

Trabajo Final de Experto

Experto Universitario en Desarrollo de Aplicaciones Blockchain

Aplicación descentralizada para la fumigación de parcelas
a través de drones

Jesús David Steer Varela

Tutor: Fidel Panigua Diez

2020-2021

Índice

Definición del problema	3
Justificación del uso de Blockchain	4
Análisis y modelo del sistema	6
Descripción del entorno de desarrollo	11
Instrucciones de despliegue	12
Testing de la solución	19
Manual de usuario	22
Conclusiones	33

Definición del problema

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. Los drones y las parcelas serán gestionados mediante tokens no fungibles basados en el estándar ERC 721, la gestión de pagos se hará por medio de un token propio basado en el estándar ERC 20.

Característica de los tokens no fungibles

Drones

Los drones tendrán un identificador único y ascendente, comenzando en 1 y que no puede repetirse, además de la empresa que lo gestiona que será la única que pueda mandar acciones al dron. Deberán almacenar la altura máxima y mínima de vuelo, el coste de operación, así como 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. La operación de fumigación es inmediata y debe lanzar un evento de parcela fumigada con el ID de la parcela.

Parcelas

Las parcelas tendrán un identificador único y ascendente, que comienza en 1 y que no puede repetirse, un propietario, altura máxima y mínima de vuelo permitido y el pesticida aceptado, que va a ser uno de la lista de pesticidas descrita anteriormente.

Operaciones que debe suministrar la plataforma

La solución debe permitir al propietario contratar un dron de la empresa para desinfectar una parcela con un pesticida determinado. Esta operación debe lanzar un evento con el ID del dron que va a realizar la fumigación y el ID de la parcela que se va a fumigar. Otra operación es transferir el pago de la operación realizada desde la cuenta del propietario a la de la empresa utilizando el token ERC20.

Características de la interfaz de usuario

La interfaz web para la empresa que debe permitir registrar los drones y asignarles trabajos. La interfaz web para que los propietarios de las parcelas puedan registrar parcelas y contratar un dron con las características que requiere su parcela y que pueda desplazarse hasta la misma.

Justificación del uso de Blockchain

El uso de blockchain para dar solución al problema está fundamentado en sus principales características, entre ellas se destacan la inmutabilidad, el ser descentralizada y sus altos niveles de seguridad. Para entrar en detalle analizaremos cada una de ellas:

La inmutabilidad permite que la información contenida en una aplicación blockchain no pueda ser alterada, lo que deriva en que los datos y las operaciones no se puedan corromper. Esto genera confianza en dichas operaciones para los diferentes actores que interactúan con la aplicación, permitiendo hacer transacciones con terceros con quienes no tenemos contacto o guardamos una relación, garantizando en que una vez las condiciones del contrato se cumplan el resultado será el esperado. También permite crear el sistema de pagos integrado con la solución como es el caso del token ERC20, el cual es un token de utilidad que será transferido una vez el trabajo sea realizado.

Al ser descentralizada, la aplicación no tiene una institución o persona que genere un control total sobre su operación, evitando así que dichas operaciones puedan ser censuradas o interrumpidas por un ente externo.

La seguridad en las aplicaciones blockchain es muy alta por las características antes mencionadas, pero adicionalmente por el uso de la criptografía para la protección de los datos del sistema, de los usuarios y de las transacciones realizadas.

En el caso específico de este proyecto se decidió hacer uso de blockchain para cumplir el rol que suele tener el backend en las aplicaciones tradicionales. Esto se pudo realizar debido a los requerimientos específicos de este sistema en particular. De esta forma la blockchain a través del código de los contratos inteligentes y su espacio de almacenamiento cumplen con las funciones básicas del backend, como son autenticación, almacenamiento persistente de datos, y almacenamiento de datos de la sesión.

La autenticación se realiza a través del uso de las llaves publica de los usuarios registrados en el nodo al que conectamos. También es posible conectar con una wallet externa como Metamask, pero por limitaciones del entorno de desarrollo utilizado se omitió esta característica.

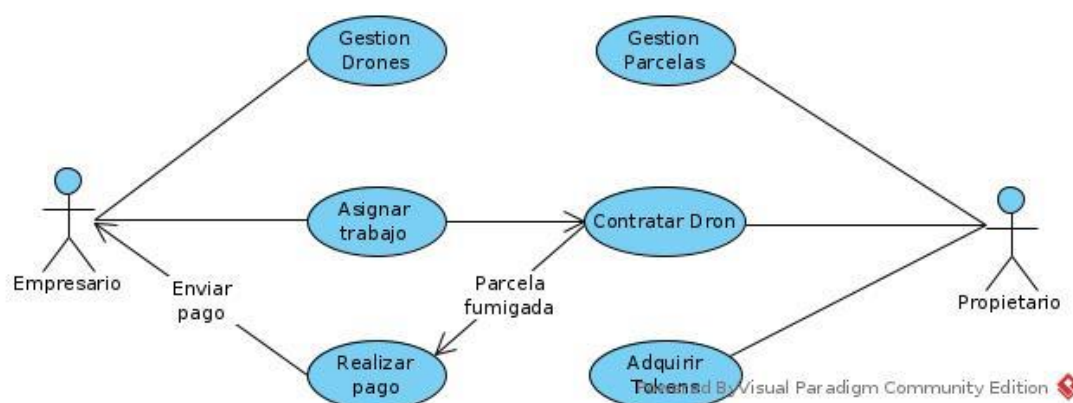
La persistencia de datos que normalmente suele realizar una base de datos o solución alternativa dentro de una aplicación tradicional se realiza en este caso a través del registro de dichos datos dentro del contrato, lo cual le proporciona seguridad y transparencia.

El almacenamiento de datos de sesión es realizado por el mismo navegador web, ya que el sistema desarrollado usa el enfoque de “One page application”, lo que permite que la información se mantenga cargada en la memoria ya que el sistema no realiza postback o cambios entre páginas. Pero de igual forma es posible hacer

una recarga completa del sitio sin perder el estado actual porque todas las transacciones se ejecutan de forma inmediata en la cadena de bloques.

