5 export const NetworkIdToUse = 5777;
6 export const TokenDecimals = 2;
```

- n. Una vez desplegados los contratos en algún Blockchain, solo falta publicar la web en móviles, en algún servidor web o realizar prueba en local. Para la prueba en local, dentro de la carpeta del proyecto ejecutar comando “npm start”. En caso del servidor local si no se configuro lo contrario se ejecutará en <http://localhost:3000>. Se vera una página web:

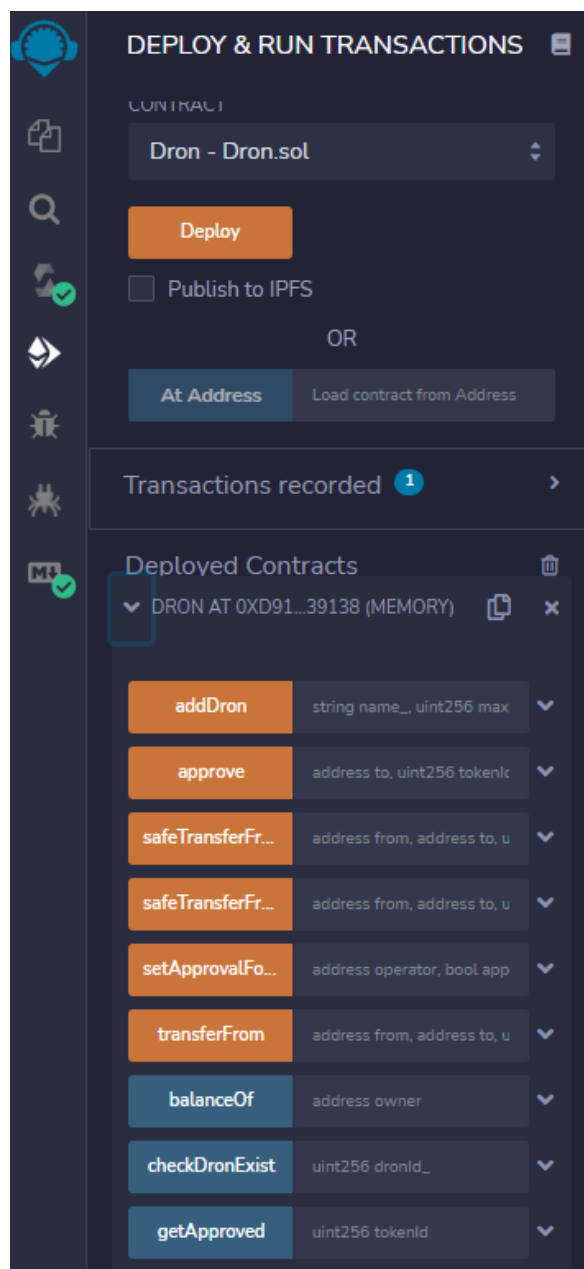


## Pruebas de validación

Para las pruebas de los Smart Contracts se han utilizado 3 técnicas. El código fuente de los ficheros que se utilizan para las pruebas están en la carpeta raíz “test” publicada en GitHub.

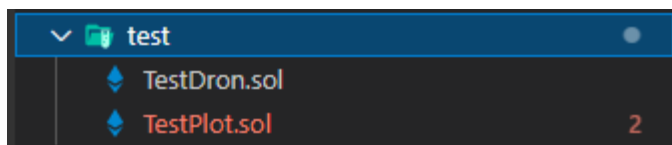
### Manual con llamadas desde Remix IDE

Lo que ha permitido detectar los errores y corregir los rápidamente sin apenas tener la pagina web preparada con interacción web3.



### Pruebas automatizadas con “truffle test” y ficheros /test/\*.sol

En la carpeta raíz del proyecto se encuentra carpeta “test”, en la cual hay varios ficheros de test de Smart Contracts. La prueba de estos ficheros se ejecuta con el comando “truffle test”.



Tras ejecución vemos los resultados en el terminal:

```
PS C:\GitHub\TFE-curso-UNIR-ExpertoBlockchain> truffle test
Using network 'development'.

Compiling your contracts...
=====
> Compiling .\src\contracts\Dron.sol
> Compiling .\src\contracts\FumigationDronToken.sol
> Compiling .\src\contracts\IDron.sol
> Compiling .\src\contracts\IFumigationDronToken.sol
> Compiling .\src\contracts\IPesticides.sol
> Compiling .\src\contracts\IPlot.sol
> Compiling .\src\contracts\IWorks.sol
> Compiling .\src\contracts\Migrations.sol
> Compiling .\src\contracts\Plot.sol
> Compiling .\src\contracts\Works.sol
> Compiling .\src\contracts\external\Address.sol
> Compiling .\src\contracts\external\Context.sol
> Compiling .\src\contracts\external\Counters.sol
> Compiling .\src\contracts\external\ERC165.sol
> Compiling .\src\contracts\external\ERC20.sol
> Compiling .\src\contracts\external\ERC721.sol
> Compiling .\src\contracts\external\ERC721Enumerable.sol
> Compiling .\src\contracts\external\IERC165.sol
> Compiling .\src\contracts\external\IERC20.sol
> Compiling .\src\contracts\external\IERC20Metadata.sol
> Compiling .\src\contracts\external\IERC721.sol
> Compiling .\src\contracts\external\IERC721Enumerable.sol
> Compiling .\src\contracts\external\IERC721Metadata.sol
> Compiling .\src\contracts\external\IERC721Receiver.sol
> Compiling .\src\contracts\external\Ownable.sol
> Compiling .\src\contracts\external\Strings.sol
> Compiling .\test\TestDron.sol
> Compiling .\test\TestPlot.sol
> Compiling truffle\Assert.sol
> Compiling truffle\AssertAddress.sol
> Compiling truffle\AssertAddressArray.sol
> Compiling truffle\AssertBalance.sol
> Compiling truffle\AssertBool.sol
> Compiling truffle\AssertBytes32.sol
> Compiling truffle\AssertBytes32Array.sol
> Compiling truffle\AssertGeneral.sol
> Compiling truffle\AssertInt.sol
> Compiling truffle\AssertIntArray.sol
> Compiling truffle\AssertString.sol
> Compiling truffle\AssertUint.sol
> Compiling truffle\AssertUintArray.sol
> Compiling truffle\DeployedAddresses.sol
> Artifacts written to C:\Users\Anton\AppData\Local\Temp\test--15196-s2sdeSCfaMcz
> Compiled successfully using:
   - solc: 0.8.15+commit.e14f2714.Emscripten.clang
```

```
TestDron
  ✓ testDeployAndGetTotalSupply (1206ms)
  ✓ testSettingAnOwnerOfDeployedContract (2212ms)
  ✓ testDronWorkPrice (1999ms)
  ✓ testCheckIfDronExist (1919ms)
  ✓ testCheckDronInformation (2363ms)

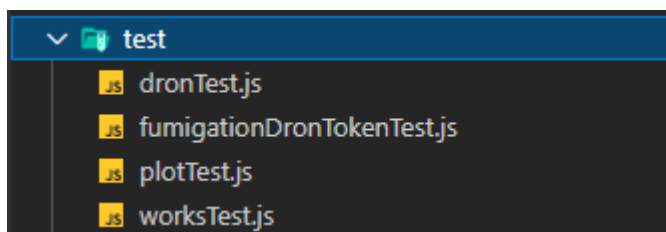
TestPlot
  ✓ testDeployAndGetTotalSupply (1308ms)
