

# TRABAJO FINAL DE EXPERTO

---

CASO GENERAL DRONES Y PARCELAS

PABLO COSIO MOLLEDA

## Contenido

<b>1.</b>	<b>Justificación del uso de la tecnología blockchain .....</b>	<b>2</b>
<b>2.</b>	<b>Análisis y modelo del sistema propuesto .....</b>	<b>3</b>
<b>2.1.</b>	<b>Arquitectura de la solución .....</b>	<b>3</b>
<b>2.2.</b>	<b>Contratos ERC20 y ERC721 .....</b>	<b>5</b>
<b>2.3.</b>	<b>Modelado de la solución .....</b>	<b>6</b>
<b>2.4.</b>	<b>Diagramas de secuencia .....</b>	<b>7</b>
<b>2.5.</b>	<b>Problemas encontrados .....</b>	<b>11</b>
<b>3.</b>	<b>Descripción del entorno de desarrollo utilizado .....</b>	<b>13</b>
<b>4.</b>	<b>Instrucciones de despliegue .....</b>	<b>14</b>
<b>4.1.</b>	<b>Diagrama de despliegue .....</b>	<b>14</b>
<b>4.2.</b>	<b>Despliegue en testnet Alastria Telsius .....</b>	<b>14</b>
<b>4.3.</b>	<b>Despliegue en red T de Alastria (PROD) .....</b>	<b>26</b>
<b>5.</b>	<b>Testing de la solución .....</b>	<b>35</b>
<b>5.1.</b>	<b>Test unitarios con truffle .....</b>	<b>35</b>
<b>5.2.</b>	<b>Casos de uso .....</b>	<b>36</b>
<b>6.</b>	<b>Manual de usuario .....</b>	<b>38</b>
<b>6.1.</b>	<b>Registro de Empresa y de drones .....</b>	<b>38</b>
<b>6.2.</b>	<b>Registro de propietarios y parcelas .....</b>	<b>46</b>
<b>6.3.</b>	<b>Menú para los Propietarios .....</b>	<b>49</b>
<b>6.4.</b>	<b>Menú empresa para asignación de trabajos a los drones .....</b>	<b>53</b>
<b>7.</b>	<b>Conclusiones .....</b>	<b>56</b>

## 1. Justificación del uso de la tecnología blockchain

En mi opinión desarrollar este tipo de soluciones utilizando la blockchain es muy interesante y aporta valor en cuanto a:

- Actua como un custodio de los tokens (drones, parcelas y tokens de pago)
- Cualquier operación de modificación de estado de estos tokens es auditado y registrado en la blockchain. Por lo que ante cualquier auditoria tenemos en un solo sitio todas las hash de las transacciones que identifican estas operaciones de modificación
- Evitan intermediarios. En un caso como el del presente trabajo en el que intervienen empresas propietarias de drones y los propietarios de parcelas, la parte intermedia se elimina, por tanto la rapidez y eficiencia desde que el propietario de parcela solicita el trabajo hasta que se comienza y finaliza es muy grande. Por este motivo blockchain tiene muchos casos de uso en el mundo de la IoT.

Además este trabajo se realiza sobre Alastria, que tiene la particularidad de que las transacciones no tienen fee (no consumen gas) y por tanto no tiene moneda como tal (ETHER en el caso de Ethereum). Esto hace que según el caso de uso, sea más conveniente implementarlo con una red privada (consorcio) como el caso de Alastria o pública como Ethereum. Factores como la privacidad de los datos y la robustez de la solución las considero clave. Por ejemplo en Ethereum los Smart contracts que desplegamos considero que serán más estables porque es una infraestructura descentralizada en la que todos los participantes les interesa que "no se pare" y funcione bien por las comisiones que se llevan a la hora de minar las transacciones. Sin embargo en un consorcio el número de nodos es mucho menor y gestionado por los participantes.

Uno de los puntos fuertes de este proyecto es el uso de la tokenización para:

- Representar el token ERC20 fungible, cuyos tokens servirán de pago por contratar el servicio de fumigación.
- Representar el token ERC721 no fungible. Estos tokens representarán a Drones y Parcelas y por las características del propio token hacen que cada uno sea único, fundamental por su id, owner y metadata asociados.

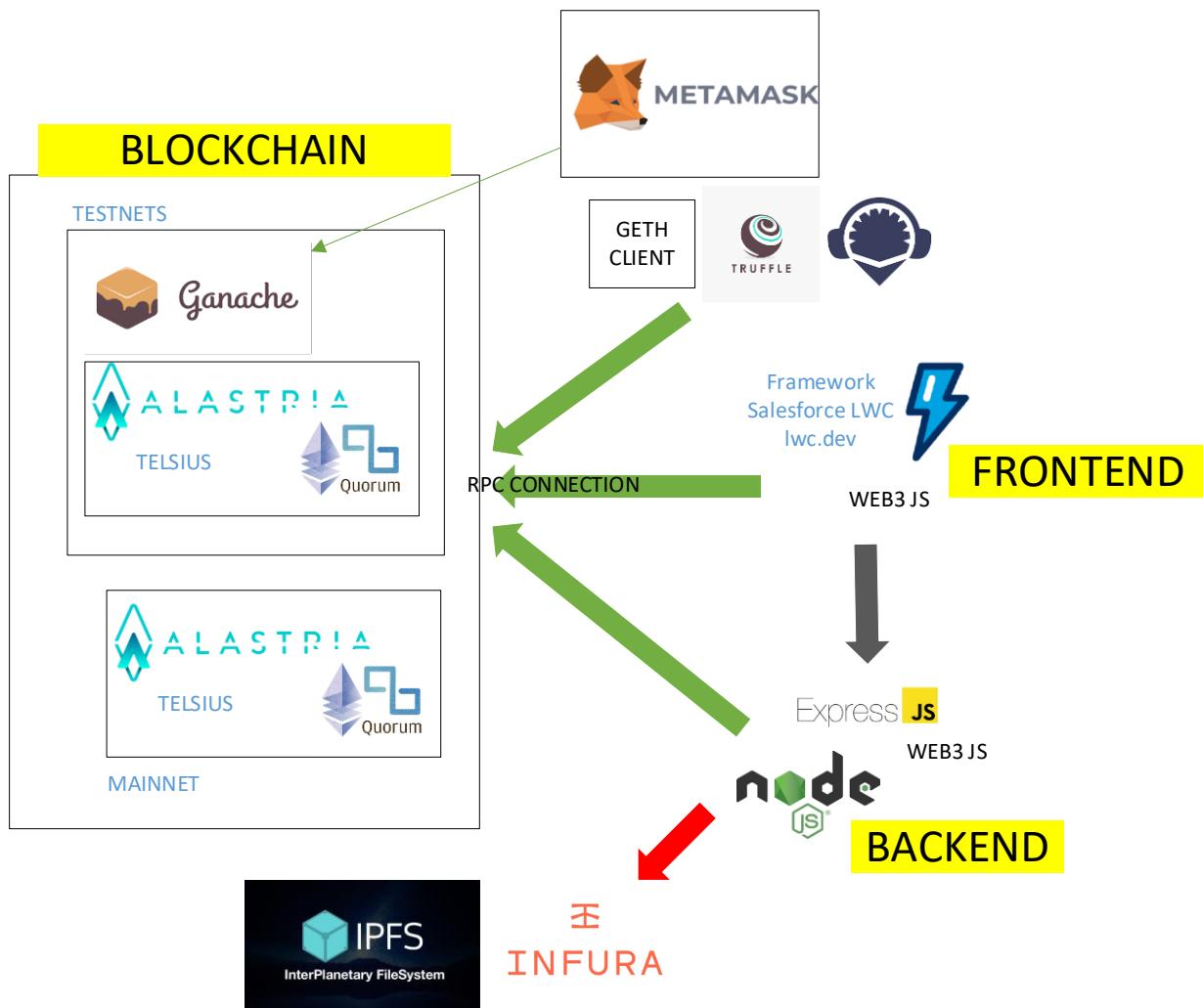
## 2. Análisis y modelo del sistema propuesto

### 2.1. Arquitectura de la solución

En solución se han utilizado diferentes soluciones y todas ellas integradas las cuales componen toda la plataforma web que permite:

- Empresas registrar sus drones, ver el listado de drones registrados y asignarles trabajos de fumigación de parcelas.
- Propietarios de parcelas. Registro de parcelas, listado, ver listado de drones disponibles y compatibles que pueden realizar la fumigación en la parcela específica.

A grandes rasgos el siguiente esquema resume el stack tecnológico utilizado:



Se ha utilizado los siguientes componentes:

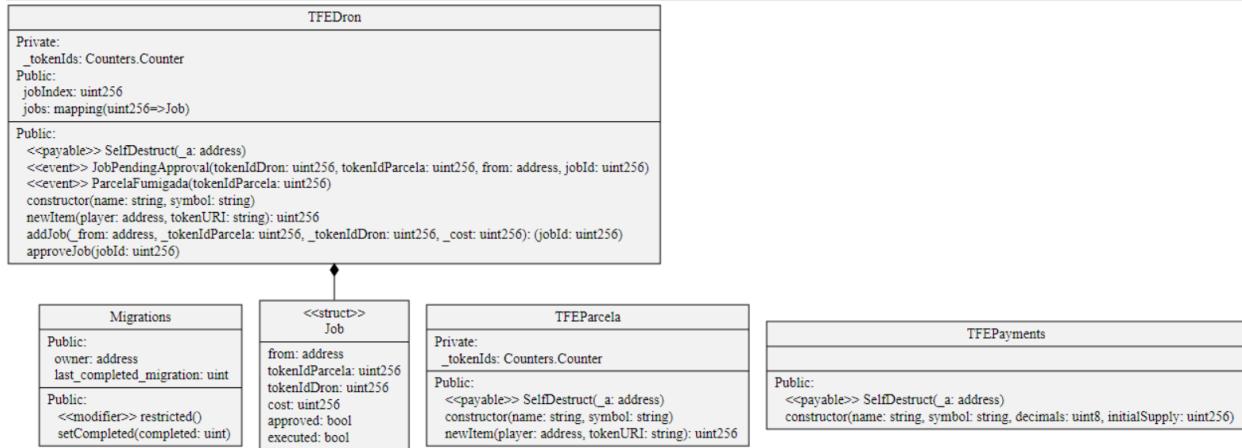
- **Frontend.** Se ha utilizado el framework de Salesforce Open Source llamado lwc (lightning web components). Interactúa tanto con el Backend como con la propia blockchain mediante web3. Es una forma de programar una SPA (Single Page Application) basada en componentes que se renderizan o no en función de variables y condiciones.
- **Backend** nodejs. Se ha utilizado express para levantar 3 endpoints que consume el frontend para:
  - Enviar el json con la metadata del dron/parcela a IPFS (Gateway Infura)
  - Obtener json de IPFS
  - Validar/chequear si un dron tiene las características válidas de vuelo (altura máxima y mínima) y tipo de pesticida. Se conecta a IPFS recupera ambos jsons y hace las validaciones
  - Transferencia inicial de tokens. En el momento del registro de los propietarios de parcelas, se hace una transferencia de tokens desde owner del contrato ERC20 hacia el propietario para que pueda gastarlos después en la contratación de Drones para fumigar parcelas.

Estos 3 endpoints no se han metido dentro de la capa frontend por seguridad por dos motivos:

- Conexión a IPFS a través del nodo de Infura. En este caso no hay autenticación pero en entornos productivos necesita realizar una autenticación.
- Transferencia de tokens. En este punto hay que desbloquear la cuenta que despliega el contrato ERC20. Para ello se necesita la password que protege la cuenta y por tanto no debería realizarse en la capa frontend, ya que al igual que el primer punto, cualquier persona debugueando en el navegador podría llegar a saber estas claves de autenticación.
- **Truffle.** Realiza 3 funciones:
  - Despliegue de los contratos. Detalle del despliegue en el apartado 4
  - Una vez desplegados genera el fichero json dentro de la carpeta build. Esto es útil tanto para el frontend como el backend ya que de ahí obtienen la dirección del Smart contract y el ABI, necesarios para hacer la conexión con web3.
  - Ejecución de los test. Detalle de los mismo en el apartado 5.1.
- **Cliente geth** para poder conectarnos al nodo y ver las cuentas, desbloquear y crear nuevas.
- **Ganache y Metamask.** Se ha utilizado la suite de ganache para poder desarrollar en local. En ganache los envíos de transacciones requieren de gas como fee. Como las nuevas direcciones que se crean cuando se registran nuevas empresas o propietarios no tienen ETHER, se ha utilizado metamask para hacer el transfer desde cuentas que por defecto crea metamask (con 100 ethers) a estas nuevas cuentas. De esta forma se ha podido probar en local con ganache todo el flujo.
- **IPFS.** Se ha utilizado el Gateway de Infura para almacenar en IPFS los JSON que contienen la metadata de parcelas y drones. IPFS genera un hash que es una especie de puntero para poder recuperar después del documento. Este hash se ha almacenado dentro del contrato ERC721.
- **Alastria.** Se ha utilizado tanto la testnet como la propia mainnet de Alastria Telsius. La tesnet en forma de máquina virtual y la mainnet atacando al nodo de Unir.
- **Remix.** Para poder probar los contratos desplegados, poder ir viendo el estado de las variables y en definitiva ir haciendo un poco de debug según iba interactuando con los contratos.

## 2.2. Contratos ERC20 y ERC721

Diagrama generado con sol2uml:



Como se puede observar faltan muchos de las propiedades y funciones de los token ERC20 y ERC721. Me he basado en los que ya provee OpenZepellin en su versión 2.3.0 que se corresponden con la versión de solidity 0.5.0. He escogido esta versión porque es la máxima versión que he visto haciendo pruebas que me permitía hacer un despliegue y uso de los contratos sin aparentes errores. Sin embargo el uso de la versión 0.5.0 puede que sea el causante de los errores que comento más adelante en el punto 2.3. que me surgieron a posteriori en fases avanzadas ya del desarrollo, por lo que me resultaba ya complicado buscar una versión inferior que fuese compatible sin errores. Además estos errores puede que sean debido a limitaciones de la propia red de Alastria.

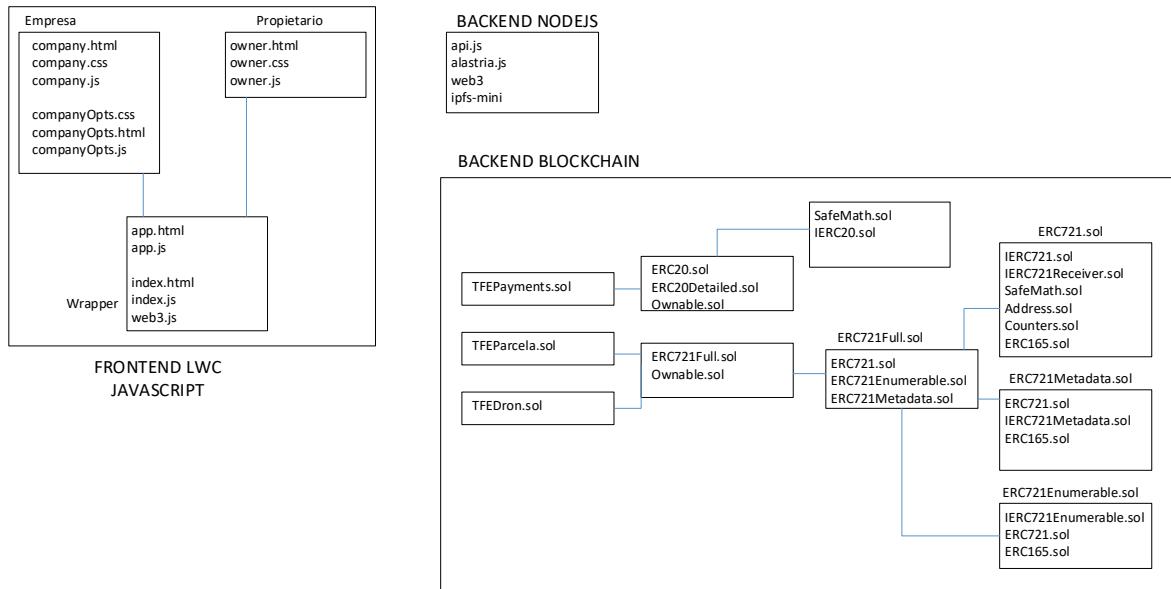
A continuación, realizo una breve explicación de los contratos (el código fuente se puede ver con los documentos de la entrega):

- *Contrato ERC721 para los drones.* Dentro del contrato se ha realizado un añadido para poder registrar todas los trabajos pendientes y ya realizados por los drones. Se ha creado un Struct para que todos los trabajos queden almacenados en la cadena de bloques. Así mismo se ha generado también la función para poder aprobar o asignar por parte de la empresa el trabajo a un dron.
- *Contrato ERC721 para las parcelas.* No se ha realizado prácticamente ningun añadido sobre la base de OpenZepellin
- *Contrato ERC20 para los pagos.* Se utiliza para los pagos por parte del propietario de la parcela a la empresa del dron. Como se ha comentado anteriormente durante el registro del propietario se le transfieren X tokens de la cuenta `owner` que desplegó el contrato a la nueva cuenta del propietario de la parcela creada. Posteriormente durante la contratación del dron es donde se realiza la transferencia de tokens como resultado del servicio prestado por el dron.

En todos ellos se ha añadido la función SelfDestruct la cuál no venia de base en las implementaciones de OpenZepellin.