Análisis y modelo del sistema

Diagrama de casos de uso

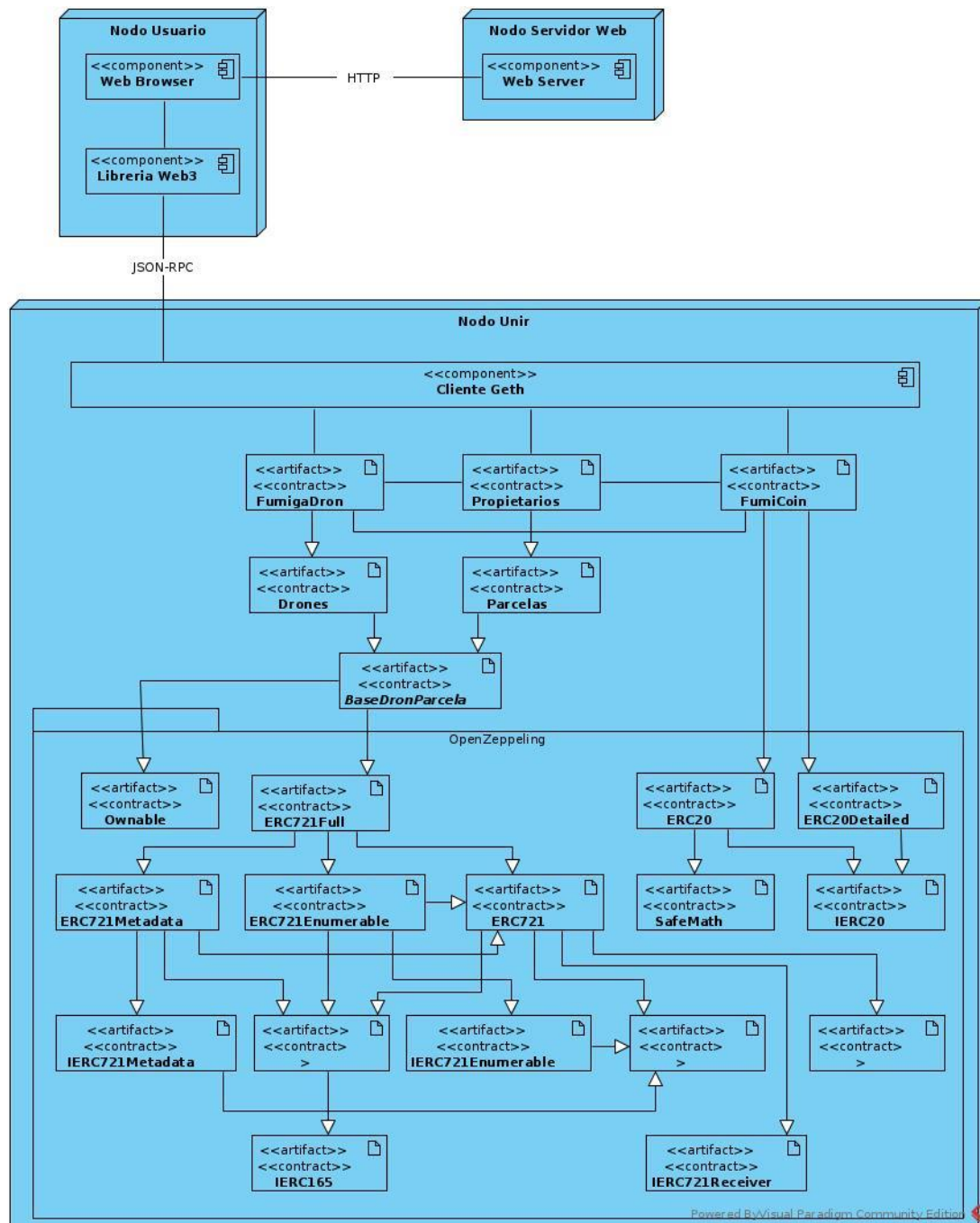


Siguiendo los requerimientos planteados se identifican dos actores en el sistema. El empresario y el propietario.

El empresario estará encargado de la gestión de los drones, que básicamente hace referencia a su creación dentro del sistema. Así mismo asignar trabajos a los drones, con la previa solicitud del trabajo por parte del propietario, y una vez completado el trabajo se realizara la transferencia de tokens FUMI de la billetera del propietario a la del empresario. Esta confirmación será realizada por el evento "TrabajoConfirmado".

El propietario tendrá la posibilidad de crear parcelas con sus características, así como contratar un dron que cumpla con las características propias de la parcela. Una vez realizada la solicitud será confirmada por el evento "TrabajoSolicitado", y se aprobará el pago del monto de tokens, una vez la tarea sea realizada correctamente.

Diagrama de despliegue



El diagrama de despliegue nos muestra la interacción entre los distintos nodos, y sus componentes. Los nodos que interactúan son el nodo del usuario, el servidor web, y el nodo unir que será el que esté conectado con la red de Alastria.

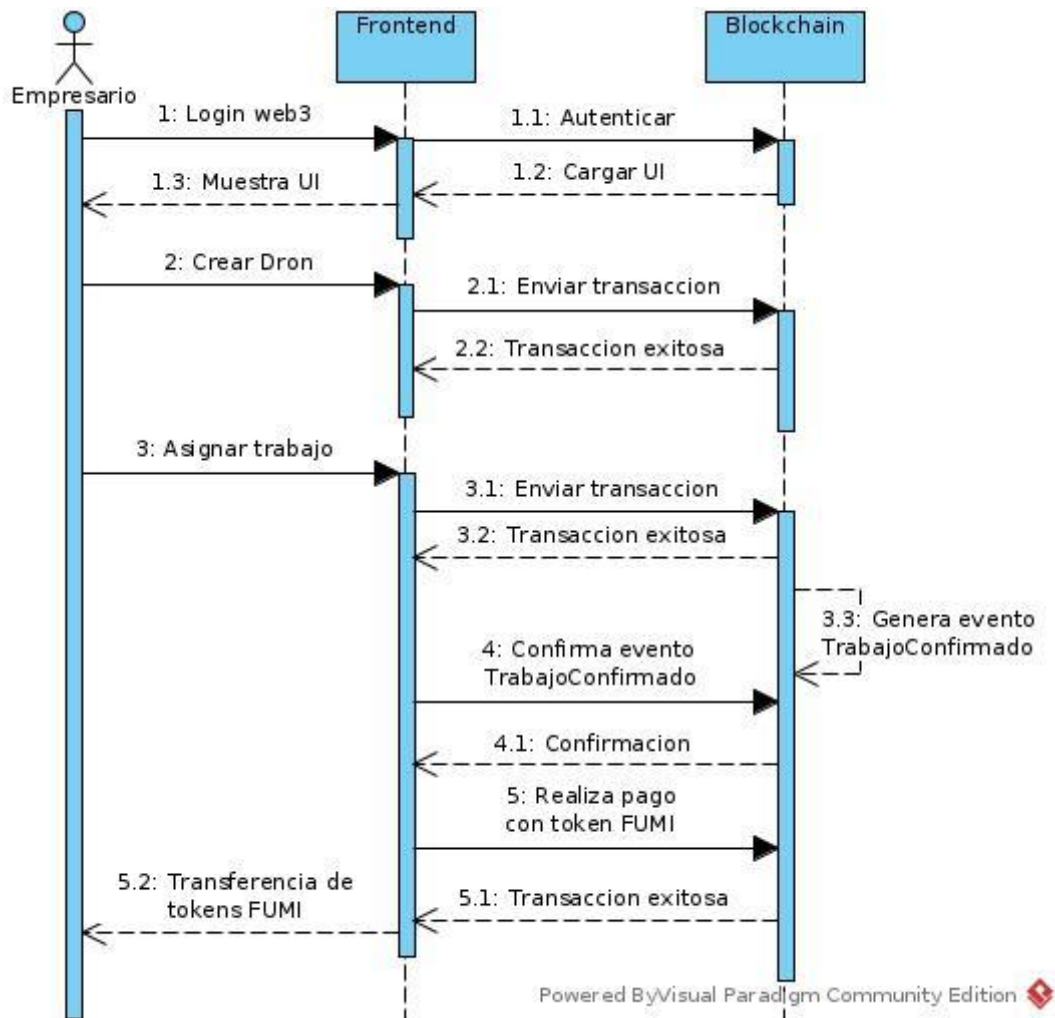
El nodo usuario es el equipo desde el cual se conecta el usuario (empresario o propietario) para conectar con la aplicación descentralizada.