  ✓ testSettingAnOwnerOfDeployedContract (2441ms)
  ✓ testCheckPlotInformation (2571ms)

8 passing (1m)

PS C:\Github\TFE-curso-UNIR-ExpertoBlockchain>
```

## Pruebas automatizadas con “truffle test” y ficheros /test/\*.js

En la carpeta raíz del proyecto se encuentra carpeta “test”, en la cual hay varios ficheros de test de Smart Contracts. La prueba de estos ficheros se ejecuta con el comando “truffle test”.



Tras ejecución vemos los resultados en el terminal:

```
PS C:\Github\TFE-curso-UNIR-ExpertoBlockchain> truffle test
Using network 'development'.

Compiling your contracts...
=====
> Compiling .\src\contracts\Dron.sol
> Compiling .\src\contracts\FumigationDronToken.sol
> Compiling .\src\contracts\IDron.sol
> Compiling .\src\contracts\IFumigationDronToken.sol
> Compiling .\src\contracts\IPesticides.sol
> Compiling .\src\contracts\IPlot.sol
> Compiling .\src\contracts\IWorks.sol
> Compiling .\src\contracts\Migrations.sol
> Compiling .\src\contracts\Plot.sol
> Compiling .\src\contracts\Works.sol
> Compiling .\src\contracts\external\Address.sol
> Compiling .\src\contracts\external\Context.sol
> Compiling .\src\contracts\external\Counters.sol
> Compiling .\src\contracts\external\ERC165.sol
> Compiling .\src\contracts\external\ERC20.sol
> Compiling .\src\contracts\external\ERC721.sol
> Compiling .\src\contracts\external\ERC721Enumerable.sol
> Compiling .\src\contracts\external\IERC165.sol
> Compiling .\src\contracts\external\IERC20.sol
> Compiling .\src\contracts\external\IERC20Metadata.sol
> Compiling .\src\contracts\external\IERC721.sol
> Compiling .\src\contracts\external\IERC721Enumerable.sol
> Compiling .\src\contracts\external\IERC721Metadata.sol
> Compiling .\src\contracts\external\IERC721Receiver.sol
> Compiling .\src\contracts\external\Ownable.sol
> Compiling .\src\contracts\external\Strings.sol
> Artifacts written to C:\Users\Anton\AppData\Local\Temp\test--11588-vMgyb5YtelwpK
> Compiled successfully using:
   - solc: 0.8.15+commit.e14f2714.Emscripten.clang
```



```

Contract: Dron
✓ 1. Despues del despliegue no tienen que haver drones creados (1121ms)
✓ 2. Despues de anyadir nuevo dron, su owner tiene que ser cuenta de la primera cuenta (2530ms)
✓ 3. Comprueba si el precio es correcto despues de añadir dron a blockchain (2587ms)
✓ 4. Comprueba que el dron existe (2629ms)
✓ 5. Comprueba que información sobre dron esta añadida correctamente (2654ms)

Contract: FumigationDronToken
✓ 1. Averiguar el precio del un token en ETH (1037ms)
✓ 2. Comprar 100 tokens y verificar balance (1752ms)
✓ 3. Comprar 100 tokens. Vender 90. Verificar balance (2775ms)
✓ 4. Comprar 100 tokens. Transferir a otra cuenta. Verificar balance en las dos cuentas (3060ms)

Contract: Plot
✓ 1. Despues del despliegue no tienen que haver parcelas creadas (1236ms)
✓ 2. Despues de anyadir nueva parcela, su owner tiene que ser cuenta de la primera cuenta (2573ms)
✓ 3. Comprueba que información sobre parcela esta añadida correctamente (2951ms)

Contract: Works
✓ 1. Despues del despliegue no tienen que haver tokens en el saldo del contrato (3533ms)
✓ 2. Añadir trabajo al pool de trabajos y se obtiene este trabajo desde blockchain. Tambien se verifica balance reservado en contrato. (10562ms)
✓ 3. Añadir trabajo y verificar que este trabajo aún no esta finalizado (10451ms)
✓ 4. Realizar el trabajo de fumigacion, recibir tokens por el trabajo y trabajo tener marcado como finalizado (13597ms)

16 passing (1m)
PS C:\Github\TFE-curso-UNIR-ExpertoBlockchain>





```







## Información general sobre los contratos mediante Surya

En la raíz de la carpeta se guarda el fichero SuryaReport.md que contiene visión general sobre los Smart Contracts.