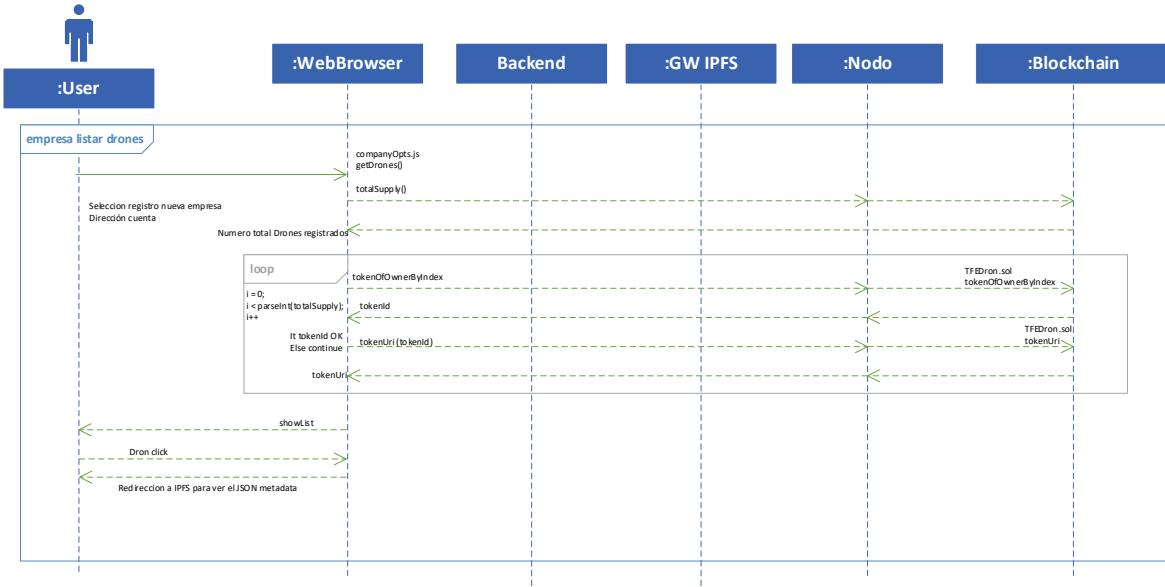
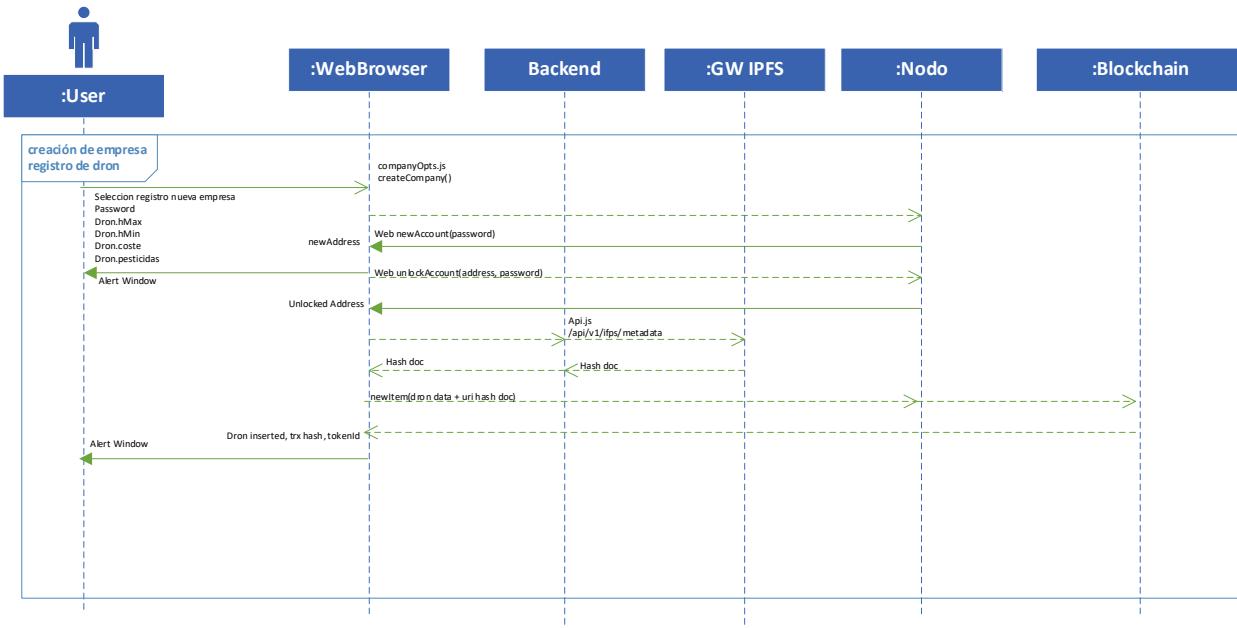
### 2.3. Modelado de la solución

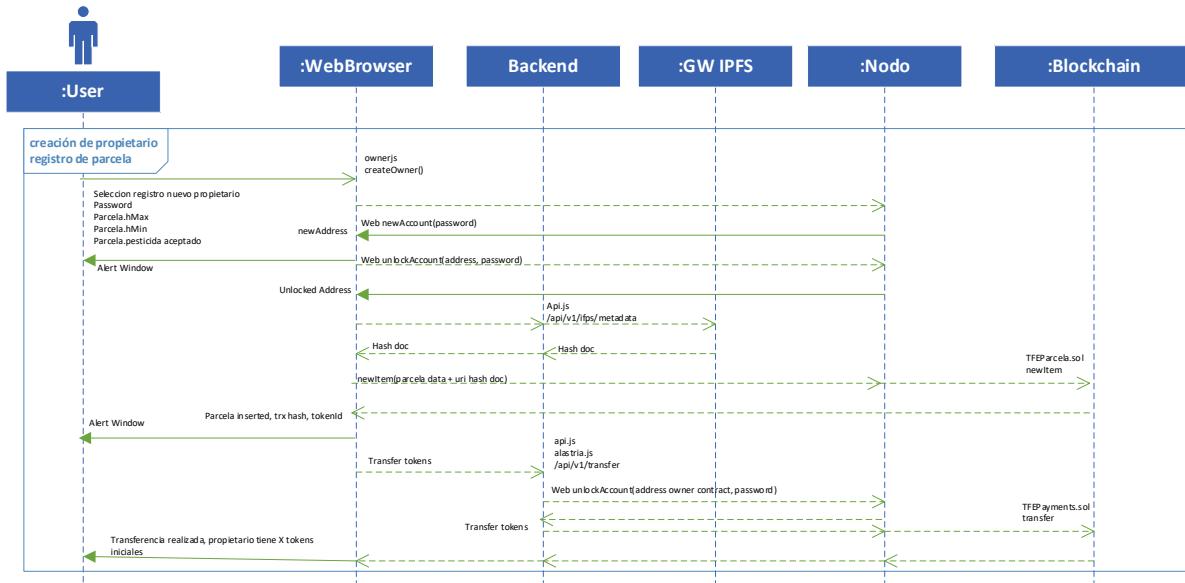
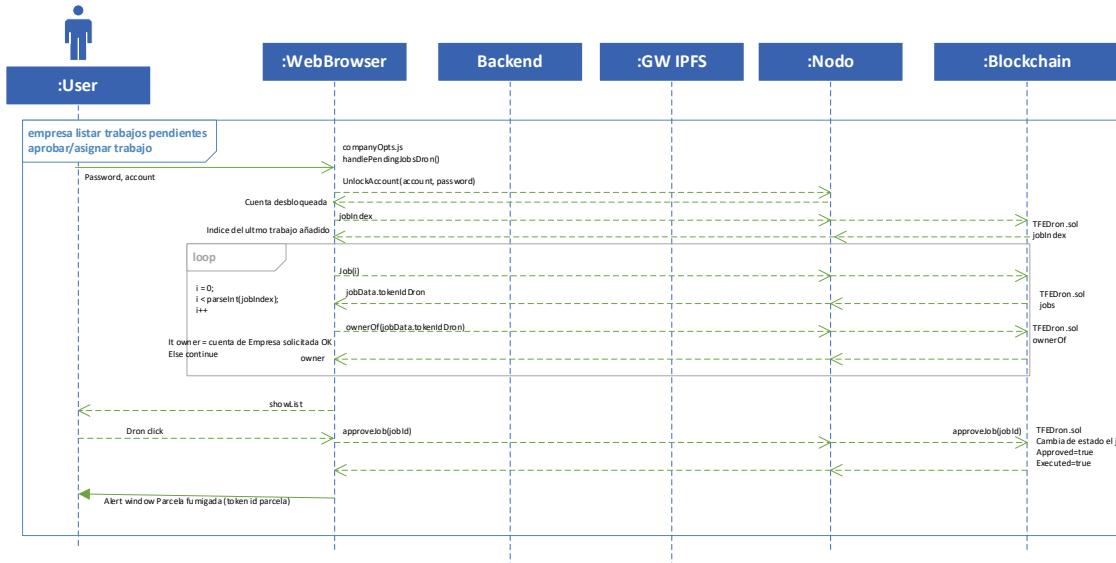
En el siguiente esquema se refleja el modelo la solución:

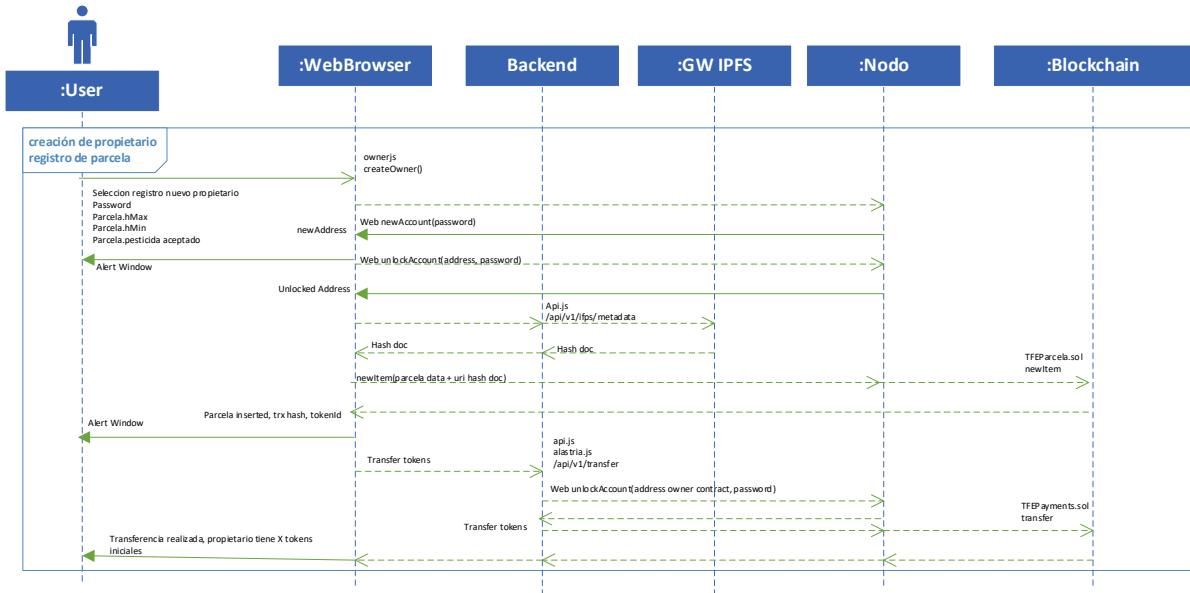


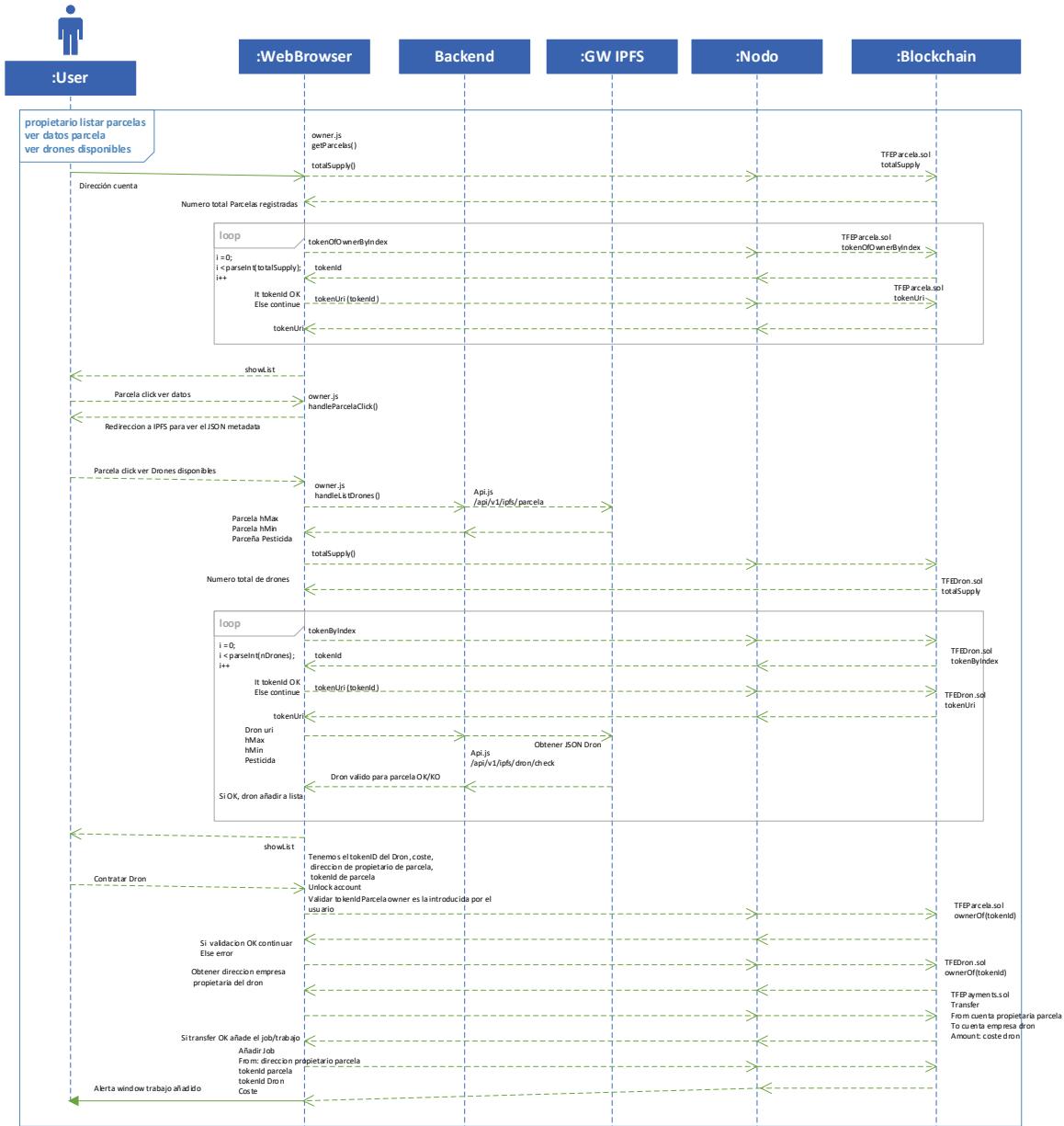
## 2.4. Diagramas de secuencia

A continuación, se detallan los principales interacciones de los componentes:









## 2.5. Problemas encontrados

No podía llamar a contratos desde el propio contrato, problema de gas aparante, y hacía que la función no me diese error, pero no se ejecutaba. Esto me ha provocado varios problemas que intento explicar a continuación:

```
function addJob(address _from, uint256 _tokenIdParcela, uint256 _tokenIdDron, uint256 _cost) public returns (uint256 jobId){
    //require(contractParcela.ownerOf(_tokenIdParcela) == msg.sender, "Only owner of the parcela can request a job");
    require(_from == msg.sender, "Only owner of the parcela can request a job");
    jobs[jobIndex].from = _from;
    jobs[jobIndex].tokenIdParcela = _tokenIdParcela;
    jobs[jobIndex].tokenIdDron = _tokenIdDron;
    jobs[jobIndex].cost = _cost;
    jobs[jobIndex].approved = false;
    jobs[jobIndex].executed = false;
    jobIndex++;
    emit JobPendingApproval(_tokenIdDron, _tokenIdParcela, _from, (jobIndex-1));
    return jobIndex-1;
}

function approveJob(uint256 jobId) public {
    uint256 tokenId = jobs[jobId].tokenIdDron;
    require(ownerOf(tokenId) == msg.sender, "Only owner of the Dron can approve jobs");
    //require(contractPayments.transferFrom(jobs[jobId].from, ownerOf(tokenId), jobs[jobId].cost), "TransferFromFailed");
    jobs[jobId].approved = true;
    emit ParcelaFumigada(jobs[jobId].tokenIdParcela);
    jobs[jobId].executed = true;
}
```

- La función addJob no podía verificar que el owner de la parcela era realmente el que estaba enviando la transacción. Esta lógica la he tenido que sacar a la parte web por web3js, lo cuál lo hace mucho más inseguro.
- La función approveJob, es la que se encarga de hacer el transfer entre el propietario de la parcela y la empresa del dron. Al no poder realizarlo, he realizado esta lógica del pago justo en el momento que el propietario contrata el dron, en la parte web con web3js.

- No se puede utilizar oráculos para validar la metadata de los drones y parcelas. Por ejemplo:
  - el coste a la hora de contratar un dron, para validar que el coste que se pone es el que está en el json de IPFS. Todo esto en su defecto, lo he sacado a la parte web, en este caso en el backend haciendo las llamadas oportunas a ipfs y recuperando la información.
  - Para validar que las características del dron (altura max y min y pesticida) se ajusta a los parámetros de la parcela.
- Me he encontrado con otro problema en la función addJob. Ya que no podía hacer de forma sencilla ninguna de las validaciones anteriores, decidí poner un control para que el Uri/hash de IPFS del dron, fuese el mismo que el que realmente tenia el dron asociado al tokenId de entrada. La función hacer esta comprobación era:

```
require(keccak256(abi.encodePacked(this.tokenURI(_tokenIdDron))) == keccak256(abi
.encodePacked(_tokenURI)), "Token uri doesn't match to the related to tokenId");
```

Al llevarlo a la tesnet de la Alastria Telsius me ha dado error, igual que en los anteriores:

**Returned error: gas required exceeds allowance or always failing transaction**

Como conclusión parece que hay algún error a nivel de la tesnet que me impide hacer cosas complicadas por alguna gestión interna del gas. Supongo que sea parametrización de la testnet el aumentar este gas, pero no he encontrado done. Por tanto al igual que con el resto, todas estas validaciones al no poderlas hacerlas onchain, lo hago fuera (frontend-backend).

Todos estos errores se ha verificado que tanto en las tesnet de Ganache como Rinkeby funcionaban correctamente.

### 3. Descripción del entorno de desarrollo utilizado

Para el desarrollo del proyecto se ha utilizado fundamental Visual Studio, desde el cual tengo la visión de todo el código fuente del proyecto que contiene basicamente el código Javascript y Solidity que digamos son los dos lenguajes de programación utilizados.