El nodo servidor web es el equipo que pone a disposición la página web para ser accedida por los usuarios. Nótese que la conexión a la blockchain no se realiza desde el servidor, sino desde el nodo usuario, ya que es la librería “Web3” es quien se encarga de enviar mensajes al nodo de blockchain al que se ha conectado.

Finalmente tenemos al nodo de Unir, que es el nodo utilizado para conectar con la red de Alastria, debido a que para efectos del trabajo se ha habilitado este nodo para poder interactuar con la cadena de bloques.

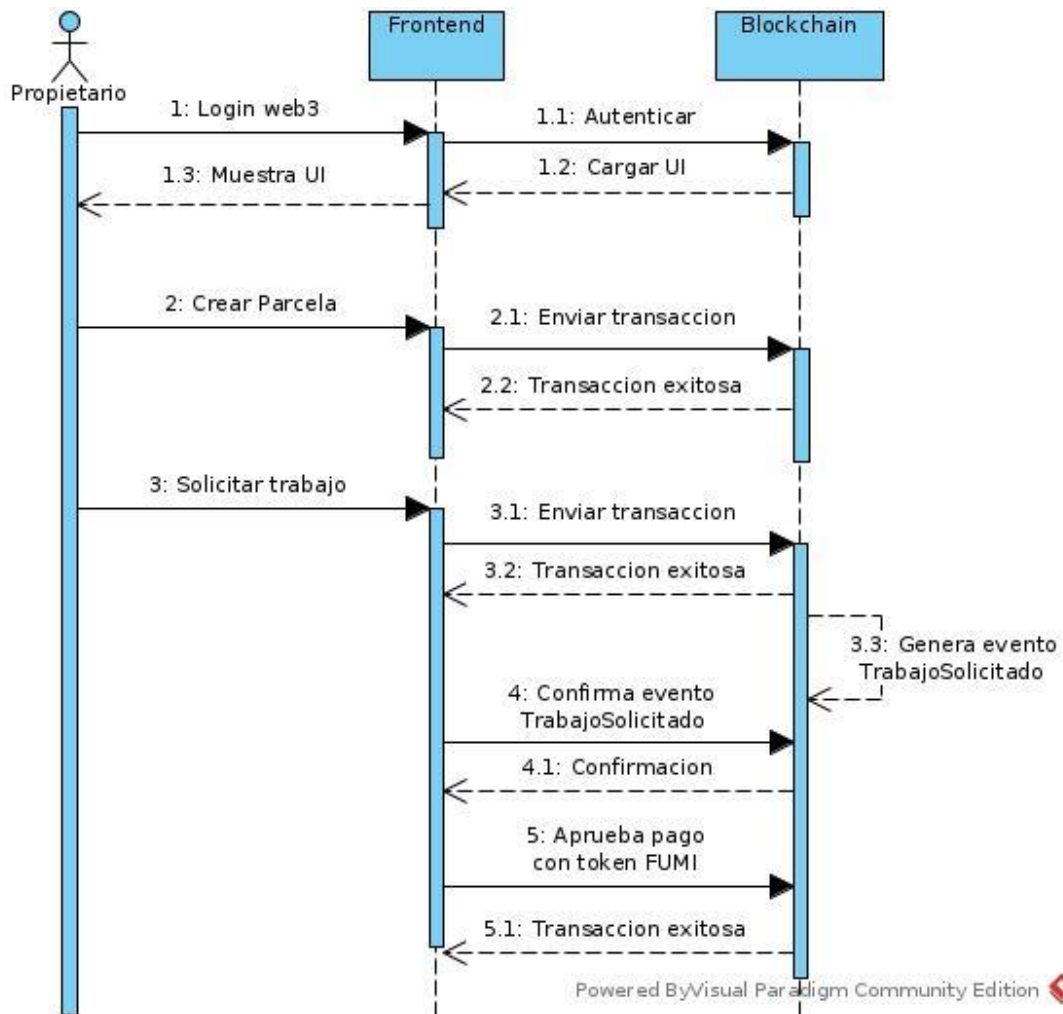
También se aprecia en el diagrama los contratos desplegados en la blockchain, como son FumigaDron.sol para implementar la funcionalidad utilizada por el empresario, el contrato Propietarios.sol que implementa la funcionalidad utilizada por los usuarios propietarios y finalmente el contrato FumigaCoin.sol que hereda su funcionalidad del contrato ERC20. De igual forma se muestra las relaciones de herencia que hay entre estos contratos y otros creados con el objetivo de implementar la reutilización del código, el ejemplo más claro puede encontrarse en el contrato BaseDronParcela.sol que implementa funcionalidades que son compartidas entre los contratos Dron.sol y Parcela.sol, y que hereda del contrato ERC721, implementando la funcionalidad de este estándar para los tokens no fungibles asociados al dron y a la parcela. Así mismo se separan las funciones de los contratos entre las operaciones del dron que son implementadas en el contrato Dron.sol, las funciones relacionadas a la parcela que se almacenan en el contrato Parcela.sol.