Sūrya's Description Report	
Files Description Table	
File Name	SHA-1 Hash
./src/contracts/IDron.sol	2e68d028c54cda0e76a9b93d8b0b30d7a0648f68
./src/contracts/IFumigationDronToken.sol	1a42e5fd430982f7d7f84fb4c1d7a9538f2ec744
./src/contracts/IPesticides.sol	2241bd9bfef3a61c53c8f9b2d07084c61bb1f91b
./src/contracts/IPlot.sol	340b852cae0c9842ac0421fd5f2ede6dd196058d
./src/contracts/IWorks.sol	573ea7e7a675ff90e95758d3ec01f7d6da2ae6ff
./src/contracts/Dron.sol	8722089a6e3773bb47d5c0d2b79b3d7f23679a6d
./src/contracts/FumigationDronToken.sol	51f75361a81831d33d831aac00f5ea3e035b3583
./src/contracts/Plot.sol	d83d51bc01ef4f5dd7078e1be675a5565a38f5eb
./src/contracts/Works.sol	2900a4f83d96b715aa83eb4f7db1d288dc5cc9fc

Contracts Description Table

Contract	Type	Bases		
L	Function Name	Visibility	Mutability	Modifiers
<b>IDron</b>	Interface	IPesticides		
L	addDron	External !		NO !
L	getDronOwner	External !		NO !
L	checkDronExist	External !		NO !
L	getDronWorkPrice	External !		NO !
L	getDronInfo	External !		NO !
<b>IFumigationDronToken</b>	Interface			
L	tokenPrice	External !		NO !
L	buy	External !		NO !
L	sell	External !		NO !
L	transferFromTo	External !		NO !
L	myBalance	External !		NO !
<b>IPesticides</b>	Interface			

<b>IPlot</b>	Interface	IPesticides		
L	addPlot	External !		NO !
L	getPlotOwner	External !		NO !
L	getPlotInfo	External !		NO !
<b>IWorks</b>	Interface			
L	addWork	External !		NO !
L	getNotFinishedWorks	External !		NO !
L	doFumigationWork	External !		NO !
L	balanceTokens	External !		NO !
L	getWorkInfo	External !		NO !
<b>Dron</b>	Implementation	IDron, ERC721Enumerable		
L		Public !		ERC721
L	addDron	External !		NO !
L	getDronOwner	External !		NO !
L	checkDronExist	External !		NO !
L	getDronWorkPrice	External !		NO !
L	getDronInfo	External !		NO !

FumigationDronToken	Implementation	ERC20, Ownable, IFumigationDronToken		
L		Public !	●	ERC20
L	decimals	Public !		NO !
L	tokenPrice	Public !		NO !
L	buy	External !	💰	NO !
L	sell	External !	💰	NO !
L	transferFromTo	External !	●	NO !
L	myBalance	External !		NO !
L	contractBalanceETH	External !		NO !
Plot	Implementation	ERC721Enumerable, IPlot		
L		Public !	●	ERC721
L	addPlot	External !	●	NO !
L	getPlotOwner	External !		NO !
L	getPlotInfo	External !		NO !

Works	Implementation	IWorks		
L		Public !	●	NO !
L	addWork	External !	●	onlyPlotOwner
L	getNotFinishedWorks	External !		NO !
L	_getWorkIndexFromArray	Internal 🗑		
L	_removeWorkFromNotFinished	Internal 🗑	●	
L	doFumigationWork	External !	●	onlyDronOwner
L	balanceTokens	External !		NO !
L	getWorkInfo	External !		NO !

### Legend

Symbol	Meaning
●	Function can modify state
💰	Function is payable

## Funcionamiento de la aplicación

En este apartado se presenta como funciona la parte DApp en conjunto con los Smart Contracts situados en este caso en la red Ganache de prueba. Para entender mejor el funcionamiento se adjuntan las pantallas para ver mejor la operativa de nuestra DApp.

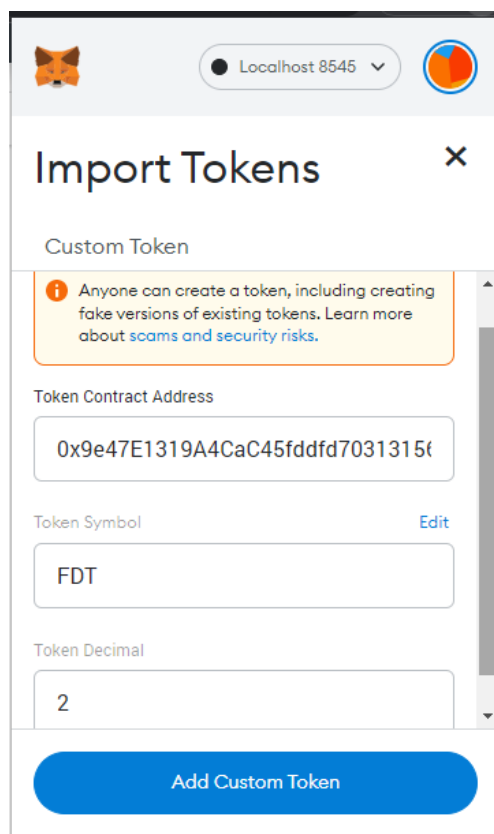
Tener en cuenta, que visualización y gestión se hace en la red local ganache. Para poder gestionar la parte de cliente que tiene parcelas y la parte cliente/proveedor que tiene drones, se eligen dos cuentas de **Ganache** y se importan al **Metamask**. También para estas cuentas se importa el contrato de tokens (el número del contrato se puede obtener en fichero "\\TFE-curso-UNIR-ExpertoBlockchain\\src\\components\\**config.js**" modificado tras el desplégue de los contratos).

El token en Metamask tiene que aparecer con el símbolo FDT (abreviado de FumigationDronToken).

La cuenta "**0x36097e8A6005120891aea8Ef80A5494C1722f751**" se utilizará para representar a cliente **propietario de parcelas**.

La cuenta "**0x12794F370758BcE4a4B503Ecbe07EE65237C9932**" se utilizará para representar a cliente/proveedor **propietario de drones**.

El contrato "**0x9e47E1319A4CaC45fddfd703131562703a6afec4**" que representa FumigationDronToken se utilizara para el **token FDT**.