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows the project structure under "WEB\_PROJECT".
  - FRONTEND:** Contains ".sfdx", "build", "contracts", "migrations", "node\_modules", "scripts", "src/client", and "src/server".
  - BACKEND:** Contains ".gitkeep", "TestDron.js", ".eslintignore", ".eslintrc.json", ".gitignore", ".prettierrc", ".prettierc", "jest.config.js", "lwc-services.config.js", "lwc.config.json", "package-lock.json", "package.json", and "README.md".
  - TRUFFLE:** Contains "truffle-config.js".
- EDITOR:** Displays the Solidity code for the TFEPayments contract.

```
pragma solidity 0.5.0;

import "@openzeppelin/contracts/token/ERC20/ERC20Detailed.sol";
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
import "@openzeppelin/contracts/ownership/Owningable.sol";

contract TFEPayments is ERC20, ERC20Detailed, Owningable {
    constructor(string memory name, string memory symbol, uint8 decimals, uint256 initialSupply) public {
        _mint(msg.sender, initialSupply);
    }

    function SelfDestruct(address payable _a) public onlyOwner payable {
        selfdestruct(_a);
    }
}
```
- TERMINAL:** Shows the command prompt output:

```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

D:\Study\UNIR - Experto Blockchain\TFE\web_project>
```

El resto de ficheros son los propios de un proyecto de nodejs, el fichero package.json contiene todas las dependencia gestionadas por npm, y la propia carpeta node\_modules que contiene todos los paquetes y dependencias indicados en el package.json pero ya instalados para poder arrancar el proyecto.

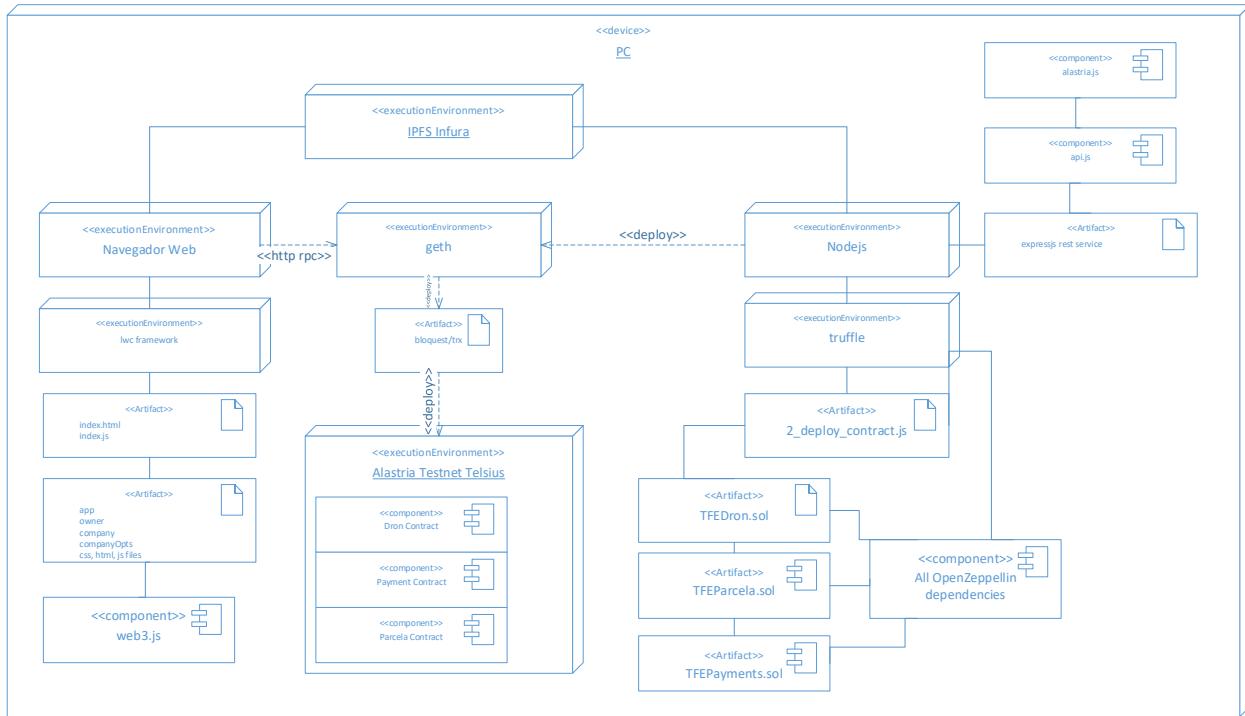
La capa backend está en src/server

La capa frontend está en src/client

En el punto 2.1. se han comentado también el resto de tecnologías y frameworks utilizados en el proyecto.

## 4. Instrucciones de despliegue

### 4.1. Diagrama de despliegue

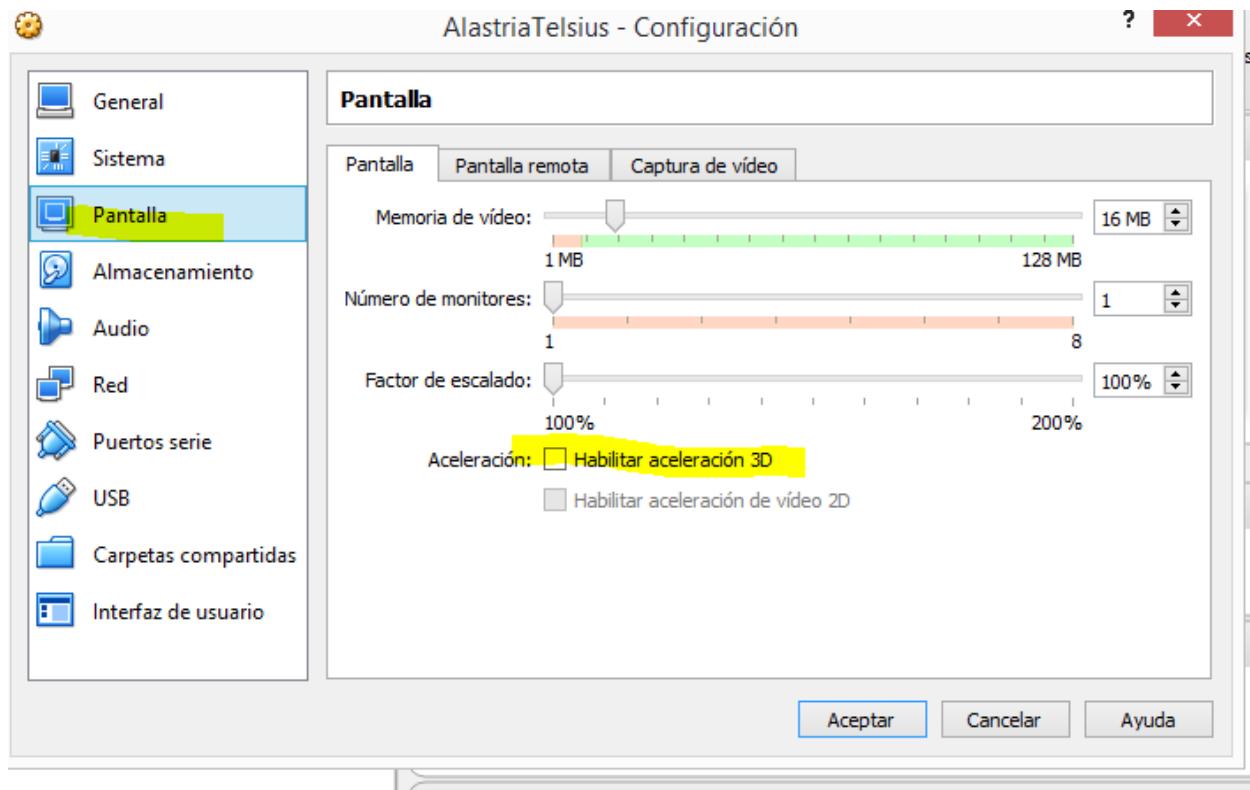


### 4.2. Despliegue en testnet Alastria Telsius

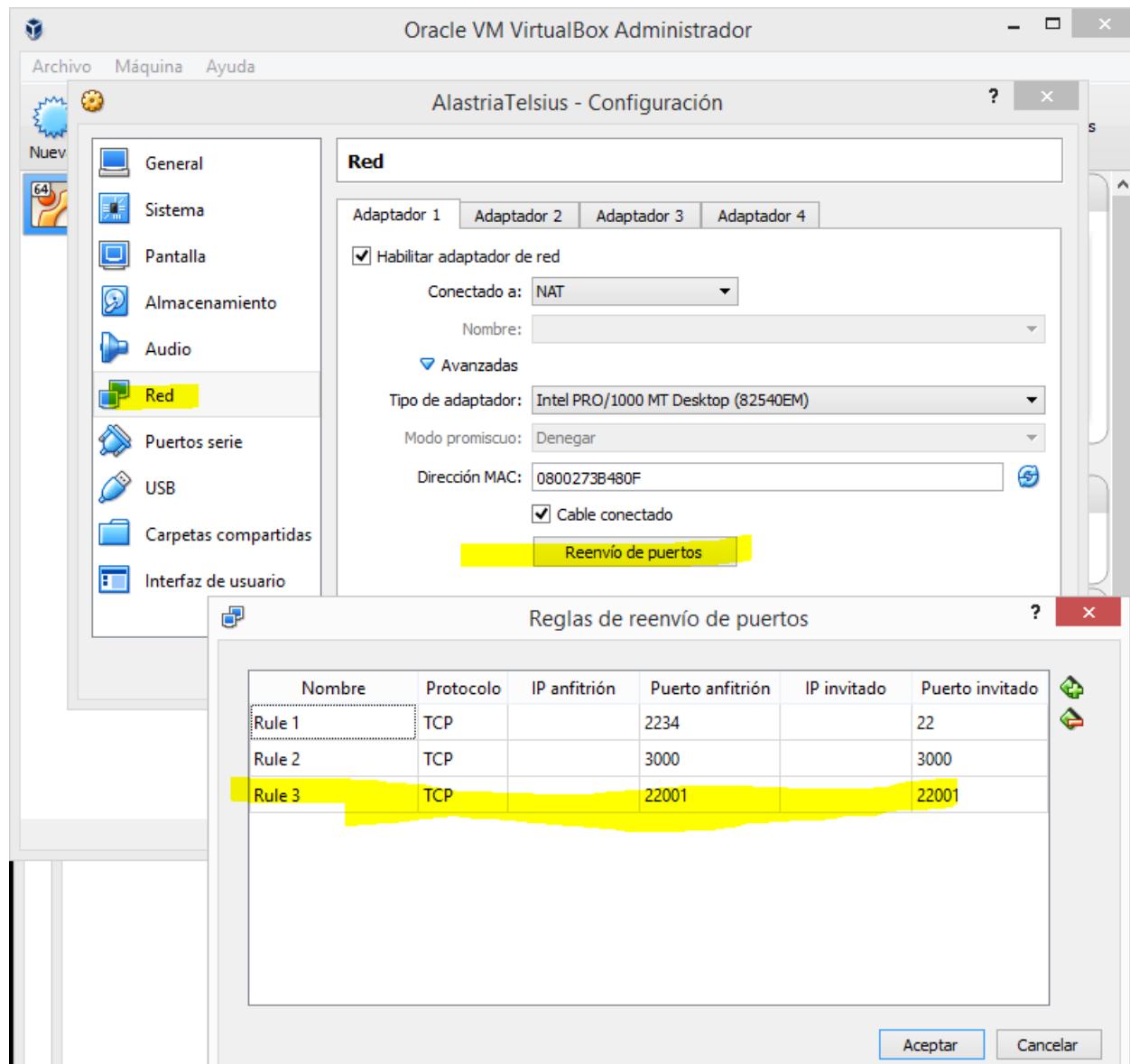
En este apartado detallo las instrucciones de despliegue para uso de la testnet de Alastria Telsius (VM) desde la VDI de Unir.

Primero. Me voy a la VDI de Unir. Abrimos la VM de Virtual BOX de Alastria Telsius. Ahí realizo dos configuraciones:

Pantalla. Desclickar “Habilitar aceleración 3D”. Si no me da error



Red. Para poder lanzar la conexión con el nodo de general1 que se ejecuta dentro la VM, hemos de realizar la configuración de red para que desde la VDI de forma cómoda accedamos al nodo.



Se inicia la VM. En ella tenemos que configurar el cors de los nodos para permitir poder conectarnos desde fuera dela VM, en este caso el Host, correspondiente a la VDI.

Screenshot of a Linux desktop environment showing a file manager window and a terminal window.

The file manager window (Nautilus) shows a directory structure under "/bin". A red circle highlights the "start\_node.sh" file.

```

bin
└── testnet
    └── bin
        ├── bootstrap.sh
        ├── build_faulty_nodes.sh
        ├── clean_env.sh
        ├── clean_ethstats.sh
        ├── config_network.sh
        ├── new_identity.sh
        ├── nodekey
        ├── start_faulty_node.sh
        ├── start_network.sh
        ├── start_node.sh
        └── stop_ethstats.sh

```

The terminal window (gedit) displays the content of the "start\_node.sh" script:

```

#!/bin/bash
# This script starts a node in the testnet

# Set the port based on the node name
if [[ "$NODE_NAME" == "general4" ]]; then
    PUERTO=4
elif [[ "$NODE_NAME" == "validator1" ]]; then
    PUERTO=5
elif [[ "$NODE_NAME" == "validator2" ]]; then
    PUERTO=6
else
    PUERTO=7
fi

# Set other network parameters
OTHER_NODES=$(cat ${PWD}/identities/CONSTELLATION_NODES)
GLOBAL_ARGS="--networkId $NETID --identity $IDENTITY --rpc --rpccaddr 0.0.0.0 --rpccapi \
admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,istanbul --rpccorsdomain=* \
--rpccport 2200$PUERTO --port 2100$PUERTO --targetgaslimit 18446744073709551615 --ethstats \
$IDENTITY:bb98a0b6442386d0cdf8a31b267892c1@$ETH_STATS_IP:3000"
CONSTELLATION_PORT="900$PUERTO"

if [ "$NODE_NAME" == "main" -o "$NODE_NAME" == "validator1" -o "$NODE_NAME" == "validator2" ]; then
    nohup geth --datadir "${PWD}/network/"$NODE_NAME" ${GLOBAL_ARGS} --mine --minerthreads 1 --syncmode "full" 2>> "${PWD}/logs/quorum_"$NODE_NAME"_"${_TIME}" .log &
else
    # TODO: Add every regular node for the constellation communication
    generate_conf "${NODE_IP}" "${CONSTELLATION_PORT}" "$OTHER_NODES" "${PWD}/network/"$NODE_NAME" > "${PWD}/network/"$NODE_NAME"/constellation/constellation.conf
    PWD="${pwd}"
    nohup constellation-node "${PWD}/network/"$NODE_NAME"/constellation/constellation.conf >> "${PWD}/logs/constellation_"$NODE_NAME"_"${_TIME}" .log &
    check_port ${CONSTELLATION_PORT}
    nohup env PRIVATE_CONFIG="${PWD}/network/"$NODE_NAME"/constellation/constellation.conf geth --datadir "${PWD}/network/"$NODE_NAME" --debug ${GLOBAL_ARGS} 2>> "${PWD}/logs/quorum_"$NODE_NAME"_"${_TIME}" .log &
fi

echo "Verify if ${PWD}/logs/ have new files."

set +u
set +e

```

Arrancar la testnet de Alastria Telsius: `sudo ./bin/start_network.sh clean 1 2`

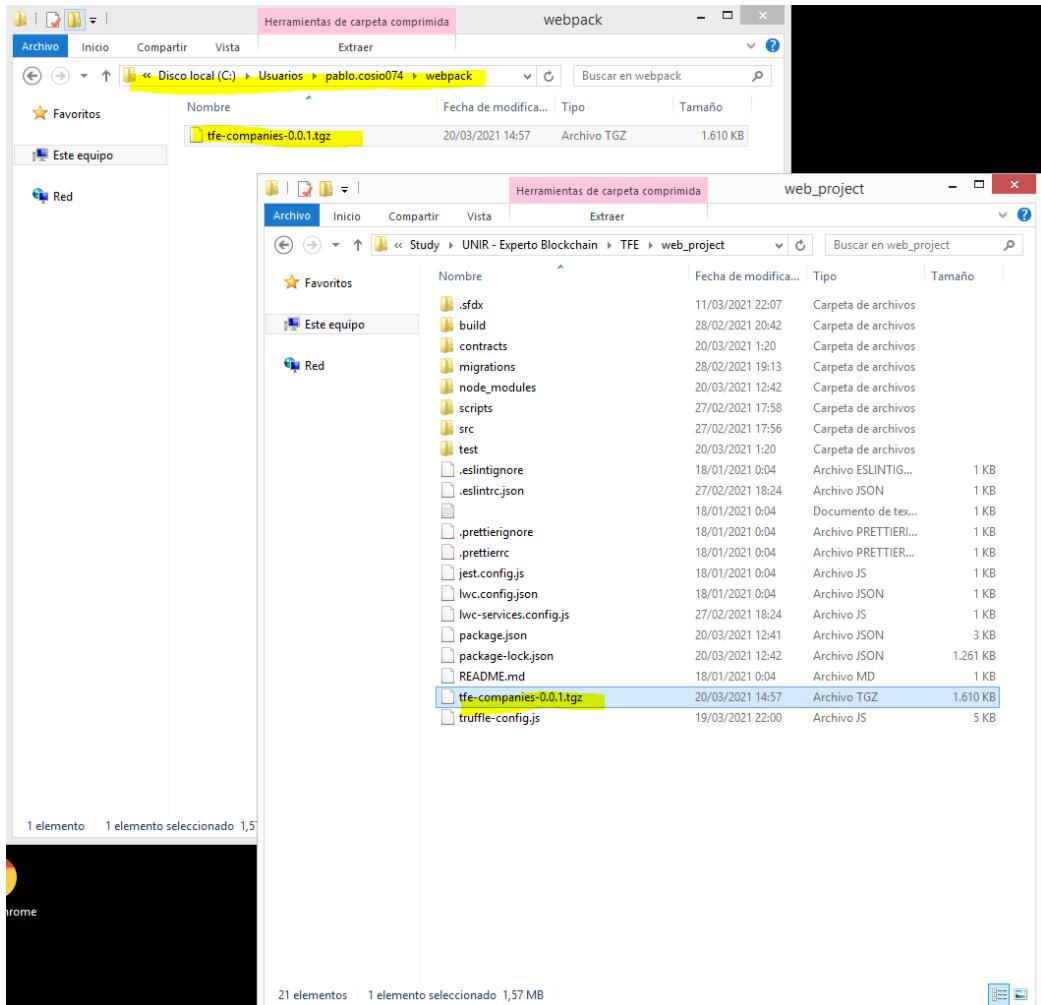
Verificar que geth está levantado para que podamos acceder con el cliente:

```
ubuntu@testenv:~$ ps -ef | grep geth
root      2802  1744  2 14:46 pts/4    00:00:01 geth --datadir /home/ubuntu/test-environment/infrastructure/testnet/network/main --networkid 9535753591 --identity main --rpc --rpccaddr 0.0.0.0 --rpccapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,isbanl --rpccorsdomain=* --rpccport 22000 --port 21000 --targetgaslimit 18446744073709551615 --ethstats main:bb98a0b6442386d0cdf8a31b267892c1@127.0.0.1:3000
root      3177  1744  2 14:47 pts/4    00:00:00 geth --datadir /home/ubuntu/test-environment/infrastructure/testnet/network/general1 --debug --networkid 9535753591 --identity general1 --rpc --rpccaddr 0.0.0.0 --rpccapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,istanbul --rpccorsdomain=* --rpccport 22001 --port 21001 --targetgaslimit 18446744073709551615 --ethstats general1:bb98a0b6442386d0cdf8a31b267892c1@127.0.0.1:3000
root      3394  1744  3 14:47 pts/4    00:00:00 geth --datadir /home/ubuntu/test-environment/infrastructure/testnet/network/general2 --debug --networkid 9535753591 --identity general2 --rpc --rpccaddr 0.0.0.0 --rpccapi admin,db,eth,debug,miner,net,shh,txpool,personal,web3,quorum,istanbul --rpccorsdomain=* --rpccport 22002 --port 21002 --targetgaslimit 18446744073709551615 --ethstats general2:bb98a0b6442386d0cdf8a31b267892c1@127.0.0.1:3000
ubuntu    3429  3413  0 14:47 pts/17   00:00:00 grep --color=auto geth
ubuntu@testenv:~$
```

Como se puede observar se levantan dos nodos regulares y uno validador que también hace de Bootstrap. Se puede observar que el parámetro rpccorsdomain se ha propagado correctamente.