Diagrama de secuencia para el empresario



El empresario interactúa con la blockchain a través de la secuencia acciones que se muestra en el diagrama. Se pueden apreciar como interactúa el usuario con el frontend, y este a su vez envía las transacciones a la blockchain.

Diagrama de secuencia para los propietario



De la misma forma que en el caso del empresario, este diagrama muestra la interacción del usuario propietario con el frontend, que envía las transacciones hacia la blockchain.

Descripción del entorno de desarrollo



Sitio oficial

<https://www.trufflesuite.com/>

Se eligió la suite de truffle como herramienta principal para el desarrollo de toda la aplicación. En primer instancia se descargó la truffle box “Webpack”, para tener una estructura base de la aplicación, que incluye un servidor web, una interfaz HTML sencilla, y un archivo de JavaScript con una configuración por defecto para la conexión con la blockchain a través de la librería Web3. También proporciona una integración con los artefactos usados al momento de compilar y desplegar los contratos, para así obtener su dirección de despliegue en la red y una copia del ABI necesario para lanzar las operaciones a la blockchain.

De igual forma fueron utilizadas otras herramientas de la suite como Ganache, entorno sandbox de develop y las herramientas para testings.

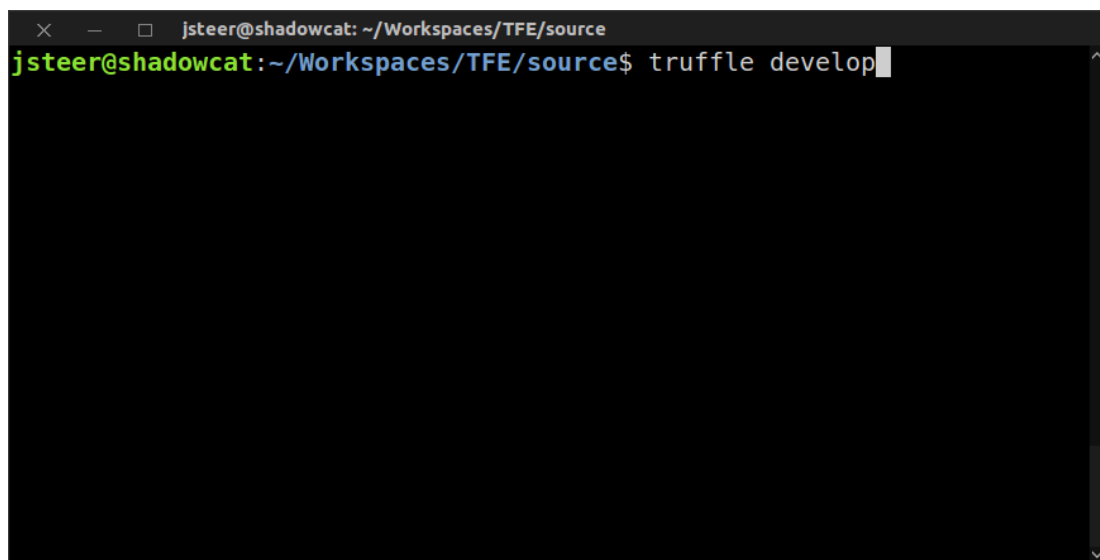
Instrucciones de despliegue

Entorno de desarrollo local

Truffle

Una vez se tengan instaladas y configuradas las dependencias del proyecto enunciadas en la sección “Descripción del entorno de desarrollo”. Se debe abrir una ventana de consola de comandos y acceder a la carpeta “source” ubicada en el directorio raíz del proyecto. Allí se ejecutara el comando:

```
truffle develop
```



Truffle iniciara un sandbox, que incluye una blockchain y 10 cuentas que serán utilizados para ejecutar nuestros contratos inteligentes.

A continuación se debe desplegar los contratos con el comando:

```
migrate --reset
```

```
jsteer@shadowcat: ~/Workspaces/TFE/source
(3) 3b301e3539d08cc852f91a702c2469390f5bccc0ccb3c692991c0939b7f9bac0
(4) baa8ec9b7532e56102efd98a7faed719a6061ad305778c8b55cbf382905ee92f
(5) c1b4cdb0aa470c90f34c899562391e19cbaa045f120edc91dbd0f5485d51f888
(6) 67a1d5b3f14eed75d26216da214264b45ce48d1436eba0f2c73e3e8e261351d0
(7) 853adeb4384eb9cc8ddeb7aba93da91bef20155a37e1fbf9e8b042d79eab33cf
(8) e1fed8ba693a0be8dd57535303527823ad7126e95a99d9a2189cbef83cdfb87c
(9) c9d590a9d34cd2a7ea6348ce3fa899c00829ae23538237093a9a674fcb186e1f

Mnemonic: cherry fluid lake volume isolate vehicle act grief ski coil m
odify lounge

△ Important △ : This mnemonic was created for you by Truffle. It is n
ot secure.
Ensure you do not use it on production blockchains, or else you risk lo
sing funds.

truffle(develop)> migrate --reset
```