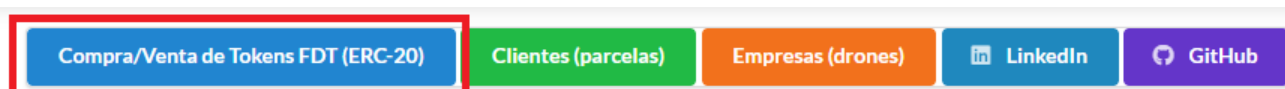


The screenshot shows the 'Import Tokens' interface in Metamask. At the top, there's a header with the Metamask logo, a network selector set to 'Localhost 8545', and a profile icon. The main title is 'Import Tokens' with a close button. Below this, the 'Custom Token' section is active. It includes a warning message: 'Anyone can create a token, including creating fake versions of existing tokens. Learn more about [scams](#) and [security risks](#).' The form fields are: 'Token Contract Address' with the value '0x9e47E1319A4CaC45fddfd703131562703a6afec4', 'Token Symbol' with the value 'FDT' and an 'Edit' link, and 'Token Decimal' with the value '2'. A large blue button at the bottom says 'Add Custom Token'.

Al entrar a DApp se activará Metamask y hay que logearse con vuestra contraseña. Sino funciona bien la extensión, se puede recargar la página web y volverá hacer el intento de conexión con el **wallet web3**.

## Parte general: Compra/Venta de tokens FDT (ERC-20)

Al entrar a la página web (DApp) se nos dirige al apartado “Compra/Venta de Tokens FDT (ERC-20)”. Si estamos en otro apartado, nos podemos dirigir a este pulsando el menú correspondiente.



### Tokens FDT (ERC-20)

Contrato del token: 0x9e47E1319A4CaC45fddfd703131562703a6afec4

#### Gestión, compra y venta de los tokens FDT



#### Compra tokens FDT

Precio de los tokens FDT: 0.01 ETH

COMPRAR TOKENS

#### Venta tokens FDT

Precio de las monedas ETH: 100 FDT


VENDER TOKENS

#### Balance de tokens FDT de un usuario

BALANCE USUARIO

## Como realizar la compra de los tokens FDT

Introducir numero de tokens a comprar en el apartado “Comprar tokens FDT”. No se revisa correcta introducción de datos en la parte web, al no ser objetivo de este curso.

 **Compra tokens FDT**

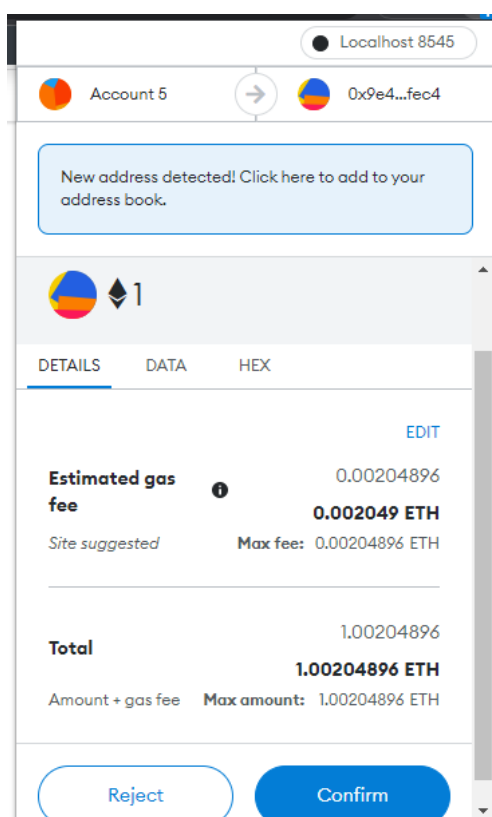
Precio de los tokens FDT: 0.01 ETH

100

COMPRAR TOKENS

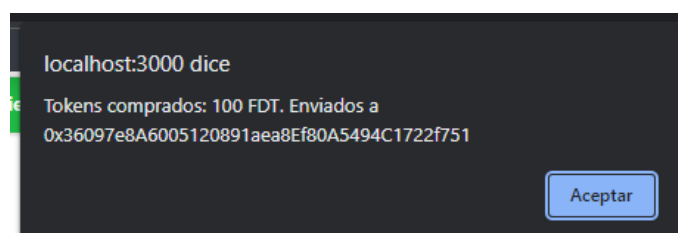
Después pulsar botón “Comprar tokens”.

Se le abrirá ventana del Metamask para confirmar la compra, tener en cuenta que el tipo de cambio son 1 FDT = 0.01 ETH.

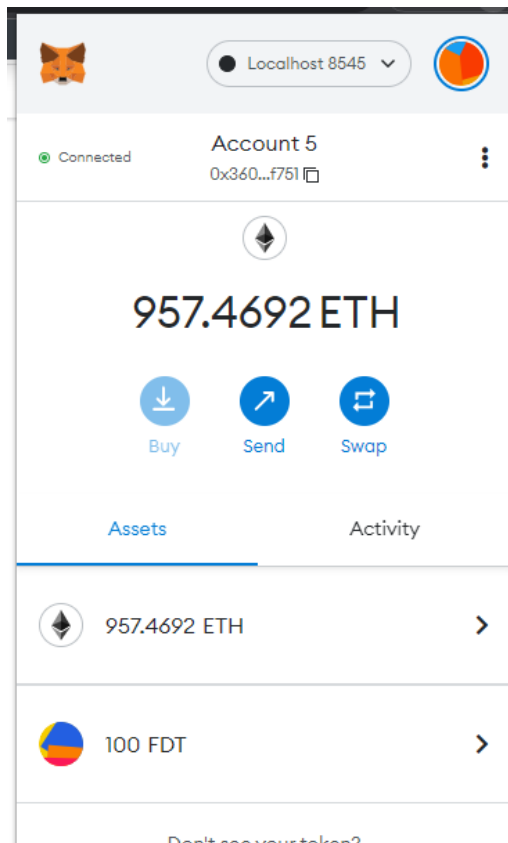


Pulsa “Confirm” para proceder con la operación de compra de tokens FDT.

Tras la compra, aparecerá una alerta en el navegador.




Pulsa aceptar y puede revisar su cuenta en Metamask, tienen que aparecer los tokens FDT.