Ya tenemos la VM lista con la Tesnet corriendo. Ahora nos vamos a la VDI para empezar a lanzar todo el proceso.

Me cogo el tar.gz con la aplicación web empaquetada.



Descomprimo el fichero y una vez descomprimido ya tengo todo el código fuente:

package

Favoritos	Nombre	Fecha de modifica...	Tipo	Tamaño
Este equipo	.sfdx	20/03/2021 14:59	Carpeta de archivos	
Red	build	20/03/2021 14:59	Carpeta de archivos	
	contracts	20/03/2021 14:59	Carpeta de archivos	
	migrations	20/03/2021 14:59	Carpeta de archivos	
	scripts	20/03/2021 14:59	Carpeta de archivos	
	src	20/03/2021 14:59	Carpeta de archivos	
	test	20/03/2021 14:59	Carpeta de archivos	
	.eslintignore	26/10/1985 10:15	Archivo ESLINTIG...	1 KB
	.eslintrc.json	26/10/1985 10:15	Archivo JSON	1 KB
	.prettierignore	26/10/1985 10:15	Archivo PRETTIERI...	1 KB
	.prettierrc	26/10/1985 10:15	Archivo PRETTIER...	1 KB
	jest.config.js	26/10/1985 10:15	Archivo JS	1 KB
	lwc.config.json	26/10/1985 10:15	Archivo JSON	1 KB
	lwc-services.config.js	26/10/1985 10:15	Archivo JS	1 KB
	package.json	26/10/1985 10:15	Archivo JSON	3 KB
	README.md	26/10/1985 10:15	Archivo MD	1 KB
	tfe-companies-0.0.1.tgz	26/10/1985 10:15	Archivo TGZ	806 KB
	truffle-config.js	26/10/1985 10:15	Archivo JS	5 KB

En este punto me descargo node v10, porque la aplicación web está realizada con LWC (lwc.dev) y necesita mínimo de esa versión

Index of /dist/latest-v10.x/

..		
docs/		23-Feb-2021 02:06
win-x64/		23-Feb-2021 01:54
win-x86/		23-Feb-2021 01:56
SHASUMS256.txt		23-Feb-2021 12:57 3347
SHASUMS256.txt.asc		23-Feb-2021 12:57 4220
SHASUMS256.txt.sig		23-Feb-2021 12:57 566
node-v10.24.0-linux-ppc64.tar.gz		23-Feb-2021 01:50 25495149
node-v10.24.0-darwin-x64.tar.gz		23-Feb-2021 02:44 18818432
node-v10.24.0-darwin-x64.tgz		23-Feb-2021 02:45 12338788
node-v10.24.0-headers.tar.gz		23-Feb-2021 02:31 546196
node-v10.24.0-headers.tar.xz		23-Feb-2021 02:31 358116
node-v10.24.0-linux-arm64.tar.gz		23-Feb-2021 01:17 20872817
node-v10.24.0-linux-arm64.tar.xz		23-Feb-2021 01:19 12600920
node-v10.24.0-linux-armv6l.tar.gz		23-Feb-2021 01:18 19809913
node-v10.24.0-linux-armv6l.tar.xz		23-Feb-2021 01:18 11552360
node-v10.24.0-linux-armv7l.tar.gz		23-Feb-2021 01:14 19668724
node-v10.24.0-linux-armv7l.tar.xz		23-Feb-2021 01:14 11510040
node-v10.24.0-linux-ppc64le.tar.gz		23-Feb-2021 01:14 20834751
node-v10.24.0-linux-ppc64le.tar.xz		23-Feb-2021 01:15 12380292
node-v10.24.0-linux-s390x.tar.gz		23-Feb-2021 01:15 21188617
node-v10.24.0-linux-s390x.tar.xz		23-Feb-2021 01:15 12508528
node-v10.24.0-linux-x64.tar.gz		23-Feb-2021 01:15 20933678
node-v10.24.0-linux-x64.tar.xz		23-Feb-2021 01:16 13139660
node-v10.24.0-linux-x64.tgz		23-Feb-2021 01:17 22142669
node-v10.24.0-linux-x64.tar.xz		23-Feb-2021 01:18 13702892
node-v10.24.0-win-x64/		23-Feb-2021 01:54 10302646
node-v10.24.0-win-x64.zip		23-Feb-2021 01:54 18308189
node-v10.24.0-win-x64.tgz		23-Feb-2021 01:56 9204737
node-v10.24.0-win-x64.zip		23-Feb-2021 01:58 16808326
node-v10.24.0-x64.msi		23-Feb-2021 01:54 19156992
node-v10.24.0-x86.msi		23-Feb-2021 01:58 17575936
node-v10.24.0.pkg		23-Feb-2021 01:47 19105820
node-v10.24.0.tar.gz		23-Feb-2021 02:26 46321470
node-v10.24.0.tar.xz		23-Feb-2021 02:30 21649616

Me descargo la versión, y cambio la variable PATH del sistema para que ejecute esa versión de node, en lugar de la v8 que es la que está en la VDI. Abro cmd, verifico la versión de node e instalo de forma global truffle en su versión 5.0.18:

```
C:\Users\pablo.cosio074>node --version  
v10.24.0  
C:\Users\pablo.cosio074>npm install -g truffle@5.0.18
```

Una vez acabada la install de truffle, verifico su versión:

```
C:\Users\pablo.cosio074>npm update  
C:\Users\pablo.cosio074>npm install -g truffle@5.0.18  
npm WARN deprecated mkdirp@0.5.1: Legacy versions of mkdirp are no longer supported. Please update to mkdirp 1.x. (Note that the API surface has changed to use Promises in 1.x.)  
C:\Users\pablo.cosio074\Downloads\node-v10.24.0-win-x64\truffle -> C:\Users\pablo.cosio074\Downloads\node-v10.24.0-win-x64\node_modules\truffle\build\cli.bundle.d.js  
+ truffle@5.0.18  
updated 1 package in 5.971s  
  
C:\Users\pablo.cosio074>truffle version  
Truffle v5.0.18 (core: 5.0.18)  
Solidity v0.5.0 (solc-jsc)  
Node v10.24.0  
Web3.js v1.0.0-beta.37
```

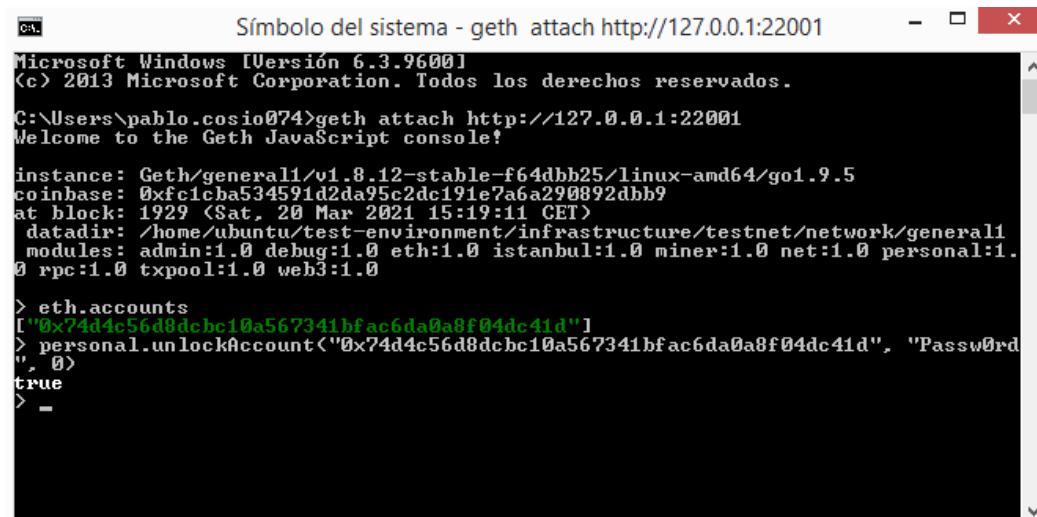
Ahora ya puedo ir a la carpeta del proyecto y ejecutar *npm install*

```
C:\Users\pablo.cosio074\webpack\package>npm install
```

Una vez la instalación finaliza, ejecuto los siguientes pasos:

- Setear variables de entorno:

Me conecto al nodo de general1 por geth

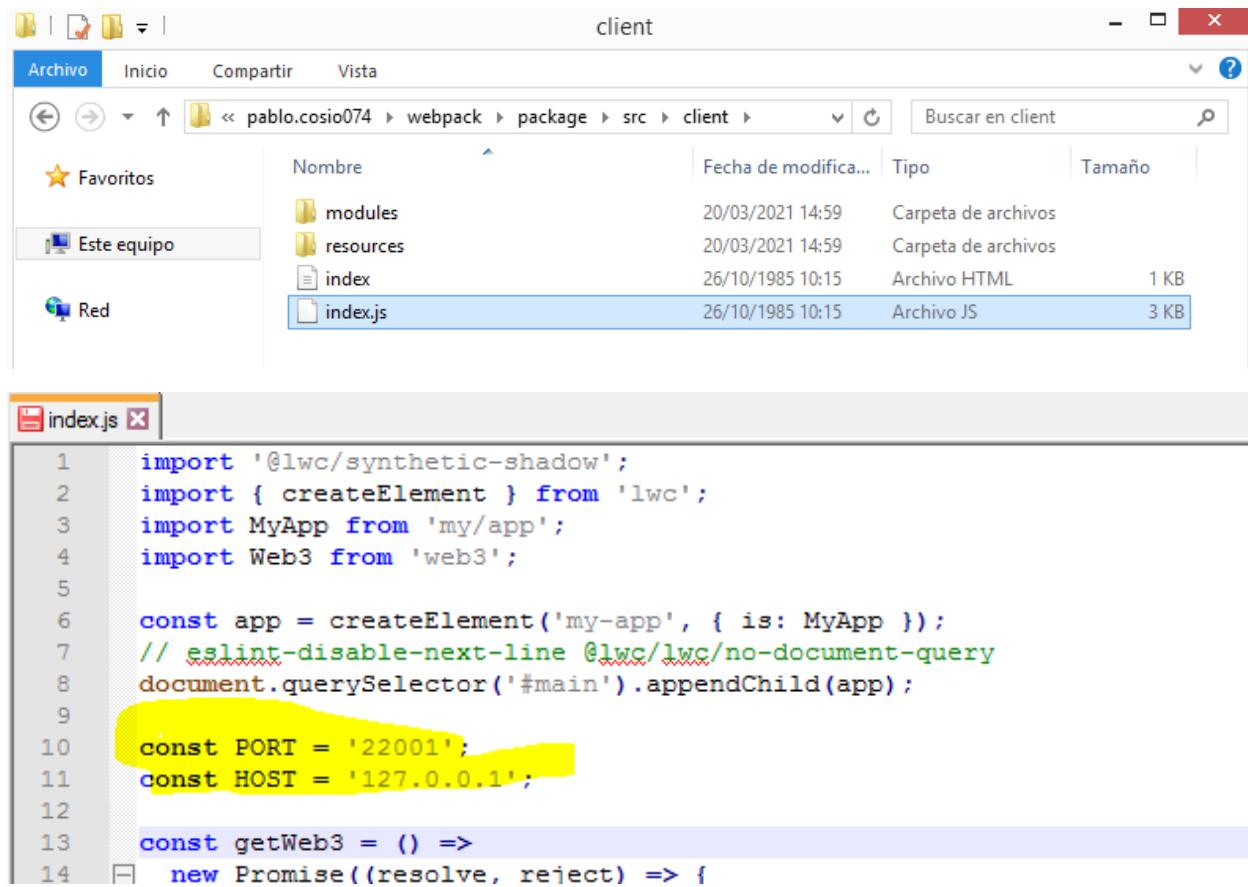


```
Símbolo del sistema - geth attach http://127.0.0.1:22001  
Microsoft Windows [Versión 6.3.9600]  
<c> 2013 Microsoft Corporation. Todos los derechos reservados.  
C:\Users\pablo.cosio074>geth attach http://127.0.0.1:22001  
Welcome to the Geth JavaScript console!  
instance: Geth/general1/v1.8.12-stable-f64dbb25/linux-amd64/go1.9.5  
coinbase: 0xfc1cba534591d2da95c2dc191e7a6a290892dbb9  
at block: 1929 (Sat, 20 Mar 2021 15:19:11 CET)  
datadir: /home/ubuntu/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  
> eth.accounts  
[ "0x74d4c56d8dbc10a567341bfac6da0a8f04dc41d" ]  
> personal.unlockAccount("0x74d4c56d8dbc10a567341bfac6da0a8f04dc41d", "Passw0rd", 0)  
true  
>
```

Obtengo el address del account y seteo las variables que necesito:

```
C:\Users\pablo.cosio074\webpack\package>set ACCOUNT_CREATION=0x74d4c56d8dcbe10a5  
67341bfac6da0a8f04dc41d  
C:\Users\pablo.cosio074\webpack\package>set AMOUNT_ADDRESS=100  
C:\Users\pablo.cosio074\webpack\package>set HOST_NETWORK=127.0.0.1  
C:\Users\pablo.cosio074\webpack\package>set PORT_NETWORK=22001  
C:\Users\pablo.cosio074\webpack\package>
```

En el fichero index.js modiflico también los valores para que el frontend pueda conectarse correctamente por web3.



The screenshot shows a Windows File Explorer window titled "client" with the following directory structure:

- Archivo
- Inicio
- Compartir
- Vista

Path: << pablo.cosio074 > webpack > package > src > client >

Nombre	Fecha de modifica...	Tipo	Tamaño
modules	20/03/2021 14:59	Carpetas de archivos	
resources	20/03/2021 14:59	Carpetas de archivos	
index	26/10/1985 10:15	Archivo HTML	1 KB
index.js	26/10/1985 10:15	Archivo JS	3 KB

Below the File Explorer is a code editor window showing the content of index.js:

```
1 import '@lwc/synthetic-shadow';
2 import { createElement } from 'lwc';
3 import MyApp from 'my/app';
4 import Web3 from 'web3';

5
6 const app = createElement('my-app', { is: MyApp });
// eslint-disable-next-line @lwc/lwc/no-document-query
7 document.querySelector('#main').appendChild(app);

8
9 const PORT = '22001';
10 const HOST = '127.0.0.1';

11
12 const getWeb3 = () =>
13   new Promise((resolve, reject) => {
```

- Ejecutar *truffle compile*. Al ejecutarlo sale un error:

```
C:\Users\pablo.cosio074\webpack\package>truffle compile

Compiling your contracts...
=====
> Compiling @openzeppelin/contracts/ownership/Ownable.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20Detailed.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721Full.sol
> Compiling @openzeppelin/contracts/GSN/Context.sol
> Compiling @openzeppelin/contracts/drafts/Counters.sol
> Compiling @openzeppelin/contracts/introspection/IERC165.sol
> Compiling @openzeppelin/contracts/introspection/IERC165.sol
> Compiling @openzeppelin/contracts/math/SafeMath.sol
> Compiling @openzeppelin/contracts/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\TFEDron.sol
> Compiling .\contracts\TFEPparcela.sol
> Compiling .\contracts\TFEPpayments.sol
> Compiling .\contracts\TFEPparcela.sol
> Compiling .\contracts\TFEPpayments.sol

> Compilation warnings encountered:

  @openzeppelin/contracts/utils/Address.sol:31:32: Warning: The "extcodehash"
instruction is not supported by the VM version "byzantium" you are currently com-
piling for. It will be interpreted as an invalid instruction on this VM.
    assembly { codehash := extcodehash(account) }
                           ^____^____^

@openzeppelin/contracts/utils/Address.sol:1:1: SyntaxError: Source file requires
different compiler version (current compiler is 0.5.0+commit.1d4f565a.Emscripten
clang - note that nightly builds are considered to be strictly less than the r
eleased version
pragma solidity ^0.5.5;
^____^

Error: Truffle is currently using solc 0.5.0, but one or more of your contracts
specify "pragma solidity ^0.5.5".
Please update your truffle config or pragma statement(s).
<See https://trufflesuite.com/docs/truffle/reference/configuration#compiler-conf
figuration for information on
configuring Truffle to use a specific solc compiler version.>

Compilation failed. See above.
Truffle v5.2.5 (core: 5.2.5)
Node v10.24.0
```