Este comando realizara una compilación de los contratos si es necesaria, y ejecutara el despliegue siguiendo el orden establecido en los ficheros de migraciones.

```
jsteer@shadowcat: ~/Workspaces/TFE/source
> gas used:          2866815 (0x2bbe7f)
> gas price:         20 gwei
> value sent:        0 ETH
> total cost:        0.0573363 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost:        0.15891374 ETH

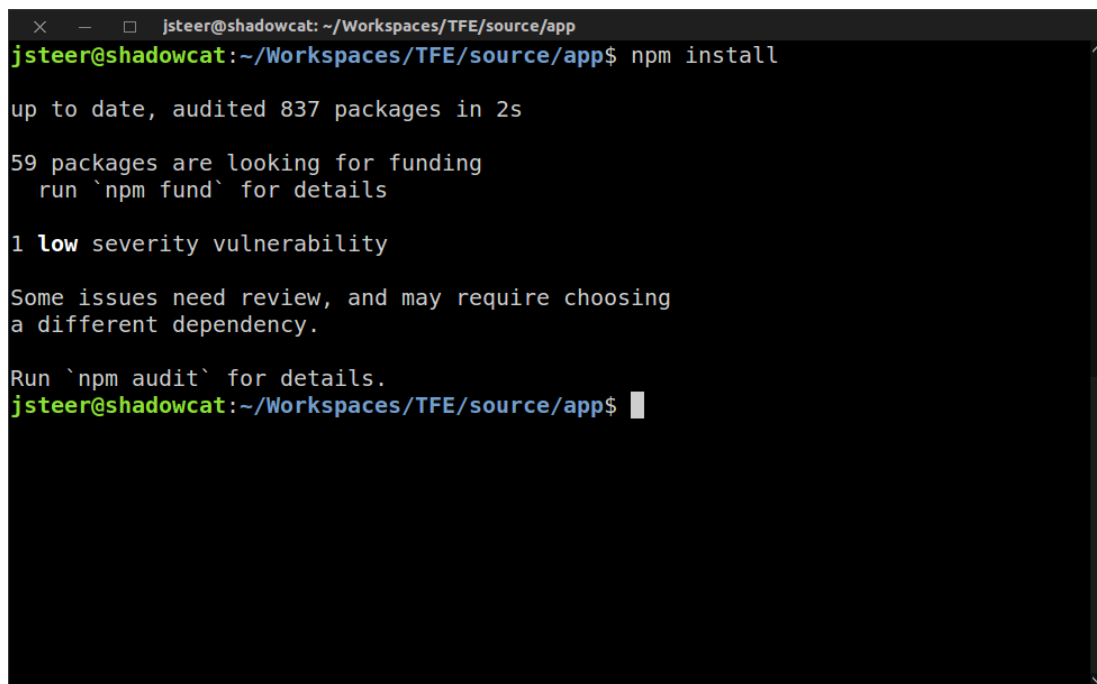
Summary
=====
> Total deployments: 4
> Final cost:        0.16235492 ETH
```

Una vez terminado el proceso la blockchain quedara a disposición para conectarnos, y hacer uso del aplicativo. Se debe mantener esta consola de comandos en ejecución.

Webpack

Para ejecutar el servidor web abrimos una nueva consola de comandos y navegamos a la carpeta, “source/app” ubicada en el directorio raíz de nuestro proyecto, y ejecutamos el comando:

```
npm install
```

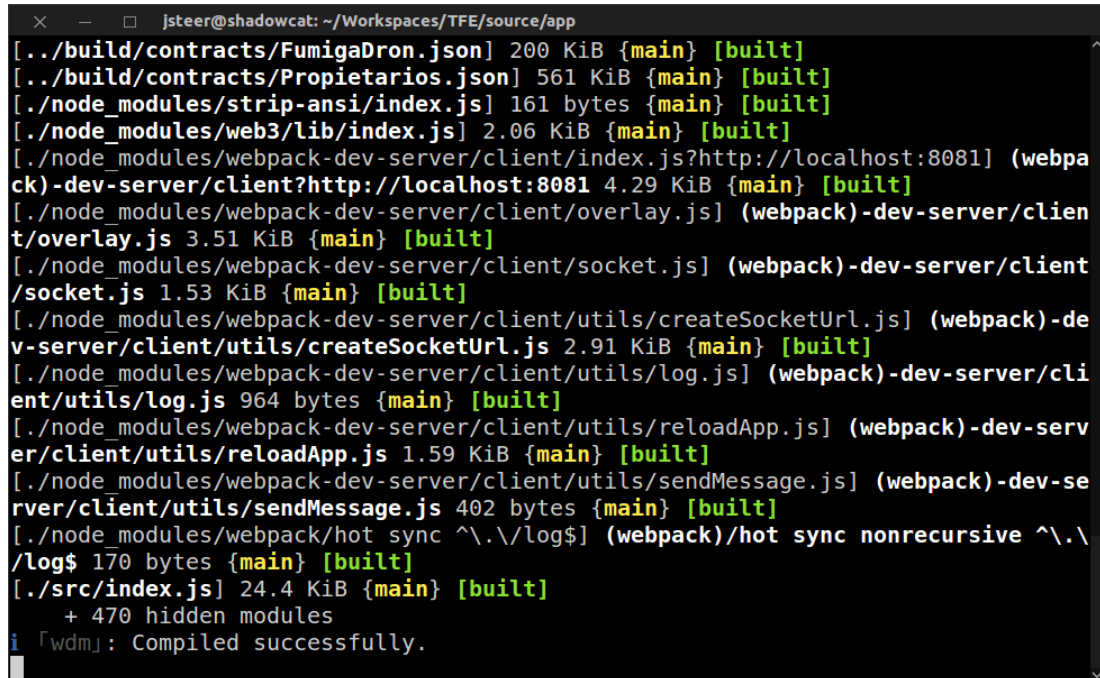
A terminal window with a dark background. The title bar shows 'jsteer@shadowcat: ~/Workspaces/TFE/source/app'. The prompt is 'jsteer@shadowcat:~/Workspaces/TFE/source/app\$'. The command 'npm install' has been executed. The output shows 'up to date, audited 837 packages in 2s', '59 packages are looking for funding', '1 low severity vulnerability', and a message about reviewing issues. The prompt is now 'jsteer@shadowcat:~/Workspaces/TFE/source/app\$' with a cursor.

```
jsteer@shadowcat: ~/Workspaces/TFE/source/app
jsteer@shadowcat:~/Workspaces/TFE/source/app$ npm install
up to date, audited 837 packages in 2s
59 packages are looking for funding
  run `npm fund` for details
1 low severity vulnerability
Some issues need review, and may require choosing
a different dependency.
Run `npm audit` for details.
jsteer@shadowcat:~/Workspaces/TFE/source/app$
```

Se espera a que el sistema de manejo de paquetes de node se encargue de instalar las dependencias necesarias para la ejecución del servidor.

El siguiente paso es ejecutar el servidor web a través del siguiente comando:

```
npm run dev
```