Como realizar la venta de los tokens FDT

Introducir número de tokens a vender en el apartado “Venta tokens FDT”. No se revisa correcta introducción de datos en la parte web, al no ser objetivo de este curso.

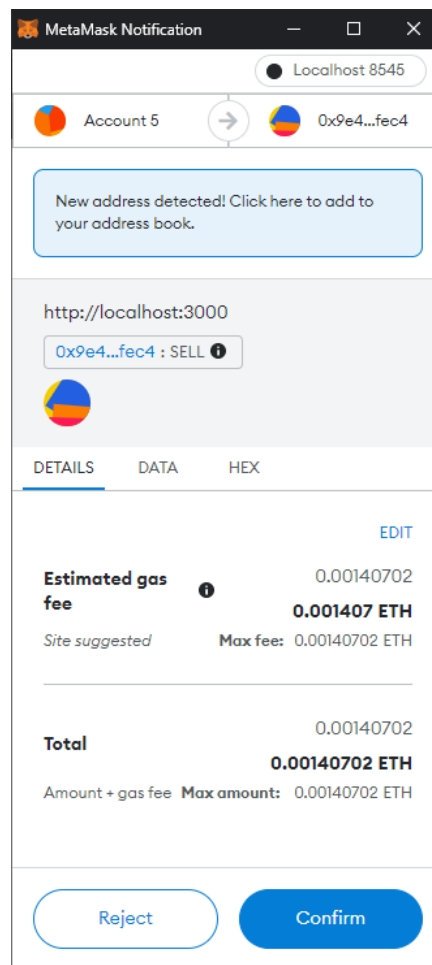
 **Venta tokens FDT**

Precio de las monedas ETH: 100 FDT

VENDER TOKENS

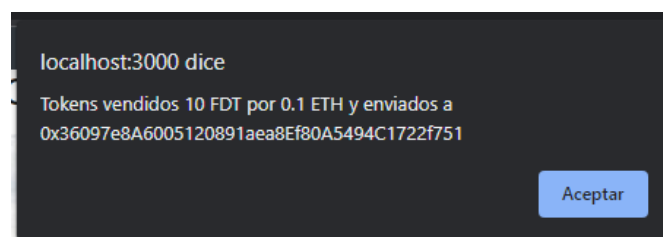
Después pulsar botón “Vender tokens”.

Se le abrirá ventana del Metamask para confirmar la compra, tener en cuenta que el tipo de cambio son 1 FDT = 0.01 ETH.



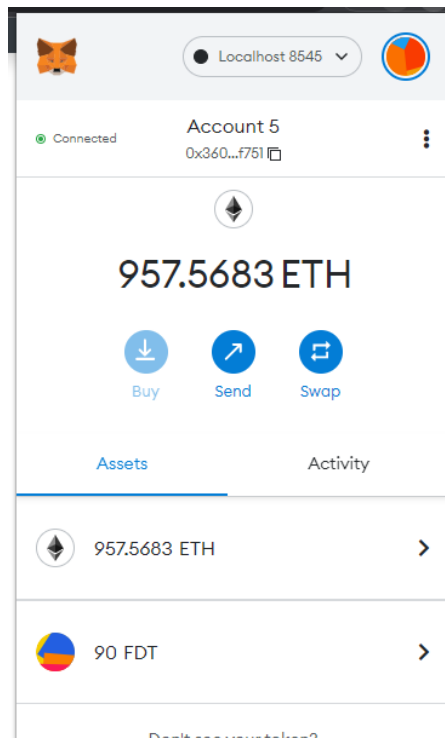
Pulsa “Confirm” para proceder con la operación de venta de tokens FDT.

Tras la venta, aparecerá una alerta en el navegador.



Pulsa aceptar y puede revisar su cuenta en Metamask, tenían que haber incrementado ETH y decrementados tokens FDT.





Como realizar verificación de numero de tokens en cualquier cuenta FDT

Introducir número de wallet en el apartado “Balance de tokens FDT de un usuario”. No se revisa correcta introducción de datos en la parte web, al no ser objetivo de este curso.

#### Balance de tokens FDT de un usuario

BALANCE USUARIO

Después pulsar botón “Balance usuario”.

Tras la pulsación, aparecerá una alerta en el navegador indicando el número de tokens que tiene este wallet.



Pulsa aceptar.

## Parte propietario de parcela


Para dirigirse al apartado de gestión de parcelas y contratación de servicios de fumigación, pulsar en el menú “Clientes (parcelas)”.

[Compra/Venta de Tokens FDT \(ERC-20\)](#) [Clientes \(parcelas\)](#) [Empresas \(drones\)](#) [LinkedIn](#) [GitHub](#)

### Area de clientes

Contrato de las parcelas: 0x847Ffd8fA70645B8430ae6676071f8693e23fD8b  
Contrato de los trabajos: 0xF00035bF13D8Ef1880CA5432bf7aDcA61c9D8e0f  
Contrato del token: 0x9e47E1319A4CaC45fddfd703131562703a6afec4

### Alta de parcelas y alquiler de drones



Registro de una parcela

Wallet propietario: 0x36097e8A6005120891aea8Ef80A5494C1722f751

Pesticida A

REGISTRAR PARCELA


Solicitar el servicio de fumigación

Wallet propietario de parcela:  
Altura máxima de vuelo permitida: 0  
Altura mínima de vuelo permitida: 0  
Pesticida de la parcela:

No hay ninguna parcela seleccionada

Como realizar alta de una nueva parcela

En el apartado “Registro de una parcela” introducir los datos necesarios: nombre de parcela, altura mínima de vuelo permitida, altura máxima de vuelo permitida, tipo de pesticida aceptado (todos son obligatorios). No se revisa correcta introducción de datos en la parte web, al no ser objetivo de este curso.

 Registro de una parcela

Wallet propietario: 0x36097e8A6005120891aea8Ef80A5494C1722f751

REGISTRAR PARCELA

Después pulsar botón “Registra parcela”.

Se le abrirá ventana del Metamask para confirmar el registro y cobrar el coste de publicación de datos en Blockchain.