Este error parece que se debe a que no se instala bien la dependencia de openzeppelin (en mi pc no me ha pasado, sólo en la VDI), ya que todas las versiones de solidity de los contratos son 0.5.0 Ejecutando de nuevo:

```
npm i @openzeppelin/contracts@2.3.0
```

y volviendo a ejecutar *truffle compile* la compilación ya es correcta:

```
C:\Users\pablo.cosio074\webpack\package>truffle compile
Compiling your contracts...
=====
> Compiling @openzeppelin/contracts/ownership/Ownable.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20.sol
> Compiling @openzeppelin/contracts/token/ERC20/ERC20Detailed.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721Full.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/token/ERC20/IERC20.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/ERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Enumerable.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Metadata.sol
> Compiling @openzeppelin/contracts/token/ERC721/IERC721Receiver.sol
> Compiling @openzeppelin/contracts/utils/Address.sol
> Compiling .\contracts\Migrations.sol
> Compiling .\contracts\TFEDron.sol
> Compiling .\contracts\TFEP parcela.sol
> Compiling .\contracts\TFEP payments.sol
> Compiling .\contracts\TFEP parcela.sol
> Compiling .\contracts\TFEP payments.sol
> Artifacts written to C:\Users\pablo.cosio074\webpack\package\build\contracts
> Compiled successfully using:
  - solc: 0.5.0+commit.1d4f565a.Emscripten clang

C:\Users\pablo.cosio074\webpack\package>
```

- Ejecutar *truffle migrate –network alastriadev*. Con esto se realiza el despliegue de los contratos.

```
C:\ Símbolo del sistema
=====

Compiling your contracts...
=====
> Compiling .\contracts\TFEPparcela.sol
> Compiling .\contracts\TFEPayments.sol
> Artifacts written to C:\Users\pablo.cosio074\webpack\package\build\contracts
> Compiled successfully using:
  - solc: 0.5.0+commit.id4f565a.Emscripten clang

Starting migrations...
=====
> Network name:      'alastriadev'
> Network id:        9535753591
> Block gas limit: 1713687130 <0x6624ca5a>

1_initial_migration.js
=====

  Deploying 'Migrations'
  -----
  > transaction hash: 0xc4e898cba365e3d9fb4d2bed30ca3ad2e3e8b3039576140efab1
14ed4da1bc8e
  > Blocks: 0          Seconds: 0
  > contract address: 0x35840d91b29a79F6A7B20d7b27208C9503a58a3A
  > block number:     3101
  > block timestamp: 1616251123
  > account:          0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
  > balance:          0
  > gas used:         237773 <0x3a0cd>
  > gas price:        0 gwei
  > value sent:       0 ETH
  > total cost:       0 ETH

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

2_deploy_contracts.js
=====

  Deploying 'TFEPparcela'
  -----
  > transaction hash: 0x8ad87395954e61c2495ab06fb20a9375258cd3ac8406cfb8428f
cf735ca7811
  > Blocks: 0          Seconds: 0
  > contract address: 0x23Ed61cFd8dA3b84ff5d0D31F285B0C07051d87b
  > block number:     3105
  > block timestamp: 1616251127
  > account:          0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
  > balance:          0
  > gas used:         3277765 <0x3203c5>
  > gas price:        0 gwei
  > value sent:       0 ETH
  > total cost:       0 ETH

PARCELA Contract deployed -> 0x23Ed61cFd8dA3b84ff5d0D31F285B0C07051d87b

  Deploying 'TFEPayments'
  -----
  > transaction hash: 0xb24c87115ba85bde8db8d60272440123cf85090c0965437f44c
45e9bf1bed3b
  > Blocks: 0          Seconds: 0
  > contract address: 0x7684fB450dE19C57e164d588Fe8FF399ba82E16D
  > block number:     3108
  > block timestamp: 1616251130
  > account:          0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
  > balance:          0
  > gas used:         1761533 <0x1ae0fd>
  > gas price:        0 gwei
  > value sent:       0 ETH
  > total cost:       0 ETH

PAYMENTS Contract deployed -> 0x7684fB450dE19C57e164d588Fe8FF399ba82E16D

  Deploying 'TFEDron'
  -----
  > transaction hash: 0x56fb9645f8d5900d12a5a738cc1863a434f91cde33c25b3a803
b447493ad537
  > Blocks: 0          Seconds: 0
  > contract address: 0x48C8EEc537a88B56492F015B2F4F054bE133D112
  > block number:     3110
  > block timestamp: 1616251132
  > account:          0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
  > balance:          0
  > gas used:         3902212 <0x3b8b04>
  > gas price:        0 gwei
  > value sent:       0 ETH
  > total cost:       0 ETH

DRON Contract deployed -> 0x48C8EEc537a88B56492F015B2F4F054bE133D112

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

Summary
=====
> Total deployments: 4
> Final cost:        0 ETH
```

- Ejecutar `npm run watch`. Este comando inicia el backend (nodejs express) y el frontend (lwc). El backend lo levanta en el puerto 3002 y el fronted en el 3001

```
C:\Users\pablo.cosio074\webpack\package>npm run watch
> tfe-companies@0.0.1 watch C:\Users\pablo.cosio074\webpack\package
> run-p watch:client watch:server

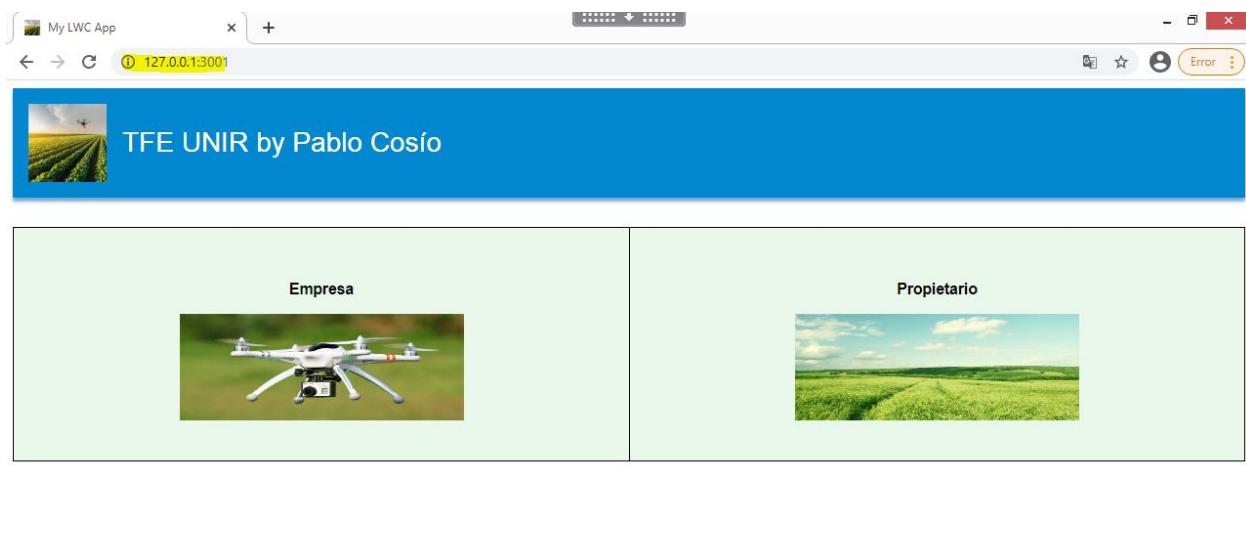
> tfe-companies@0.0.1 watch:client C:\Users\pablo.cosio074\webpack\package
> lwc-services watch

> tfe-companies@0.0.1 watch:server C:\Users\pablo.cosio074\webpack\package
> nodemon

????? Lightning Web Components ?????

? Starting build process.
? Local server listening: http://localhost:3001
```

Verificamos que accedemos a la webapp correctamente



#### 4.3. Despliegue en red T de Alastria (PROD)

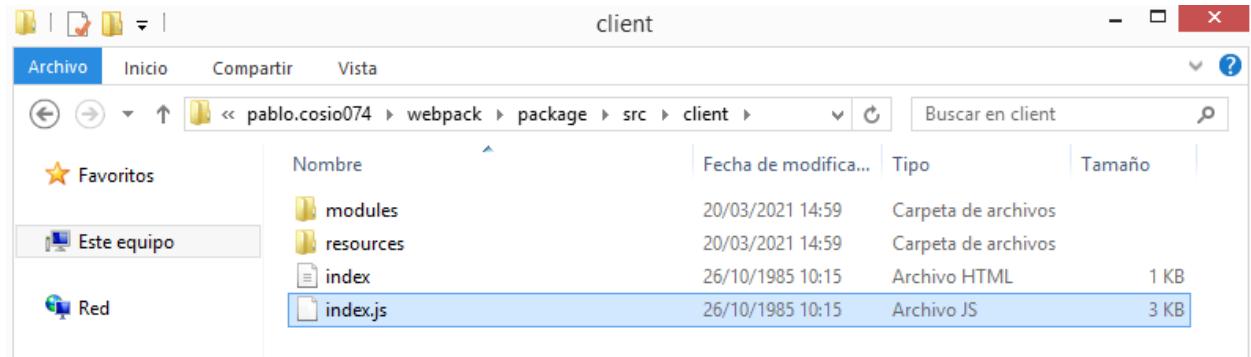
Para desplegar en la mainnet de Alastria Telsius, se han seguido los pasos descritos en el punto 4. Entre ellos los de instalar truffle, y hacer el npm install sobre el proyecto ya descomprimido. Una vez en este punto para poder hacer el despliegue es necesario realizar alguna cosa diferente:

Me conecto al nodo de Unir mediante geth y desbloqueo 3 de las cuentas que los alumnos podemos utilizar:

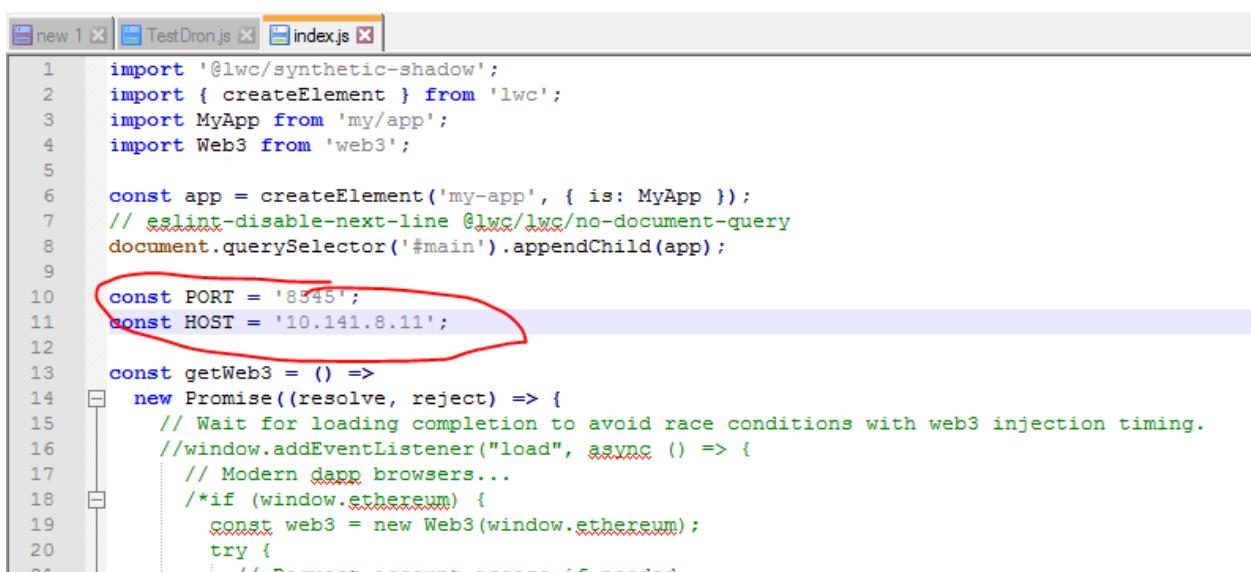
Obtengo el address del account con el que quiero desplegar y seteo las variables que necesito:

```
C:\Users\pablo.cosio074\webapp\package>set ACCOUNT_CREATION=0xbc869c21d631e122d35789942a573241ec04d2e4
C:\Users\pablo.cosio074\webapp\package>set AMOUNT_ADDRESS=100
C:\Users\pablo.cosio074\webapp\package>set HOST_NETWORK=10.141.8.11
C:\Users\pablo.cosio074\webapp\package>set PORT_NETWORK=8545
C:\Users\pablo.cosio074\webapp\package>
```

En el fichero index.js modifíco también los valores para que el frontend pueda conectarse correctamente por web3.



Nombre	Fecha de modificación	Tipo	Tamaño
modules	20/03/2021 14:59	Carpeta de archivos	
resources	20/03/2021 14:59	Carpeta de archivos	
index	26/10/1985 10:15	Archivo HTML	1 KB
<b>index.js</b>	26/10/1985 10:15	Archivo JS	3 KB

```

1 import '@lwc/synthetic-shadow';
2 import { createElement } from 'lwc';
3 import MyApp from 'my/app';
4 import Web3 from 'web3';

5 const app = createElement('my-app', { is: MyApp });
// eslint-disable-next-line @lwc/lwc/no-document-query
document.querySelector('#main').appendChild(app);

10 const PORT = '8545';
11 const HOST = '10.141.8.11';

12 const getWeb3 = () =>
13   new Promise((resolve, reject) => {
14     // Wait for loading completion to avoid race conditions with web3 injection timing.
15     //window.addEventListener("load", async () => {
16       // Modern dapp browsers...
17       /*if (window.ethereum) {
18         const web3 = new Web3(window.ethereum);
19         try {
20           // Request account access if needed
21         }
22       }
23     })
24   })
25 
```

Modifico también el fichero de test para decirle a truffle que utilice las cuentas que yo le diga para hacer el test:

Screenshot of a file explorer window showing a directory structure and a code editor.

The file explorer window shows a directory path: `<< Usuarios > pablo.cosio074 > webapp > package > test`. The current folder is `test`. Inside, there are two files: `.gitkeep` (modified 26/10/1985 10:15, 0 KB) and `TestDron.js` (modified 21/03/2021 16:45, 4 KB). A red circle highlights `TestDron.js`.

The code editor window shows the contents of `TestDron.js`:

```

1 const DronContract = artifacts.require("TFEDron");
2 const ParcelaContract = artifacts.require("TFEParcela");
3 const PaymentsContract = artifacts.require("TFEPayments");
4
5 const assert = require("chai").assert;
6 const TruffleAssert = require('truffle-assertions');
7
8 contract("DronContract", accounts => {
9     accounts[0] = '0xbc8e9c21d631e122d35789942a573211ec04d2e4';
10    accounts[1] = '0xad11f232919a54696791387e3a74a63399c2dafb';
11    accounts[2] = '0x35ad5e72cb2ec714b80154b796c7835f97053d3e';
12    let drones;
13    let parcelas;
14    let paymentContract;
15    let dronUri = 'https://googles.es/1';
16    let parcelaUri = 'https://googles.es/2';
17    let tokenIdParcela;
18    let tokenIdDron;
19    let jobId;
20
21    beforeEach("Desplegar nuevo contrato Drones", async function() {
22        if(!drones)
23    })

```

A red circle highlights the line `accounts[2] = '0x35ad5e72cb2ec714b80154b796c7835f97053d3e';`

Compilar los contratos:

```
C:\Users\pablo.cosio074\webapp\package>truffle compile  
Compiling your contracts...  
=====  
> Compiling Copenzeppelin\contracts\ownership\Ownable.sol  
> Compiling Copenzeppelin\contracts\token\ERC20\ERC20.sol  
> Compiling Copenzeppelin\contracts\token\ERC20\ERC20Detailed.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\ERC721Full.sol  
> Compiling Copenzeppelin\contracts\drafts\Counters.sol  
> Compiling Copenzeppelin\contracts\introspection\ERC165.sol  
> Compiling Copenzeppelin\contracts\introspection\IERC165.sol  
> Compiling Copenzeppelin\contracts\math\SafeMath.sol  
> Compiling Copenzeppelin\contracts\token\ERC20\IERC20.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\ERC721.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\ERC721Enumerable.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\ERC721Metadata.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\IERC721.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\IERC721Enumerable.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\IERC721Metadata.sol  
> Compiling Copenzeppelin\contracts\token\ERC721\IERC721Receiver.sol  
> Compiling Copenzeppelin\contracts\utils\Address.sol  
> Compiling .\contracts\Migrations.sol  
> Compiling .\contracts\TFEDron.sol  
> Compiling .\contracts\TFEParcela.sol  
> Compiling .\contracts\TFEPayments.sol  
> Compiling .\contracts\TFEParcela.sol  
> Compiling .\contracts\TFEPayments.sol  
> Artifacts written to C:\Users\pablo.cosio074\webapp\package\build\contracts  
> Compiled successfully using:  
  - solc: 0.5.0+commit.1d4f565a.Emscripten clang
```

Desplegar los contratos: *truffle migrate --network alastriadev*

```

Compiling your contracts...
=====
> Compiling .\contracts\TFEParela.sol
> Compiling .\contracts\TFEPayments.sol
> Artifacts written to C:\Users\pablo.cosio074\webapp\package\build\contracts
> Compiled successfully using:
  - solc: 0.5.0+commit.1d4f565a.Emscripten clang

Starting migrations...
=====
> Network name: 'alastriadev'
> Network id: 83584648538
> Block gas limit: 0x29b68b6f

1_initial_migration.js
=====

  Deploying 'Migrations'
  -----
  > transaction hash: 0xebada23781999cfafa2ba62564c8ea03137a949fd4c3745910e4be
0cd0ec2e0304
  > Blocks: 0 Seconds: 0
  > contract address: 0xA7D47416B939d81522cc83e3cEB86d94d3DAEef6
  > block number: 65191036
  > block timestamp: 161634178
  > account: 0xcb869c21d631E122d35789942A573241ec04D2E4
  > balance: 0
  > gas used: 237723
  > gas price: 0 gwei
  > value sent: 0 ETH
  > total cost: 0 ETH

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

2_deploy_contracts.js
=====

  Deploying 'TFEParela'
  -----
  > transaction hash: 0x20c974d26b24ec0d91401c90f4478e86bf771635a1e039b64fc9
7c11b9ea7c6b
  > Blocks: 0 Seconds: 0
  > contract address: 0xA342097441f3df1331c687a9b33A72BC1008d2c7
  > block number: 65191041
  > block timestamp: 1616341783
  > account: 0xcb869c21d631E122d35789942A573241ec04D2E4
  > balance: 0
  > gas used: 3277765
  > gas price: 0 gwei
  > value sent: 0 ETH
  > total cost: 0 ETH

PARCELA Contract deployed -> 0xA342097441f3df1331c687a9b33A72BC1008d2c7

  Deploying 'TFEPayments'
  -----
  > transaction hash: 0xf632a9eb9fac45614f525b3c576d1959704f2ef1f3f3741dc39d
45924426c7a3
  > Blocks: 0 Seconds: 0
  > contract address: 0xB1dB7B51339bDdF5013318537fd74059a8C55c8
  > block number: 65191043
  > block timestamp: 1616341785
  > account: 0xcb869c21d631E122d35789942A573241ec04D2E4
  > balance: 0
  > gas used: 1761533
  > gas price: 0 gwei
  > value sent: 0 ETH
  > total cost: 0 ETH

PAYMENTS Contract deployed -> 0xB1dB7B51339bDdF5013318537fd74059a8C55c8

  Deploying 'TFEDron'
  -----
  > transaction hash: 0xe9494a9d1e09049cf613acf7c335e32b314b6acf8aee8c98145
31d04d01c06f
  > Blocks: 0 Seconds: 0
  > contract address: 0x73E2A85F598c2c59440CF37F9F37bC34d4510C84
  > block number: 65191045
  > block timestamp: 1616341787
  > account: 0xcb869c21d631E122d35789942A573241ec04D2E4
  > balance: 0
  > gas used: 3902212
  > gas price: 0 gwei
  > value sent: 0 ETH
  > total cost: 0 ETH

DRON Contract deployed -> 0x73E2A85F598c2c59440CF37F9F37bC34d4510C84

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

Summary
=====
> Total deployments: 4
> Final cost: 0 ETH

```

Como se puede observar los contratos están desplegados en la red de Alastria:

*PARCELA Contract deployed -> **0xA342097441f3df1331c687a9b33A72BC1008d2c7***

*PAYMENTS Contract deployed -> **0xB1dB7B51339bDdF5013318537fdD74059a8C55c8***

*DRON Contract deployed -> **0x73E2A85F598c2c59440CF37F9F37bC34d4510C84***

Nos podemos ir al explorar de bloques de la red T de Alastria, y ver los datos de cada transacción asociada a cada despliegue

<https://blkexplorer1.telsius.alastria.io/transaction/0xe9494a9d1e09049fcfc613acf7c335e32b314b6acf8aee8c9814531d04d01c06f>

<https://blkexplorer1.telsius.alastria.io/transaction/0xf632a9eb9fac45614f525b3c576d1959704f2ef1f3f3741dc39d45924426c7a3>

<https://blkexplorer1.telsius.alastria.io/transaction/0x20c974d26b24ec0d91401c90f4478e86bf771635a1e039b64fc97c11b9ea7c6b>

Vemos que están en bloques consecutivos por lo que cada transacción ha sido minada o añadida a la red en bloques diferentes.