A terminal window with a dark background and light-colored text. The title bar shows 'jsteer@shadowcat: ~/Workspaces/TFE/source/app'. The terminal output lists various files and their sizes, all marked as '[built]'. The files include JSON files, JavaScript files, and a directory. The output ends with a message: 'i [wdm]: Compiled successfully.'

```
jsteer@shadowcat: ~/Workspaces/TFE/source/app
[../build/contracts/FumigaDron.json] 200 KiB {main} [built]
[../build/contracts/Propietarios.json] 561 KiB {main} [built]
[./node_modules/strip-ansi/index.js] 161 bytes {main} [built]
[./node_modules/web3/lib/index.js] 2.06 KiB {main} [built]
[./node_modules/webpack-dev-server/client/index.js?http://localhost:8081] (webpack)-dev-server/client?http://localhost:8081 4.29 KiB {main} [built]
[./node_modules/webpack-dev-server/client/overlay.js] (webpack)-dev-server/client/overlay.js 3.51 KiB {main} [built]
[./node_modules/webpack-dev-server/client/socket.js] (webpack)-dev-server/client/socket.js 1.53 KiB {main} [built]
[./node_modules/webpack-dev-server/client/utils/createSocketUrl.js] (webpack)-dev-server/client/utils/createSocketUrl.js 2.91 KiB {main} [built]
[./node_modules/webpack-dev-server/client/utils/log.js] (webpack)-dev-server/client/utils/log.js 964 bytes {main} [built]
[./node_modules/webpack-dev-server/client/utils/reloadApp.js] (webpack)-dev-server/client/utils/reloadApp.js 1.59 KiB {main} [built]
[./node_modules/webpack-dev-server/client/utils/sendMessage.js] (webpack)-dev-server/client/utils/sendMessage.js 402 bytes {main} [built]
[./node_modules/webpack/hot sync ^\\.\\.\/log$] (webpack)/hot sync nonrecursive ^\\.\\.\/log$ 170 bytes {main} [built]
[./src/index.js] 24.4 KiB {main} [built]
+ 470 hidden modules
i [wdm]: Compiled successfully.
```

Una vez se compilen los assets, se ejecutara el servidor web el cual estará esperando peticiones del navegador. El siguiente paso es abrir una ventana de cualquier navegador y acceder a la dirección:

```
http://localhost:8080/
```

Despliegue en otras redes

En caso de utilizar otra red blockchain para ejecutar los contratos el primer paso es preparar el nodo para que truffle pueda hacer uso de él, asegurándose de tener la cuenta configurada como “from” en el archivo de configuraciones “truffle-config.js” desbloqueada para poder realizar el despliegue.

```
module.exports = {  
  networks: {  
    develop: {  
      port: 8545  
    },  
    ganache: {  
      host: "localhost",  
      port: 7545,  
      network_id: "5777"  
    },  
    alastria_testnet: {  
      host: "127.0.0.1",  
      port: 22001,  
      network_id: "*",  
      gas: 0x6cff0e28,  
      gasPrice: 0x0,  
      from: "0x74d4c56d8dcbc10a567341bfac6da0a8f04dc41d"  
    },  
    alastria: {  
      host: "10.141.8.11",  
      port: 8545,  
      network_id: "*",  
      gas: 0x29b68b6f,  
      gasPrice: 0x0,  
      from: "0xbc869c21d631e122d35789942a573241ec04d2e4"  
    },  
  },  
},
```


Luego debemos ejecutar una de las siguientes instrucciones dependiendo de la red a la que queramos conectar:

```
truffle migrate --network ganache --reset
truffle migrate --network alastria_testnet -reset
truffle migrate --network alastria -reset
```

```
jsteer@shadowcat:~/Workspaces/TFE/source$ truffle migrate --network alastria_testnet^
--reset

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

Starting migrations...
=====
> Network name:      'alastria_testnet'
> Network id:        9535753591
> Block gas limit: 45860129760 (0xaad7a0be0)

1_initial_migration.js
=====

  Replacing 'Migrations'
  -----
  > transaction hash:    0xd90cc737e874b08e16479fc800287cf3eb17fa81fb4e494d67dfc35d
b768cebc
  > Blocks: 0           Seconds: 0
  > contract address:    0x4C16D6905a30bE20a3b9Ab71F61A26534417E2e6
  > block number:        16566
  > block timestamp:     1616160019
  > account:             0x74d4C56d8dcbC10A567341bFac6DA0A8F04DC41d
  > balance:             0
  > gas used:            196887 (0x30117)
```

En la imagen de ejemplo se muestra el despliegue sobre la test de Alastria que se ejecuta a través de una máquina virtual.

Despliegue en la red de Alastria

Siguiendo los pasos enunciados en la sección “Despliegue en otras redes” se realizó el despliegue de los smart contract sobre la red de Alastria, como se muestra en las imágenes a continuación. Primero se ejecutó el comando de despliegue.

```
truffle migrate --network alastria -reset
```

```
ubuntu@testenv:~/fumigadron/source
ubuntu@testenv:~/fumigadron/source$ truffle migrate --network alastria

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

Starting migrations...
=====
> Network name:      'alastria'
> Network id:       83584648538
> Block gas limit:  699999958 (0x29b926d6)

> 2_deploy_fumigadron.js
=====

Deploying 'FumiCoin'
-----
> transaction hash:  0x5048c1abf320eec77261c24a66914defb9cbefde951218b737d278983f24c15d
> Blocks: 0         Seconds: 0
> contract address: 0x31cc8B760a70c8A79a907e36e391f7B6640e8ad0
> block number:     65098917
> block timestamp:  1616249500
> account:          0xbc869c21d631E122d35789942A573241ec04D2E4
> balance:          0
> gas used:         1698299 (0x19e9fb)
> gas price:        0 gwei
> value sent:       0 ETH
> total cost:       0 ETH

Deploying 'Propietarios'
-----
> transaction hash:  0x8382f1077671656a1bb93aefc62a8f12ececbb41515f3e67992a12b57742937d
> Blocks: 3         Seconds: 4
> contract address: 0x8Fb5fc5a492a028A684A1a3b7F200d9D3dA26c9D
> block number:     65098920
> block timestamp:  1616249503
```

```
ubuntu@testenv:~/fumigadron/source
> Blocks: 3         Seconds: 4
> contract address: 0x8Fb5fc5a492a028A684A1a3b7F200d9D3dA26c9D
> block number:     65098920
> block timestamp:  1616249503
> account:          0xbc869c21d631E122d35789942A573241ec04D2E4
> balance:          0
> gas used:         4220661 (0x4066f5)
> gas price:        0 gwei
> value sent:       0 ETH
> total cost:       0 ETH

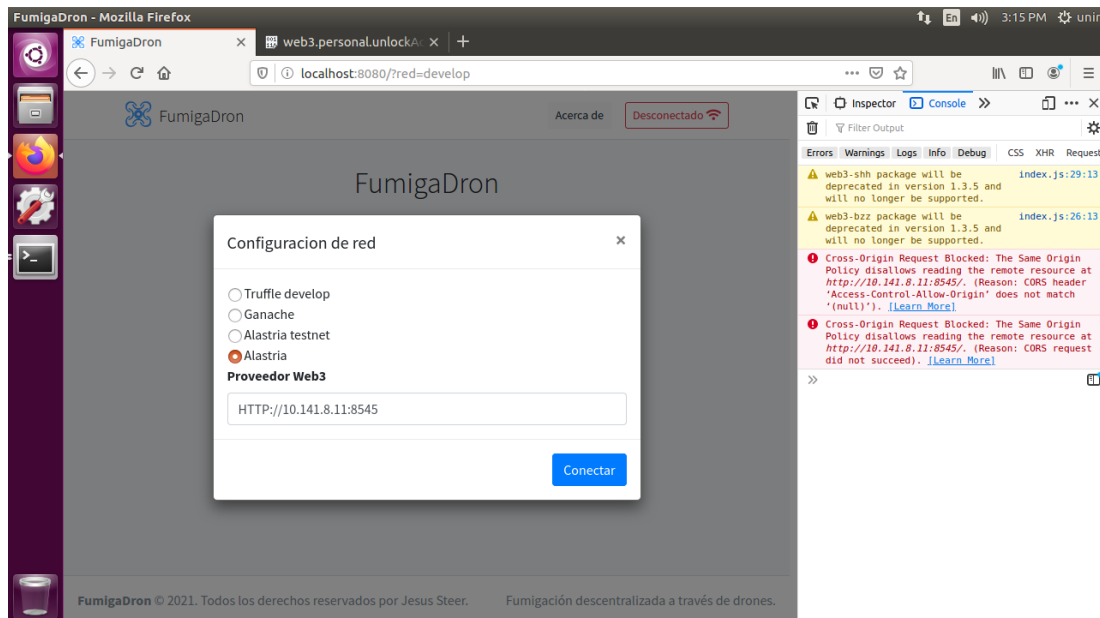
Deploying 'FumigaDron'
-----
> transaction hash:  0x0da9119ab766d5ebd056d5a9755d2a250129388d8ea8990d19c8944984da2ef5
> Blocks: 0         Seconds: 0
> contract address: 0x2602615d41c8cE464dfF34ed4E81cC1675918fEb
> block number:     65098923
> block timestamp:  1616249506
> account:          0xbc869c21d631E122d35789942A573241ec04D2E4
> balance:          0
> gas used:         3362945 (0x335081)
> gas price:        0 gwei
> value sent:       0 ETH
> total cost:       0 ETH

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

Summary
=====
> Total deployments: 3
> Final cost:       0 ETH

ubuntu@testenv:~/fumigadron/source$
```