MetaMask Notification

Localhost 8545

Account 5 → 0x847...fD8b

New address detected! Click here to add to your address book.

http://localhost:3000

0x847...fD8b CONTRACT INTERACTION

DETAILS DATA HEX

EDIT

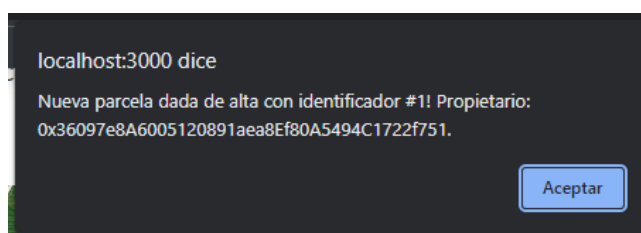
**Estimated gas fee** 0.00668378  
**0.006684 ETH**  
*Site suggested* **Max fee:** 0.00668378 ETH

**Total** 0.00668378  
**0.00668378 ETH**  
Amount + gas fee **Max amount:** 0.00668378 ETH

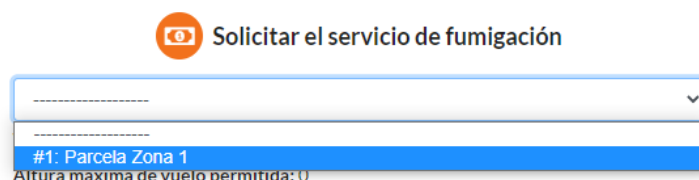
Reject Confirm

Pulsa “Confirm” para proceder con la operación de registro de parcela.

Seguido aparecerá una alerta en el navegador.

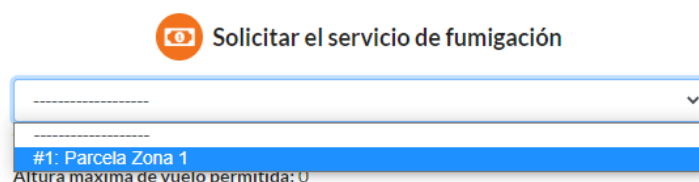


Pulsa aceptar y puede revisar que, en listado de sus parcelas, ahora hay una nueva parcela.



Como solicitar el servicio de fumigación

En el apartado “Solicitar el servicio de fumigación” tiene que seleccionar la parcela, a la que quiere solicitar el servicio.



Tras la selección de parcela, se mostrará los datos de esta parcela.



Después tiene que seleccionar uno de los drones aptos para esta parcela.

 Solicitar el servicio de fumigación

#1: Parcela Zona 1

Wallet propietario de parcela: 0x36097e8a6005120891aea8ef80a5494c1722f751  
 Altura máxima de vuelo permitida: 100  
 Altura mínima de vuelo permitida: 5  
 Pesticida de la parcela: Pesticida A


-----

#1: Dron 10L (10 FDT)

Altura máxima de vuelo:  
 Altura mínima de vuelo:  
 Total a pagar: FDT

No hay ningún dron seleccionado

Una vez seleccionado el dron adecuado, se mostrarán sus datos y el precio a pagar por el servicio de fumigación.

 Solicitar el servicio de fumigación

#1: Parcela Zona 1

Wallet propietario de parcela: 0x36097e8a6005120891aea8ef80a5494c1722f751  
 Altura máxima de vuelo permitida: 100  
 Altura mínima de vuelo permitida: 5  
 Pesticida de la parcela: Pesticida A

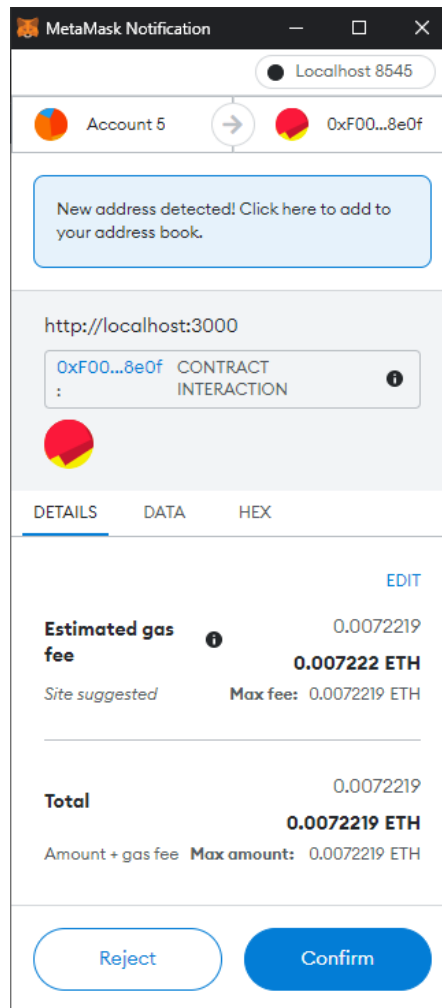
#1: Dron 10L (10 FDT)

Wallet propietario del dron: 0x12794F370758BcE4a4B503Ecbe07EE65237C9932  
 Altura máxima de vuelo: 500  
 Altura mínima de vuelo: 1  
 Total a pagar: 10 FDT

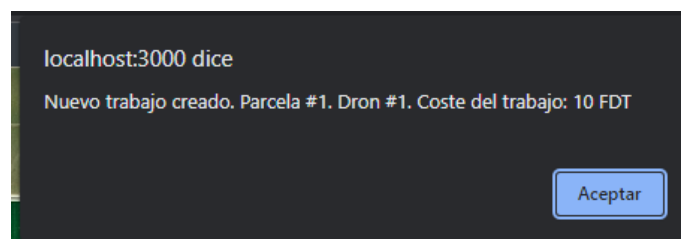
SOLICITAR SERVICIO

Después pulsar botón “Solicitar servicio”.

Se le abrirá ventana del Metamask para confirmar la solicitud, le cobrará el coste del servicio y cobrar el coste de publicación de datos en Blockchain.