En Remix también podemos ver que el contrato del Dron se ha desplegado correctamente:

The screenshot shows the Truffle UI interface for deploying and running transactions. The environment is set to "Web3 Provider" (localhost:8546) and the account is "0xcb8...4D2E4 (0 ether)". The contract being deployed is "TFEDron - browser/TFEDron.sol". The code editor displays the Solidity source code for the TFE Dron contract, which interacts with ERC721Full and Ownership contracts. The Deploy button is highlighted. The Deployed Contracts section lists various functions such as addJob, approve, and isOwner. The isOwner function is circled in red, and its implementation is shown in the code editor:

```

    ...
    function isOwner() public view returns (bool) {
        return msg.sender == owner;
    }
    ...

```

Se prueba también a ejecutar los tests. Para ello tengo que hacer un cambio en las siguientes líneas del test:

```
assert.equal(ownerDron.toLowerCase(), accounts[1], "La parcela no se ha creado o no tiene el owner correcto");
```

```
assert.equal(ownerParcela.toLowerCase(), accounts[2], "La parcela no se ha creado o no tiene el owner correcto");
```

Es decir, pasar a minisculas las account address ya que sino el assert equal lo detecta como diferente, puesto que las direcciones de entrada las pusimos en lower case:

```
accounts[0] = '0xbc869c21d631e122d35789942a573241ec04d2e4';
accounts[1] = '0xad11f232919a54696791387e3a74a63394c2dafb';
accounts[2] = '0x35ad6e72cb2ec714b80154b796c7835f97053d3e';
```

Con ese pequeño cambio los tests ya se ejecutan correctamente:

```
C:\Users\pablo.cosio074\webapp\package>truffle test --network alastriadev
Using network 'alastriadev'.

Compiling your contracts...
=====
> Compiling ./contracts\TFEParela.sol
> Compiling ./contracts\TFEPayments.sol
> Artifacts written to C:\Users\PABLO~1.COS\AppData\Local\Temp\test-121221-5036-005dff.slnw
> Compiled successfully using:
  - solc: 0.5.0+commit.1d4f565a.Emscripten clang

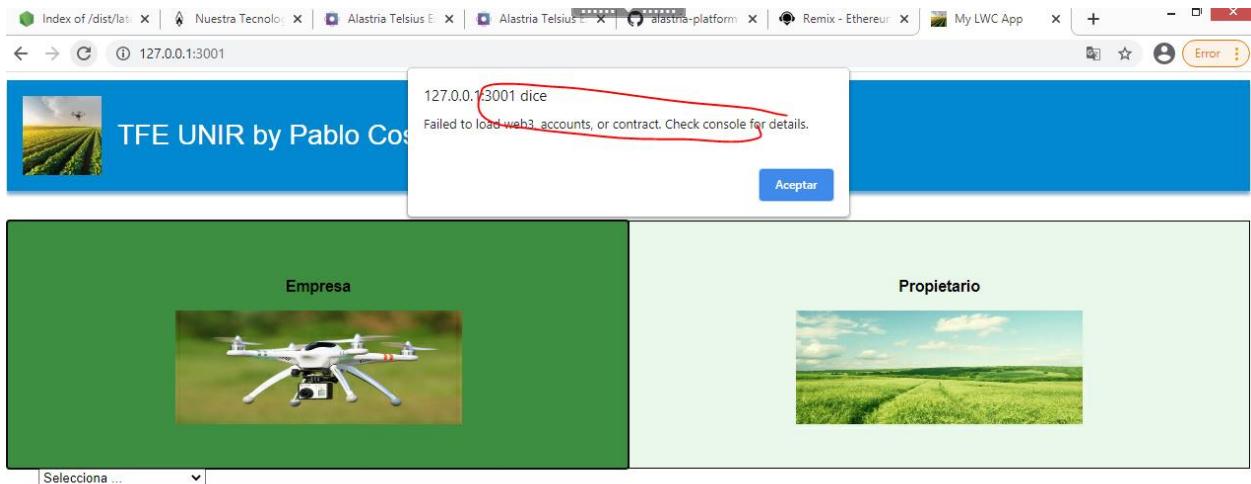
PARCELA Contract deployed -> 0x85b302d2aEeE680726701882f89f6ff1E7cA1bC1
PAYMENTS Contract deployed -> 0x1FAEC1af49D261a76e71a84e03395D0C45e5ff56
DRON Contract deployed -> 0x444D9Faa50C3819F6Ae5e687e36F0Aa317414A6B

  Contract: DronContract
    Prueba flujo completo
  Contrato Dron desplegado
  Contrato Dron Parela
  Contrato Payments desplegado
    ✓ Crear nuevo dron <3047ms>
    ✓ Crear nueva parcela <60ffms>
    ✓ Propietario parcela contratar dron <3032ms>
    ✓ Propietario parcela realiza el pago <3019ms>
    ✓ Empresa recibe el pago
    ✓ Empresa del Dron asigna el trabajo pendiente al dron y verifica que la parcela está fumigada <3025ms>

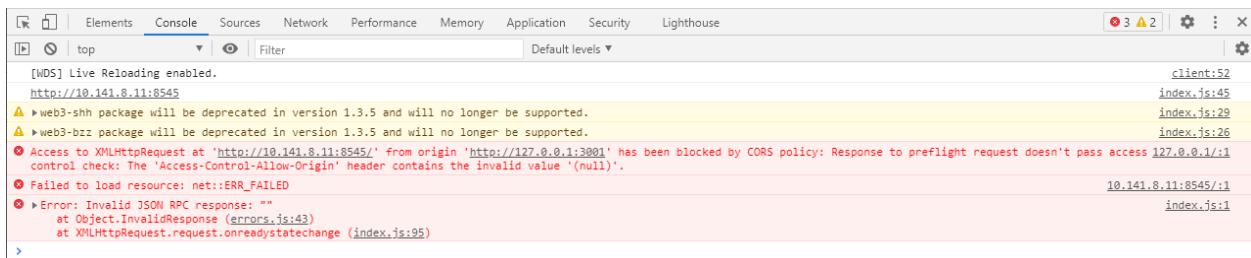
  6 passing (18s)
```

Ahora verificamos que la aplicación arranca correctamente, ejecutando: `npm run watch`

Vemos que la aplicación al cargar no consigue conectar:



Si abrimos la consola del Chrome, vemos:



El node de Alastria debe tener configurado algún tipo de regla en el Firewall o WAF para no permitir conexiones desde web3 desde aplicaciones webs no autorizadas.

## 5. Testing de la solución

Teniendo en cuenta que las dos lógicas principales, que son:

- la de comprobar que el owner del tokenID de la parcela corresponde con la dirección que está enviando la transacción de contratar dron
- la del pago desde el propietario de la parcela a la empresa del dron

Estas como se ha comentado en el punto 2, no se ha sido capaz de realizar onchain por lo que dicha lógica se ha hecho desde el Javascript. Por falta de tiempo, debería por tanto haber ejecutado el test de toda esta capa frontend donde reside esta lógica, con herramientas por ejemplo como **Selenium**. Por falta de tiempo y falta de conocimiento en esta área, no he podido llevarlo a cabo, simplemente quería dejarlo reflejado en la memoria.

Teniendo en cuenta esto he dividido los tests en los siguientes apartados:

- Test automático con truffle. En este caso he realizado solamente los tests unitarios con javascript, basados en Mocha y chai (aserciones). Debería haber realizado también todas las pruebas también con Solidity, pero por falta de tiempo no he podido implementarlo.
- Test funcionales de los casos de uso de la aplicación

### 5.1. Test unitarios con truffle

Para las pruebas con truffle, pasos que he seguido:

- Setear variables de entorno, de la misma forma que en el punto 4.

```
C:\Users\pablo.cosio074\webpack\package>set ACCOUNT_CREATION=0x74d4c56d8dcbe10a5  
67341bfac6da0a8f04dc41d  
C:\Users\pablo.cosio074\webpack\package>set AMOUNT_ADDRESS=100  
C:\Users\pablo.cosio074\webpack\package>set HOST_NETWORK=127.0.0.1  
C:\Users\pablo.cosio074\webpack\package>set PORT_NETWORK=22001
```

crear dos nuevas cuentas y desbloquearlas

Símbolo del sistema - geth attach http://127.0.0.1:22001

```
C:\Users\pablo.cosio074>geth attach http://127.0.0.1:22001
Welcome to the Geth JavaScript console!

instance: Geth/general1/v1.8.12-stable-f64dbb25/linux-amd64/go1.9.5
coinbase: 0xfc1cba534591d2da95c2dc191e7a6a290892dbb9
at block: 1929 <Sat, 20 Mar 2021 15:19:11 CET>
  datadir: /home/ubuntu/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

> eth.accounts
["0x74d4c56d8dcfc10a567341bfac6da0a8f04dc41d"]
> personal.unlockAccount("0x74d4c56d8dcfc10a567341bfac6da0a8f04dc41d", "Passw0rd", 0)
true
> personal.newAccount("Passw0rd")
"0x2822996190b05c548d16564c404475f8161a081a"
> personal.unlockAccount("0x2822996190b05c548d16564c404475f8161a081a", "Passw0rd", 0)
true
> personal.newAccount("Passw0rd")
"0xcf96d2264e19717d20fc22a9dcaee3924b0e0d46"
> personal.unlockAccount("0xcf96d2264e19717d20fc22a9dcaee3924b0e0d46", "Passw0rd", 0)
true
> -
```

ejecutar test: truffle test

```
C:\Users\pablo.cosio074\webpack\package>truffle test --network alastriadev
Using network 'alastriadev'.

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

PARCELA Contract deployed -> 0xABBF8190b14EFbA89C1A3788491Bbdc4f4c79c4B
PAYMENTS Contract deployed -> 0x04ba82ee89A51e45955985AA0047B503A1b3b855
DRON Contract deployed -> 0xc5B7101F0331D880726d339987A32C15Bab0edFE

  Contract: DronContract
    Prueba flujo completo
  Contrato Dron desplegado
  Contrato Dron Parcels
  Contrato Payments desplegado
    ✓ Crear nuevo dron <2068ms>
    ✓ Crear nueva parcela <4068ms>
    ✓ Propietario parcela contratar dron <2039ms>
    ✓ Propietario parcela realiza el pago <2031ms>
    ✓ Empresa recibe el pago
    ✓ Empresa del Dron asigna el trabajo pendiente al dron y verifica que la parcela está fumigada <2030ms>

  6 passing <13s>
```

## 5.2. Casos de uso

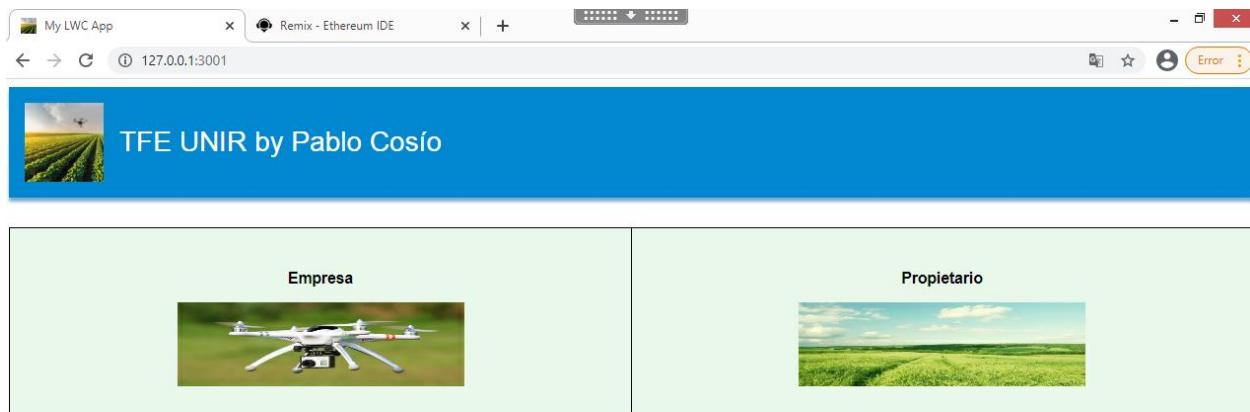
Se han realizado las pruebas resumidas en esta tabla:

0	<b>Creación de empresa</b> -> se crea una nueva Account Addres protegida con la password suministrada <b>Registro de primer dron</b> -> se crea un nuevo item en el contrato del dron (asignándole un tokenId y un owner) además de almacenarse la metadata en formato JSON en IPFS
1	<b>Empresa listar drones</b> -> listar drones de los cuales es propietario. Verificar que este filtro es correcto
2	<b>Creación de propietario</b> -> se crea una nueva Account Addres protegida con la password suministrada y además se le asignan 100 (variable de entorno AMOUNT_ADDRESS) tokens (desplegado en contrato TFEPayments) <b>Registro primera parcela</b> -> se crea un nuevo item en el contrato de parcela (asignándole un tokenId y un owner) además de almacenarse la metadata en formato JSON en IPFS
3	<b>Propietario listar parcelas</b> -> listar parcelas de los cuales es propietaria. Verificar que este filtro es correcto.
4	<b>Propietario ver drones disponibles y compatibles</b> -> en el listado de parcelas de la empresa, asociar un botón a cada parcela para poder ver todos los drones disponibles y compatibles con la parcela que se quiere fumigar
5	<b>Propietario parcela contratar dron</b> -> verificar que se añade el trabajo pendiente de asignar a la empresa, verificar que se realiza la transferencia de los tokens correspondientes al coste del dron desde la cuenta del propietario de la parcela a la empresa del dron. Verificar que se captura el evento de trabajo pendiente de asignación con el jobId generado.
6	<b>Empresa listar trabajos pendientes de asignar</b> -> la empresa visualiza todos los Jobs pendientes de asignar a sus drones Verificar que solo se listan los trabajos pendientes para drones de las cuales es owner. Verificar que sólo salen los que nos están aprobados/asignados
7	<b>Empresa asignar un trabajo a dron</b> -> verificar que el jobId creado en el punto 5, cambia de estado a aprobado, y que se emite el evento de parcela fumigada.

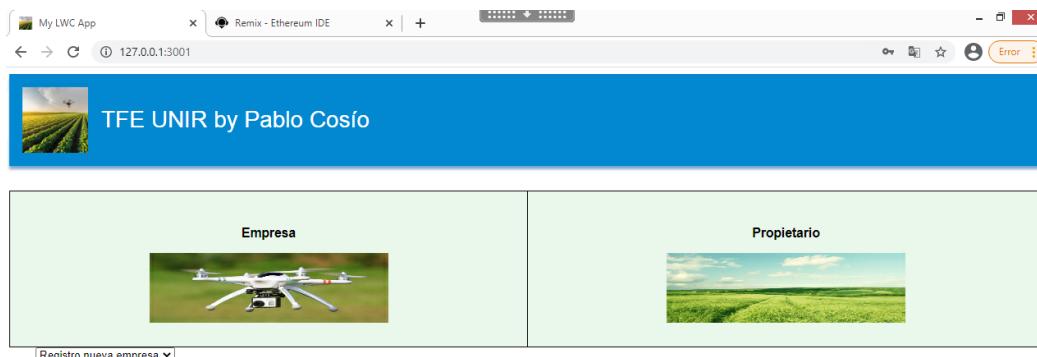
## 6. Manual de usuario

URL de entrada: <http://127.0.0.1:3001/>

### 6.1. Registro de Empresa y de drones



Empresa registrar dron. Introducir los valores



Introduzca los datos para poder registrar la empresa

Nombre Empresa:

Password:

Introduzca los datos para poder registrar el Dron

Dron -> Altura vuelo máxima:

Dron -> Altura vuelo mínima:

Dron -> coste:

Dron -> Pesticidas que puede suministrar:  
Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E

Se crea una nueva cuenta para la empresa

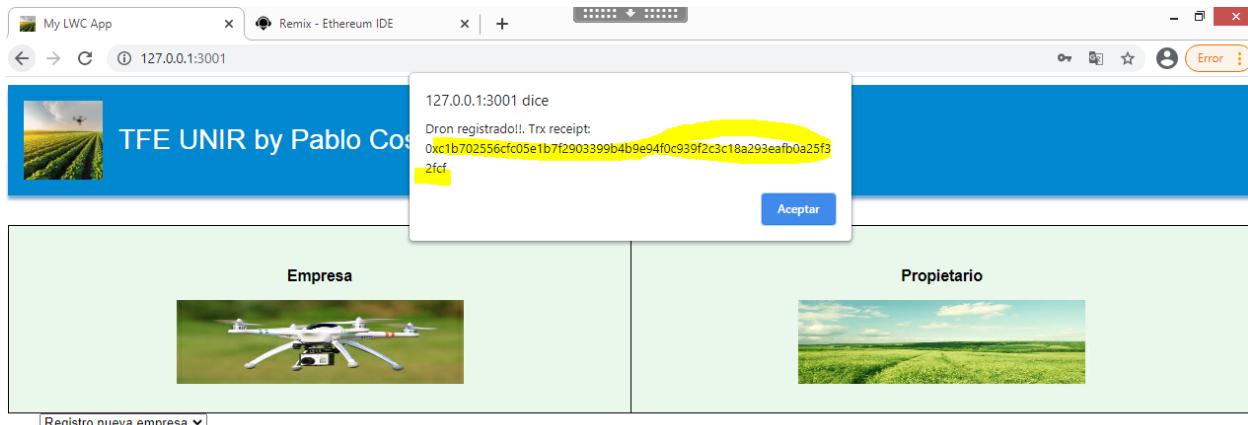
## Introduzca los datos para poder registrar la empresa

Nombre Empresa:  
empresa 5.07  
Password:  
....