El despliegue se hizo de forma exitosa. Sin embargo no fue posible conectarse a través del aplicativo web por las restricciones de CORS de lado del nodo habilitado por Alastria para esta labor, por tanto se deja constancia de este suceso con la siguiente imagen como evidencia.

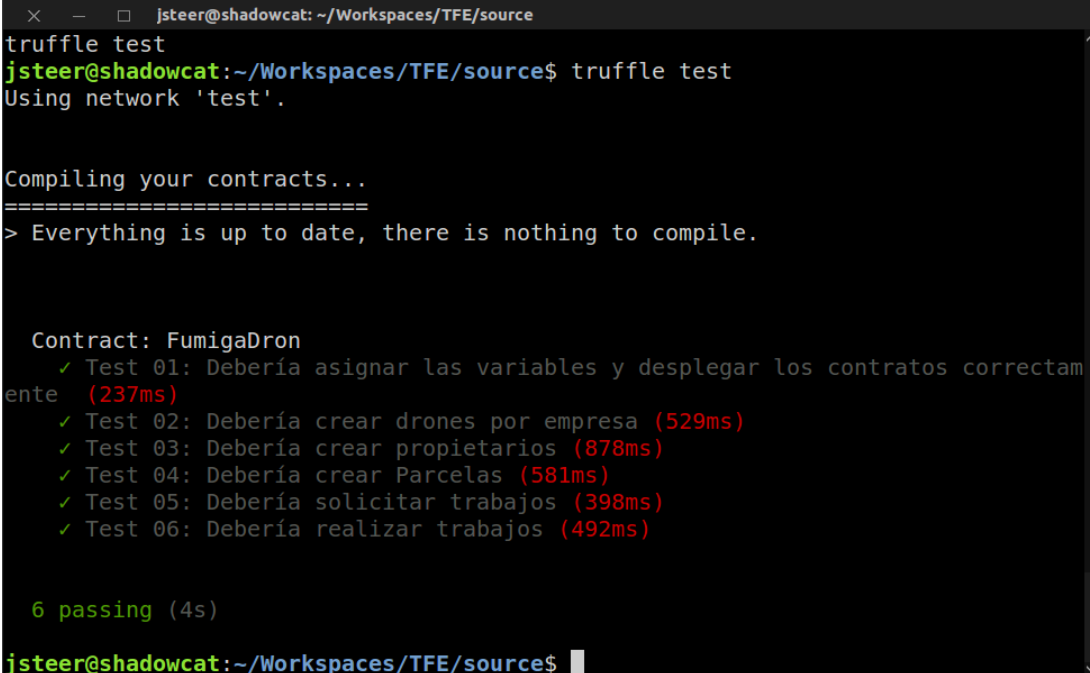


Testing de la solución

Para mejorar la calidad del código, evitar errores y entregar una aplicación con buenos estándares de seguridad se creó una batería de pruebas que contempla el “camino feliz” de la misma. Estas pruebas se encuentran en el fichero “test/fumigadron.js”, y están agrupadas en temas principales que se listan a continuación:

- ▶ Test 01: Debería asignar las variables y desplegar los contratos correctamente
- ▶ Test 02: Debería crear drones por empresa
- ▶ Test 03: Debería crear propietarios
- ▶ Test 04: Debería crear Parcelas
- ▶ Test 05: Debería solicitar trabajos
- ▶ Test 06: Debería realizar trabajos

Estos grupos de pruebas contienen 72 operaciones con la blockchain y 56 validaciones de resultados (asserts), de los cuales 12 están relacionados con eventos emitidos al realizar transacciones. Adicionalmente los contratos incluyen 11 validaciones de tipo “require” para comprobar parámetros anómalos en las funciones.



```
jsteer@shadowcat: ~/Workspaces/TFE/source
truffle test
jsteer@shadowcat:~/Workspaces/TFE/source$ truffle test
Using network 'test'.

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

Contract: FumigaDron
  ✓ Test 01: Debería asignar las variables y desplegar los contratos correctamente (237ms)
  ✓ Test 02: Debería crear drones por empresa (529ms)
  ✓ Test 03: Debería crear propietarios (878ms)
  ✓ Test 04: Debería crear Parcelas (581ms)
  ✓ Test 05: Debería solicitar trabajos (398ms)
  ✓ Test 06: Debería realizar trabajos (492ms)

6 passing (4s)
jsteer@shadowcat:~/Workspaces/TFE/source$
```