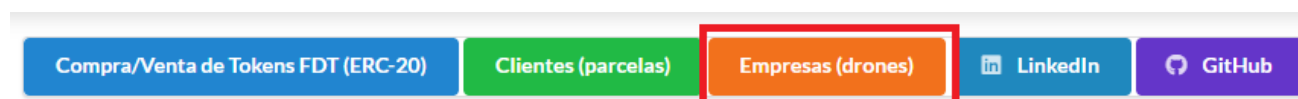
Pulsa “Confirm” para proceder con la operación de registro de parcela.  
Seguido aparecerá una alerta en el navegador.



Pulsa aceptar.

Parte propietario del dron (o empresa gestora)

Para dirigirse al apartado de gestión de drones y realización de servicios de fumigación, pulsar en el menú “Empresas (drones)”.



## Area de empresa

Contrato de los drones: 0xd5fA36e342541Dd721B172d3FBEE5cD2e0fE2Af  
Contrato de los trabajos: 0xF00035bF13D8Ef1880CA5432bf7aDcA61c9D8e0f  
Contrato del token: 0x9e47E1319A4CaC45fd703131562703a6afec4

### Alta de drones y ejecución de trabajos



#### Registro de un Dron

Wallet propietario del dron: 0x36097e8A6005120891aea8Ef80A5494C1722f751

Nombre del dron
Altura mínima de vuelo
Altura máxima de vuelo
Pesticida A
Precio por trabajo en tokens FDT
REGISTRAR DRON



#### Realizar tareas de fumigación

--

### Como realizar alta de nuevo dron

En el apartado “Registro de un Dron” introducir los datos necesarios: nombre del dron, altura mínima de vuelo, altura máxima de vuelo, tipo de pesticida que puede repartir, precio por el trabajo valorado en tokens FDT (todos son obligatorios). No se revisa correcta introducción de datos en la parte web, al no ser objetivo de este curso.

 **Registro de un Dron**

Wallet propietario del dron: 0x36097e8A6005120891aea8Ef80A5494C1722f751

Dron 10L
1
500
Pesticida A
10

REGISTRAR DRON

Después pulsar botón “Registra dron”.

Se le abrirá ventana del Metamask para confirmar el registro y cobrar el coste de publicación de datos en Blockchain.

MetaMask Notification

Localhost 8545

Account 6 → 0xd5f...E2Af

New address detected! Click here to add to your address book.

http://localhost:3000

0xd5f...E2Af : CONTRACT INTERACTION ⓘ

DETAILS DATA HEX

EDIT

**Estimated gas fee** ⓘ 0.0072959  
**0.007296 ETH**  
*Site suggested* **Max fee:** 0.0072959 ETH

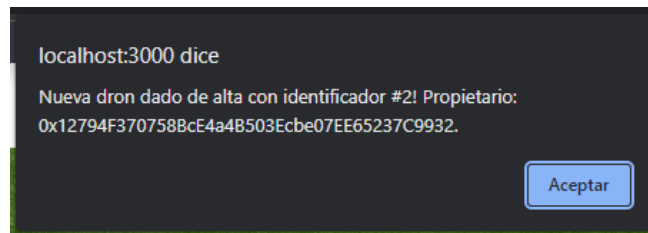
**Total** 0.0072959  
**0.0072959 ETH**  
*Amount + gas fee* **Max amount:** 0.0072959 ETH

Reject Confirm

Pulsa “Confirm” para proceder con la operación de registro del dron.

Seguido aparecerá una alerta en el navegador.

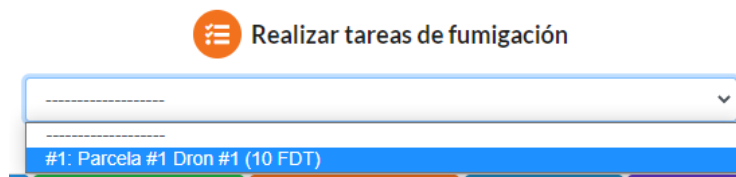




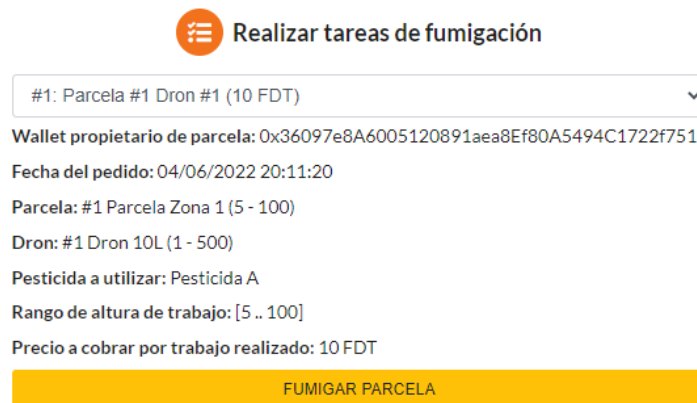
Pulsa aceptar.

### Como realizar tareas de fumigación

En el apartado “Realizar tareas de fumigación” tiene que seleccionar uno de los trabajos pendientes a realizar. Solo se visualizan los trabajos pendientes para el propietario del dron, si es un dron de otro propietario que tiene trabajos pendientes, en este listado no se verá.

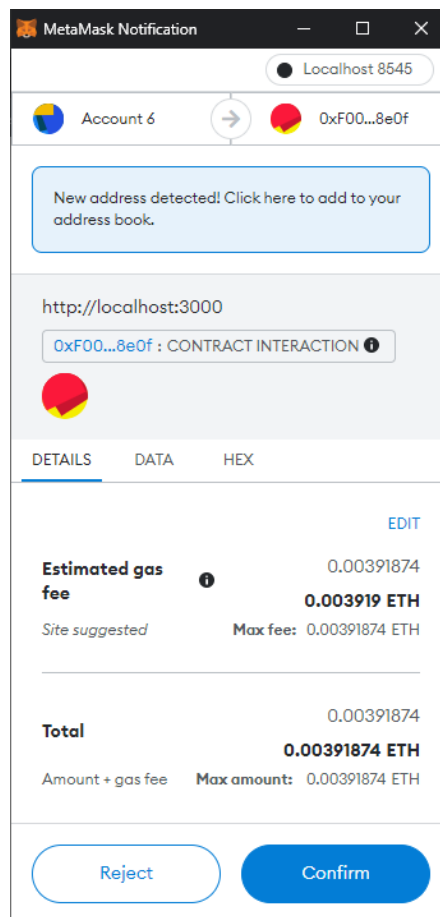


Una vez seleccionado el trabajo pendiente, se visualizarán todos los datos del trabajo a realizar.