## Introduzca los datos para poder registrar el Dron

Dron -> Altura vuelo máxima:  
1500  
Dron -> Altura vuelo mínima:  
200  
Dron -> coste:  
15  
Dron -> Pesticidas que puede suministrar:  
Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E

Se realiza la llamada al contrato del Dron y se añade un nuevo item/dron. Se devuelve la transacción y además en la consola se podría ver la url de ipfs generada al almacenar allí el JSON con la metadata del dron

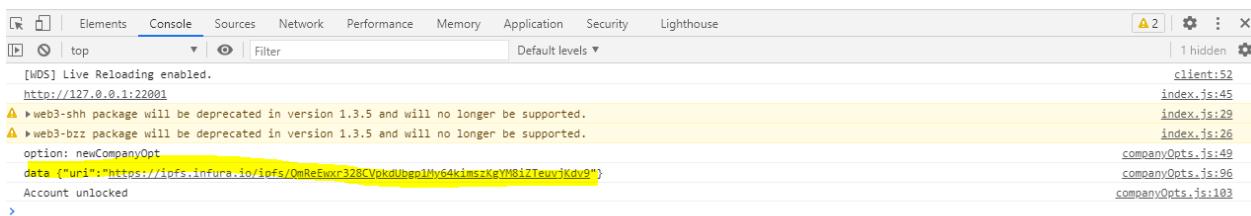


## Introduzca los datos para poder registrar la empresa

Nombre Empresa:  
empresa 5.07  
Password:  
....

## Introduzca los datos para poder registrar el Dron

Dron -> Altura vuelo máxima:  
1500  
Dron -> Altura vuelo mínima:  
200  
Dron -> coste:  
15  
Dron -> Pesticidas que puede suministrar:  
Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E



Ahora vuelvo a entra a la web pero esta vez en lugar de clickar en registro de nueva empresa, selecciono, Empresa existente y click en listar drones de la empresa.

Screenshot of a web browser showing a LWC application titled "TFE UNIR by Pablo Cosío". The page displays two sections: "Empresa" (with an image of a drone) and "Propietario" (with an image of a green field). A dropdown menu at the bottom left shows "Empresa existente".

## Introduzca la public address de la empresa

Public Address:  
`f5837b2869b7fC5112a9EE`

[Listar Drones de la empresa](#)

Screenshot of a web browser showing a LWC application titled "TFE UNIR by Pablo Cosío". The page displays two sections: "Empresa" (with an image of a drone) and "Propietario" (with an image of a green field). A dropdown menu at the bottom left shows "Empresa existente". Below the sections are buttons for "Registrar nuevo dron" and "Listar todos los trabajos pendientes por asignar". A password input field is present, and a link to an IPFS URL is shown.

Al salir el listado, hay 3 opciones:

Registrar nuevo dron

Listar todos los trabajos de los drones pendientes de asignar

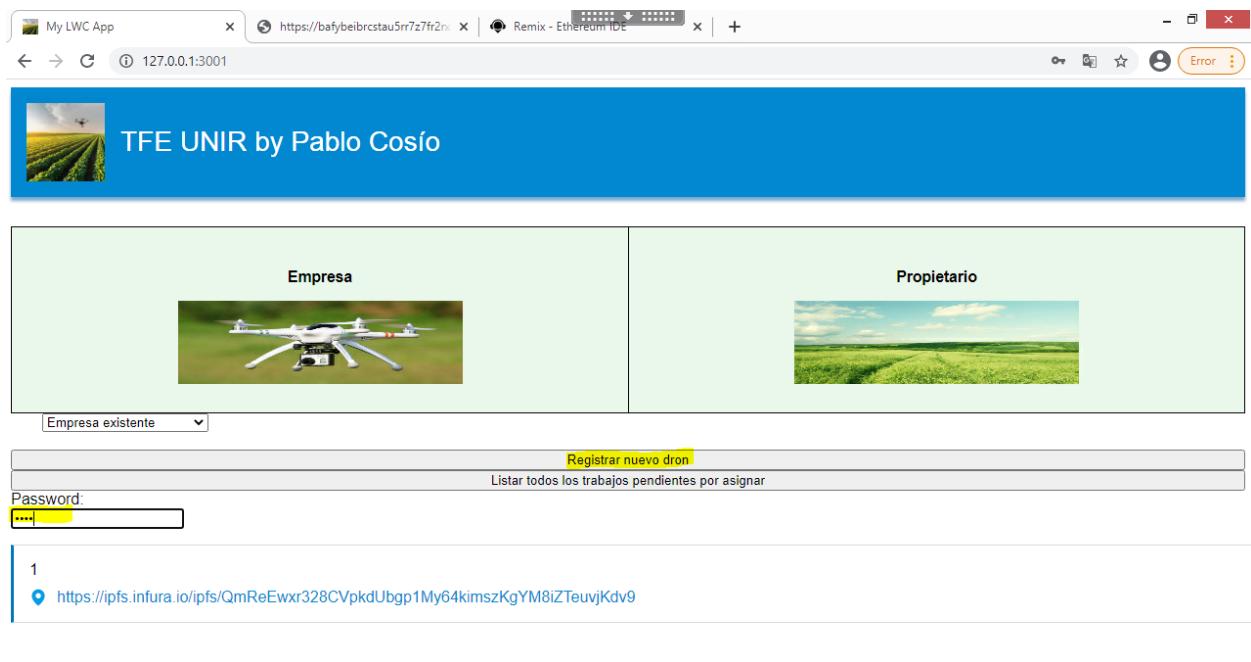
Haciendo click en un elemento del listado, nos podemos descargar el json con la metadata que se encuentra almacenado en IPFS



The screenshot shows a browser window with three tabs: "My LWC App", "https://bafybeibrcstau5rr7z7fr2xong5gsvectfhgyjldnq6vdfcy6mkv2fi.ipfs.infura-ipfs.io", and "Remix - Ethereum IDE". The middle tab contains the following JSON code:

```
{"companyName": "empresa 5:07", "dronHmax": 1500, "dronHmin": 200, "coste": 15, "pesticidas": {"pesticidaA": true, "pesticidaB": false, "pesticidaC": false, "pesticidaD": false, "pesticidaE": false}}
```

En este punto lo único que podríamos hacer, sería registrar un nuevo dron. Para ello, introducir password y hacer click en Registrar nuevo dron



The screenshot shows the TFE UNIR app interface. At the top, there is a header with a logo and the text "TFE UNIR by Pablo Cosío". Below the header, there are two sections: "Empresa" (with an image of a quadcopter) and "Propietario" (with an image of a green field). A dropdown menu shows "Empresa existente". Below these sections is a form with fields for "Password" (containing "....") and a button labeled "Registrar nuevo dron". A link at the bottom left says "Listar todos los trabajos pendientes por asignar". A sidebar on the left shows a count of 1 and a link to an IPFS URL.

En este caso ya no se pide el nombre de la empresa, al ser una ya existente, únicamente añadimos el nuevo dron:

My LWC App https://bafybeibrcstau5rr7z7fr2n... Remix - Ethereum IDE

127.0.0.1:3001 Error

## TFE UNIR by Pablo Cosío

Empresa



Propietario



Empresa existente

### Introduzca los datos para poder registrar el Dron

Dron -> Altura vuelo máxima:  
5000

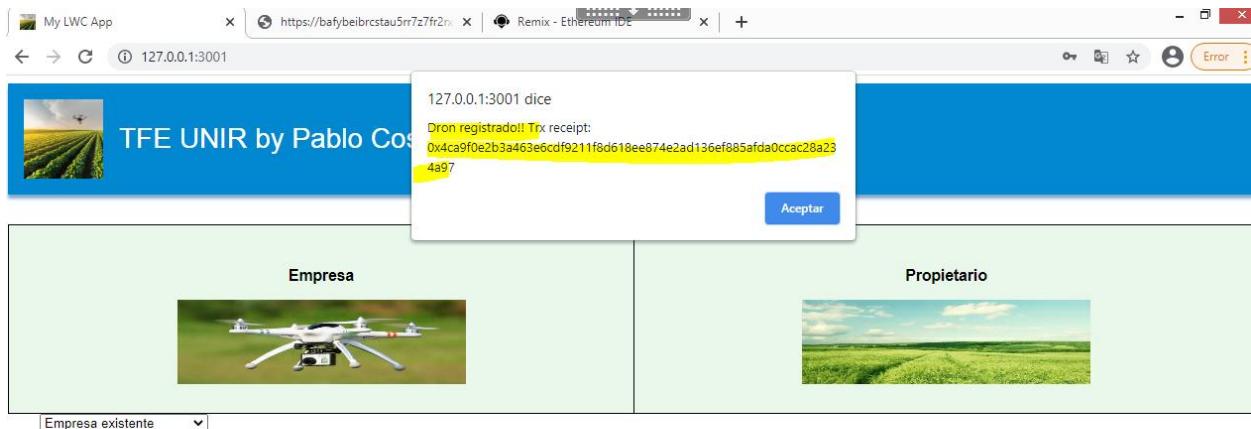
Dron -> Altura vuelo mínima:  
500

Dron -> coste:  
60

Dron -> Pesticidas que puede suministrar:  
Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E

Confirmar

Esperamos que se confirme la transacción y al igual que antes en la consola podemos ver el link que genera ipfs



Una vez registrado el nuevo dron, se vuelve a la pantalla del listado, donde se puede observar todos los drones de la empresa:

My LWC App https://bafybeibrcstau5rr7z7fr2... Remix - Ethereum IDE

127.0.0.1:3001 Error

# TFE UNIR by Pablo Cosío

Empresa



Propietario



Password:

1 <https://ipfs.infura.io/ipfs/QmReEwxr328CVpkdUbgp1My64kimszKgYM8iTTeuvjKdv9>

2 <https://ipfs.infura.io/ipfs/QmZ9bcaShoF8iCNYLpJdhW3ZzGnmqoaAMgTUanm2AzUewJ>

## 6.2. Registro de propietarios y parcelas

Haciendo click en Propietario y luego seleccionando la opción de Registro de nuevo propietario, se despliega el formulario de registro de empresa y parcela. Para la parcela sólo se puede seleccionar un pesticida.

My LWC App https://bafybeibrcstau5rr7z7fr2nk... Remix - Ethereum IDE

127.0.0.1:3001

Error

TFE UNIR by Pablo Cosío

Empresa

Propietario

Registro nueva propietario ▾

### Introduzca los datos para poder registrar el Propietario

Nombre Propietario:  
propietario 17.23

Password:  
....

### Introduzca los datos para poder registrar la parcela

Parcela -> Altura vuelo máxima:  
10000

Parcela -> Altura vuelo mínima:  
100

Parcela -> Pesticida aceptado:  
Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E

Al igual que para las empresas, se crea una nueva cuenta para el propietario, pero **además se le asignan 100 tokens para que la empresa** pueda contratar drones para fumigar sus parcelas y que puedan pagar por tanto con estos tokens.

My LWC App https://bafybeibrcstau5rr7z7fr2n... Remix - Ethereum IDE 127.0.0.1:3001

127.0.0.1:3001 dice  
Cuenta propietario creada:  
**0x5ad10331932e21Fc88EAbA94ca8DA13b31952EF5**

Aceptar

TFE UNIR by Pablo Cosme

Empresa 

Propietario 

Registro nueva propietario ▾

**Introduzca los datos para poder registrar el Propietario**

Nombre Propietario:  
propietario 17:23

Password:  
....

**Introduzca los datos para poder registrar la parcela**

Parcela -> Altura vuelo máxima:  
10000

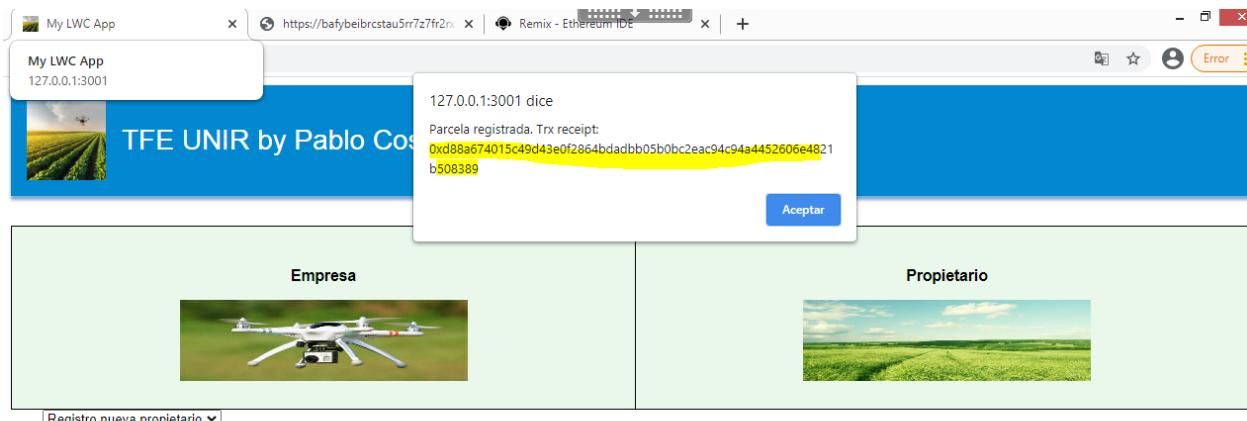
Parcela -> Altura vuelo mínima:  
100

Parcela -> Pesticida aceptado:

Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E

Confirmar

Esperamos a que la transacción se confirme, y al igual que para las empresas podemos ver el link de ipfs generado al almacenar allí el json con la metadata de la parcela

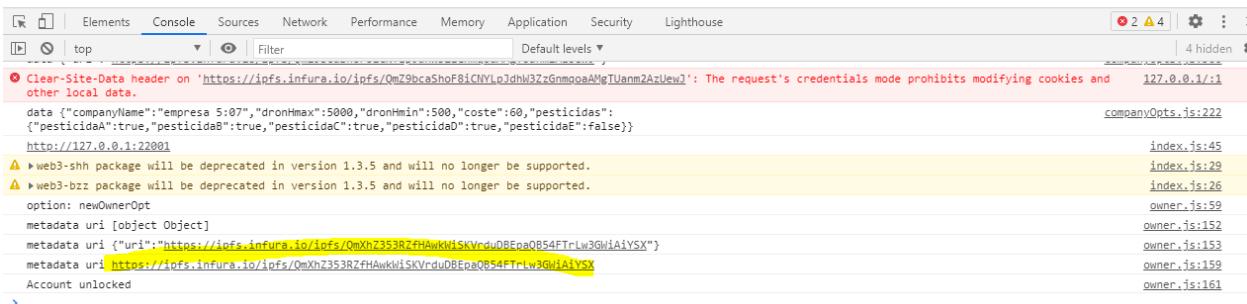


## Introduzca los datos para poder registrar el Propietario

Nombre Propietario:  
propietario 17:23  
Password:  
....

## Introduzca los datos para poder registrar la parcela

Parcela -> Altura vuelo máxima:  
10000  
Parcela -> Altura vuelo mínima:  
100  
Parcela -> Pesticida aceptado:  
Pesticida A  Pesticida B  Pesticida C  Pesticida D  Pesticida E



### 6.3. Menú para los Propietarios

Desde este menú los propietarios podrán listar todas sus parcelas, registrar nuevas, y contratar un dron que se ajuste a las características de la parcela para su fumigación.

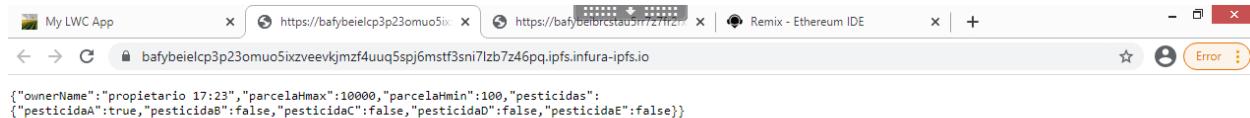
Listado de parcelas:

The screenshot shows a web browser window with three tabs: "My LWC App", "https://bafybeibrcstau5rr7z7fr2x...", and "Remix - Ethereum IDE". The main content area has a blue header with the text "TFE UNIR by Pablo Cosío" and a small drone icon. Below the header, there are two sections: "Empresa" (with an image of a white quadcopter) and "Propietario" (with an image of a green field). At the bottom left is a dropdown menu labeled "Propietario existente" with a dropdown arrow. On the right side, there is a button labeled "Listar Parcelas del propietario".

The screenshot shows a web browser window with three tabs: "My LWC App", "https://bafybeibrcstau5rr7z7fr2x...", and "Remix - Ethereum IDE". The main content area has a blue header with the text "TFE UNIR by Pablo Cosío" and a small drone icon. Below the header, there are two sections: "Empresa" (with an image of a white quadcopter) and "Propietario" (with an image of a green field). At the bottom left is a dropdown menu labeled "Propietario existente" with a dropdown arrow. On the right side, there is a button labeled "Registrar nuevo dron".