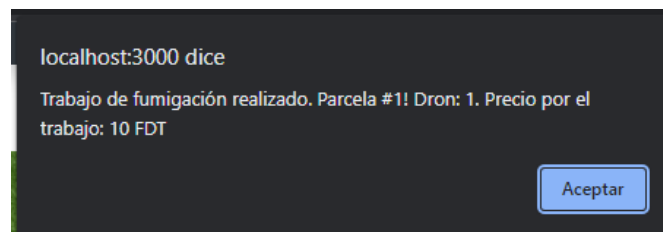
Para realizar el trabajo pulsar el botón “Fumigar parcela”.

Se le abrirá ventana del Metamask para confirmar la realización del trabajo, cobrará el coste de publicación de datos en Blockchain y le realizará una transferencia de los tokens FDT según consta en el trabajo a realizar.



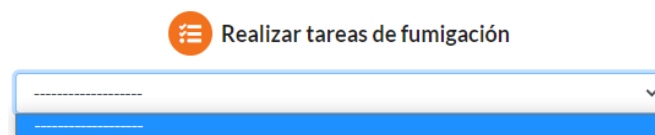
Pulsa “Confirm” para proceder con la operación de registro de parcela.

Seguido aparecerá una alerta en el navegador.

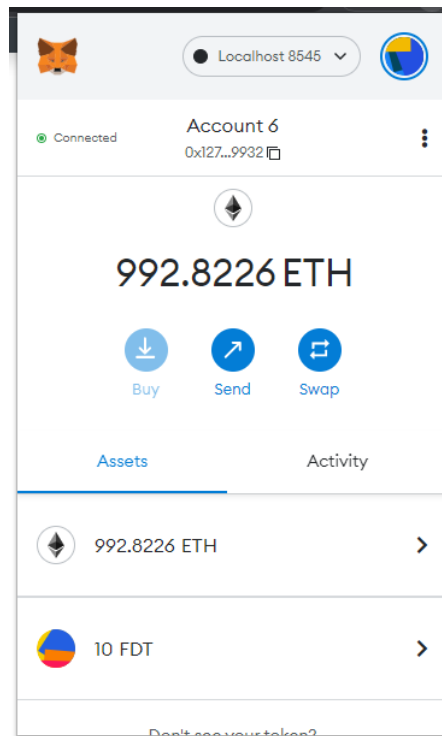


Pulsa aceptar.

Si revisa los trabajos pendientes a realizar, este trabajo ya no debería aparecer.



Si revisa su cuenta de wallet vera que se le ha sumado la cantidad de tokens indicada por la realización del trabajo de fumigación de parcela.



## Conclusiones

En este proyecto se han utilizado los conocimientos aprendidos durante el curso en UNIR, sobre criptografía, funcionamiento de las redes Blockchain, uso específico de los contratos basados en los estándares ERC721 y ERC20, también la parte de la librería web3 necesaria para la creación de la DApp. Aún que, con falta de tiempo y posibles mejoras, se ha procedido a realizar este proyecto aplicando todos los conocimientos, incluyendo la parte de testeo de los contratos que también se explica en la documentación. En general la idea principal del trabajo se ha plasmado en el código y en la memoria del documento presente.

Por desgracia se ha tenido problemas con el nodo de Alastria, aún que ya se ha realizado un trabajo sobre esta red en otra asignatura, probablemente por haber escogido la última versión del solc 0.8.15 o por la cantidad de ficheros a publicar, el nodo no logro aceptar la publicación de dichos contratos. Después mas tarde no dejaba ni conectarse a la maquina virtual de UNIR, posiblemente por colapso entre todos los alumnos que intentan a ultima hora desplegar los contratos.

Como alternativa se despliega en 4 redes alternativas desde la cuenta 0x12794F370758BcE4a4B503Ecbe07EE65237C9932:

1. Ganache (local):
  - a. FumigationDronToken: 0x9e47E1319A4CaC45fddfd703131562703a6afec4
  - b. Dron: 0xd5fA36e342541Dd721B172d3FBEE5cD2eE0fE2Af
  - c. Plot: 0x847Ffd8fA70645B8430ae6676071f8693e23fD8b
  - d. Works: 0xF00035bF13D8Ef1880CA5432bf7aDcA61c9D8e0f
2. Polygon (Matic TestNet):
  - a. FumigationDronToken: 0x216375c663bD531570D3736B4c780960b36B5239
  - b. Dron: 0x61bC65A7AC93D6e9002Ca2b20659aA8641CB2AD0
  - c. Plot: 0x2e9fdeEa6af43Dfce638b33b179b5787f5011bf8
  - d. Works: 0x5eBAaD6212872f0BBcF3Ae2CbA4ABe5b43621186
3. Binance Smart Chain (TestNet):
  - a. FumigationDronToken: 0x216375c663bD531570D3736B4c780960b36B5239
  - b. Dron: 0xdee3561f504Ca28472CC1a73Cf50ad0b3293484F
  - c. Plot: 0x61bC65A7AC93D6e9002Ca2b20659aA8641CB2AD0
  - d. Works: 0x2e9fdeEa6af43Dfce638b33b179b5787f5011bf8
4. Ethernet (Rinkeby):
  - a. FumigationDronToken: 0xdee3561f504Ca28472CC1a73Cf50ad0b3293484F
  - b. Dron: 0x216375c663bD531570D3736B4c780960b36B5239
  - c. Plot: 0x61bC65A7AC93D6e9002Ca2b20659aA8641CB2AD0
  - d. Works: 0x2e9fdeEa6af43Dfce638b33b179b5787f5011bf8