The screenshot shows a web browser window with three tabs: "My LWC App", "https://ipfs.infura.io/ipfs/QmXhZ353RZhAwkWiSKVrdubDEpaQB54FTrLw3GWiAiYSX", and "Remix - Ethereum IDE". The main content area has a blue header with the text "TFE UNIR by Pablo Cosío" and a small drone icon. Below the header, there is a section titled "Seleccione la parcela sobre la que desea contratar un dron" with a number "1" and a link "https://ipfs.infura.io/ipfs/QmXhZ353RZhAwkWiSKVrdubDEpaQB54FTrLw3GWiAiYSX". At the bottom, there are two buttons: "Ver datos parcela" and "Ver drones disponibles".

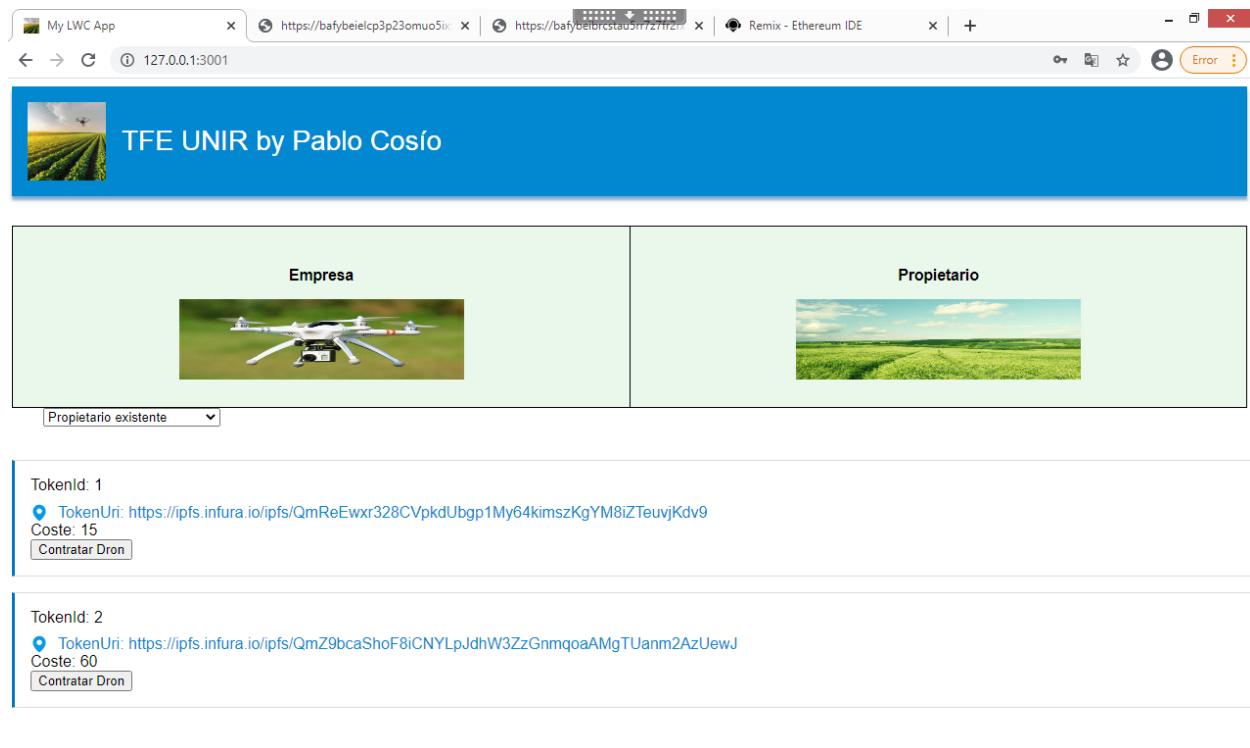
En este menú, el propietario puede ver los datos de la parcela, clickando en el botón se le redirigirá a IPFS para ver el json generado durante el registro del misma.



A screenshot of a browser window showing four tabs. The active tab displays a JSON object:

```
{"ownerName": "propietario 17:23", "parcelaHmax": 10000, "parcelaHmin": 100, "pesticidas": {"pesticidaA": true, "pesticidaB": false, "pesticidaC": false, "pesticidaD": false, "pesticidaE": false}}
```

Haciendo click en Ver drones disponibles se presentará el lisado de todos los drones que cumplen las condiciones de altura y pesticida para poder fumigar la parcela:



The screenshot shows the LWC App interface. At the top, there's a header with a logo and the text "TFE UNIR by Pablo Cosío". Below the header, there are two sections: "Empresa" (left) and "Propietario" (right), each containing an image of a drone. A dropdown menu below the "Empresa" section is set to "Propietario existente".

**Empresa**

**Propietario**

Propietario existente ▾

TokenId: 1  
TokenUri: <https://ipfs.infura.io/ipfs/QmReEwrx328CVpkdUbgp1My64kimszKgYM8iZTeujjKdv9>  
Coste: 15  
[Contratar Dron](#)

TokenId: 2  
TokenUri: <https://ipfs.infura.io/ipfs/QmZ9bcaShoF8iCNYLpJdhW3ZzGnmqoaAMgTUanm2AzUewJ>  
Coste: 60  
[Contratar Dron](#)

Haciendo click en Contratar Dron, se procede a la contratación. Esto genera una transferencia de tokens por el coste del dron, y además añadirá el trabajo pendiente de asignación para la empresa propietaria del dron.

My LWC App | https://bafybeielcp3p23omuo5ic... | https://bafybeibrcstaudir7z7fr2r... | Remix - Ethereum IDE | 127.0.0.1:3001 | Error

127.0.0.1:3001 dice  
Validación propietario de parcela es que el que envía la transacción ->  
OK

Aceptar

TFE UNIR by Pablo Co.

Empresa

Propietario

Propietario existente

TokenId: 1  
TokenUri: https://ipfs.infura.io/ipfs/QmReEwrxr328CVpkdUbgp1My64kimszKgYM8iZTeuvjKdv9  
Coste: 15  
Contratar Dron

TokenId: 2  
TokenUri: https://ipfs.infura.io/ipfs/QmZ9bcaShoF8iCNYLpJdhW3ZzGnmqoaAMgTUanm2AzUewJ  
Coste: 60  
Contratar Dron

My LWC App | https://bafybeielcp3p23omuo5ic... | https://bafybeibrcstaudir7z7fr2r... | Remix - Ethereum IDE | 127.0.0.1:3001 | Error

127.0.0.1:3001 dice  
Se ha realizado el pago correctamente de 60 tokens a la empresa del Dron

Aceptar

TFE UNIR by Pablo Co.

Empresa

Propietario

Propietario existente

TokenId: 1  
TokenUri: https://ipfs.infura.io/ipfs/QmReEwrxr328CVpkdUbgp1My64kimszKgYM8iZTeuvjKdv9  
Coste: 15  
Contratar Dron

TokenId: 2  
TokenUri: https://ipfs.infura.io/ipfs/QmZ9bcaShoF8iCNYLpJdhW3ZzGnmqoaAMgTUanm2AzUewJ  
Coste: 60  
Contratar Dron

Este pantallazo siguiente, sería la confirmación de que el trabajo ha sido añadido correctamente a la cola, y ahora la empresa debería poder asignarle el trabajo de fumigación a

su dron (a partir de aquí las imágenes no corresponden con las direcciones y pruebas anteriores ya que se tuvo que redesplegar el código por un error, pero el flujo no varía).

My LWC App https://bafybeielcp3p23omuo5ix... https://bafybeircsta5irr7z/rzr... Remix - Ethereum IDE Error

127.0.0.1:3001

TFE UNIR by Pablo Cos... 127.0.0.1:3001 dice Job pendiente de aprobación creado 0 Aceptar

Empresa 

Propietario 

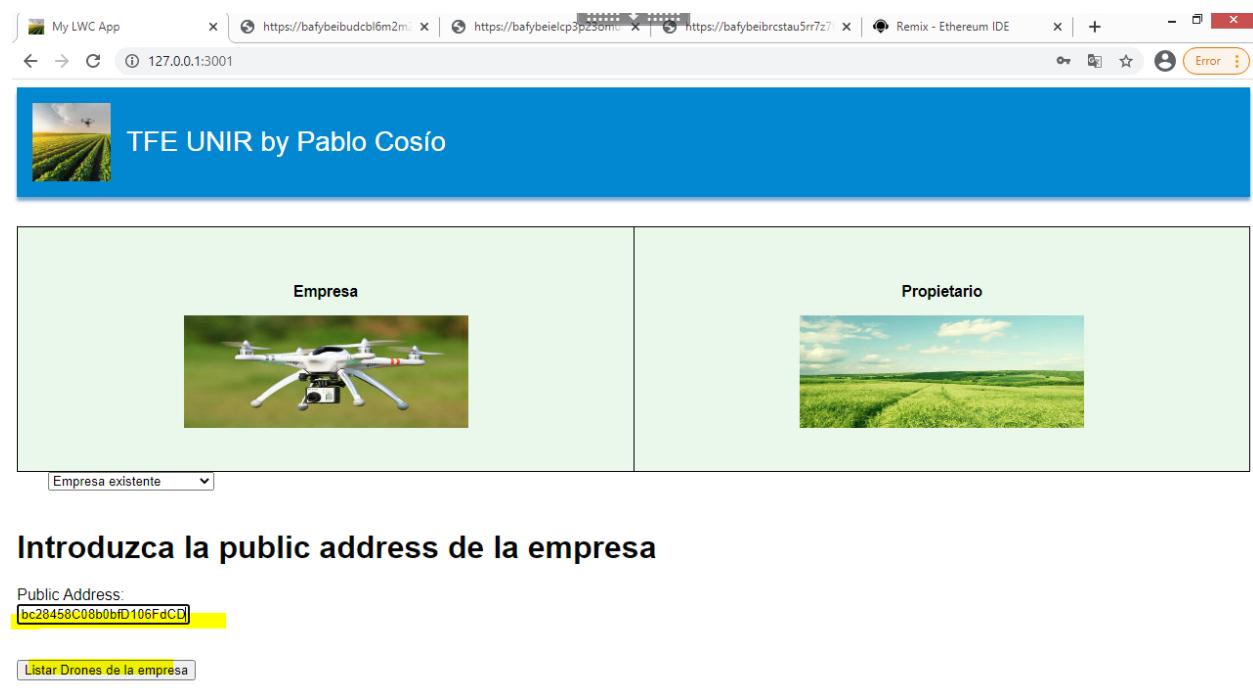
Propietario existente

TokenId: 1  
TokenUri: <https://ipfs.infura.io/ipfs/QmRr1b1j5CFTejpaZZaJ6H1GN1TDutMwhefADbGATFDnv>  
Coste: 20  
Contratar Dron

TokenId: 2  
TokenUri: <https://ipfs.infura.io/ipfs/QmR2Bp3EbFNwpdwoQUmWS3jVCs8fezSLRxP5mFHCWHeSHe>  
Coste: 60  
Contratar Dron

## 6.4. Menú empresa para asignación de trabajos a los drones

Nos vamos a la cuenta de la empresa, para poder listar los trabajos:



TFE UNIR by Pablo Cosío

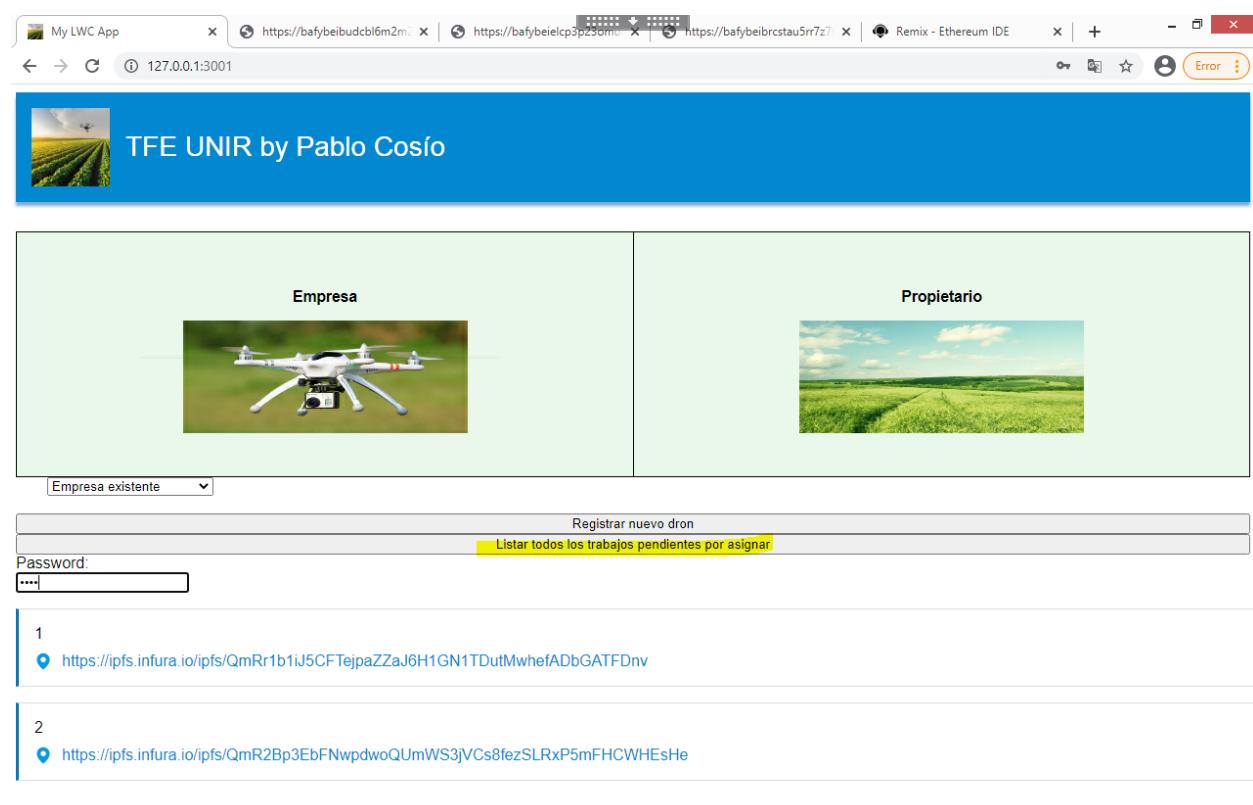
Empresa

Propietario

Empresa existente

bc28458C08b0bD106FdCD

Listar Drones de la empresa



My LWC App

https://bafybeibudcb16m2m... x https://bafybeielcp3p23om... x https://bafybeibrcstau5rr7z... x Remix - Ethereum IDE x + - Error

127.0.0.1:3001

TFE UNIR by Pablo Cosío

Empresa

Propietario

Empresa existente

Registrar nuevo dron

Listar todos los trabajos pendientes por asignar

Password:

1 https://ipfs.infura.io/ipfs/QmRr1b1iJ5CFTejpaZZaJ6H1GN1TDutMwhefADbGATFDnv

2 https://ipfs.infura.io/ipfs/QmR2Bp3EbFNwpdwoQUmWS3jVCs8fezSLRxP5mFHCWHEsHe

Se listarán todos los trabajos pendientes de asignar.

My LWC App | https://bafybeibudcbl6m2m... | https://bafybeielcp3pz30m... | https://bafybeibrcstau5rr7z... | Remix - Ethereum IDE | + | - | Error | 127.0.0.1:3001

## TFE UNIR by Pablo Cosío



Empresa



Propietario

Empresa existente

### Listado de todos los trabajos pendientes por asignar de la empresa

```
JobId: 0  
Propietario parcela: 0x5a7b18F905CDB03F49F574413b250CB690Eceac2  
TokenId parcela: 1  
TokenId dron: 2  
Precio dron: 60
```

Clickando en el elemento se procederá a la aprobación

My LWC App | https://bafybeibudcbl6m2m... | https://bafybeielcp3pz30m... | https://bafybeibrcstau5rr7z... | Remix - Ethereum IDE | + | - | Error | 127.0.0.1:3001

## TFE UNIR by Pablo Cosío



Empresa



Propietario

Empresa existente

127.0.0.1:3001 dice  
Se asignó correctamente el trabajo al Dron

Aceptar

### Listado de todos los trabajos pendientes por asignar de la empresa

```
JobId: 0  
Propietario parcela: 0x5a7b18F905CDB03F49F574413b250CB690Eceac2  
TokenId parcela: 1  
TokenId dron: 2  
Precio dron: 60
```

Y como dice el enunciado del ejercicio, la fumigación es inmediata, por lo que la web captura ese evento y se muestra por pantalla

My LWC App | https://bafybeibudcbl6m2m... | https://bafybeielcp3pz3om... | https://bafybeircstau5rr7z... | Remix - Ethereum IDE | + | - | X

127.0.0.1:3001

127.0.0.1:3001 dice  
Dron finalizó el trabajo. Parcela Fumigada!

Aceptar

TFE UNIR by Pablo Cos

Empresa

Propietario

Empresa existente

**Listado de todos los trabajos pendientes por asignar de la empresa**

JobId: 0  
Propietario parcela: 0x5a7b18F905CDB03F49F574413b250CB690Eceac2  
TokenId parcela: 1  
TokenId dron: 2  
Precio dron: 60

Si volviésemos a la pantalla del listado, nos sale un aviso indicando que no hay trabajos pendientes de asignar

My LWC App | https://bafybeibudcbl6m2m... | https://bafybeielcp3pz3om... | https://bafybeircstau5rr7z... | Remix - Ethereum IDE | + | - | X

127.0.0.1:3001

127.0.0.1:3001 dice  
No hay trabajos pendientes de asignar.

Aceptar

TFE UNIR by Pablo Cos

Empresa

Propietario

Empresa existente

**Listado de todos los trabajos pendientes por asignar de la empresa**

## 7. Conclusiones

Se han visto las dificultades de desarrollar sobre una red como Alastria, con versiones obsoletas de Solidity y con incompatibilidades como las comentadas en el punto 2 referente al gas y que obligan a hacer un replanteamiento de la solución. Se ha visto que una parte importante son las pruebas del desarrollo que se va haciendo en las diferentes testnet como ganache o rinkeby, de tal forma que incluso puede llegar a hacerse un poco pesado el desarrollo, por la lentitud de las pruebas y montaje del entorno, conexiones, etc.

La realización de este trabajo me ha dado conocimiento tanto web (partía de conocimientos muy básicos en la parte de desarrollo frontend) como de solidity y web3. Se ha visto la complejidad de abordar proyectos de este tipo, compatibilidades de versiones, decisiones de funcionamiento, desarrollo de contratos o extensión de ya existentes como OpenZepellin, uso de librerías o contratos de terceros, conectividad, etc. Todo ello ha hecho que por mi parte haya sido muy enriquecedor tanto el curso en general como la propia realización del proyecto que me ha servido para asentar los conocimientos adquiridos durante el curso.