Se empleo la aplicación “Solidity Coverage” para medir el porcentaje de cobertura de estos test. Como se puede apreciar en la siguiente grafica los contratos con mejor cobertura fueron los realizados durante el desarrollo de este trabajo, esto es debido a que las funciones de las librerías de “OpenZeppelin” no fueron incluidas dentro de las baterías de prueba por tanto es entendible que tengan un porcentaje de cobertura menor.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	82.19	46.15	78.13	82.67	
BaseDronParcela.sol	66.67	0	75	66.67	19,20
Dron.sol	83.33	50	60	83.33	26,30
FumiCoin.sol	100	100	100	100	
FumigaDron.sol	75	0	75	75	24,25
Parcela.sol	100	50	100	100	
Propietarios.sol	80.56	60	76.92	81.58	... 72,76,77,78
contracts/OpenZeppelin/introspection/	75	50	66.67	75	
ERC165.sol	75	50	66.67	75	41
IERC165.sol	100	100	100	100	
contracts/OpenZeppelin/math/	18.75	8.33	20	18.75	
SafeMath.sol	18.75	8.33	20	18.75	... 42,44,62,63
contracts/OpenZeppelin/ownership/	36.36	25	42.86	41.67	
Ownable.sol	36.36	25	42.86	41.67	... 63,71,72,73
contracts/OpenZeppelin/token/ERC20/	18.18	5	18.75	18.18	
ERC20.sol	13.16	5	16.67	13.16	... 203,207,209
ERC20Detailed.sol	50	100	25	50	26,33,40
IERC20.sol	100	100	100	100	
contracts/OpenZeppelin/token/ERC721/	22.73	7.5	27.27	21.98	
ERC721.sol	18.37	10	22.22	18.37	... 325,326,327
ERC721Enumerable.sol	28.57	0	37.5	26.67	... 146,147,148
ERC721Full.sol	100	100	100	100	
ERC721Metadata.sol	27.27	0	16.67	25	... 70,80,83,84
IERC721.sol	100	100	100	100	
IERC721Enumerable.sol	100	100	100	100	
IERC721Full.sol	100	100	100	100	
IERC721Metadata.sol	100	100	100	100	
IERC721Receiver.sol	100	100	100	100	
contracts/OpenZeppelin/utils/	0	100	0	0	
Address.sol	0	100	0	0	16,24,25
All files	41.18	18.27	44.33	41.22	

> Istanbul reports written to ./coverage/ and ./coverage.json
> solidity-coverage cleaning up, shutting down ganache server

Una última prueba realizada fue la ejecución del programa “Solium” para verificar el formato de los contratos y corregir cualquier error o imprecisión identificada. Como se aprecia en la imagen, los contratos no contienen errores de formato.

```
(mythril) jsteer@shadowcat:~/Workspaces/TFE/source$ solium -f contracts/FumiCoin.sol
No issues found.

(mythril) jsteer@shadowcat:~/Workspaces/TFE/source$ solium -f contracts/BaseDronParcela.sol
No issues found.

(mythril) jsteer@shadowcat:~/Workspaces/TFE/source$ solium -f contracts/Dron.sol
No issues found.

(mythril) jsteer@shadowcat:~/Workspaces/TFE/source$ solium -f contracts/Parcela.sol
No issues found.

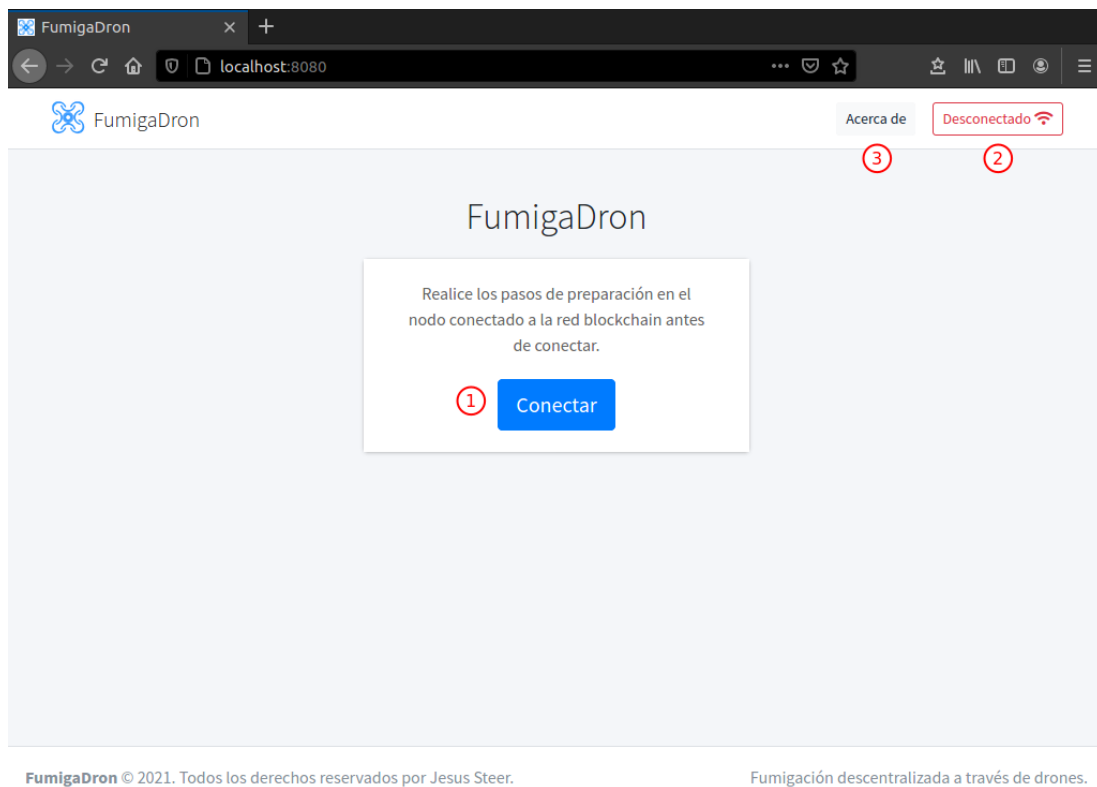
(mythril) jsteer@shadowcat:~/Workspaces/TFE/source$ solium -f contracts/FumigaDron.sol
No issues found.

(mythril) jsteer@shadowcat:~/Workspaces/TFE/source$ solium -f contracts/Propietarios.sol
No issues found.
```

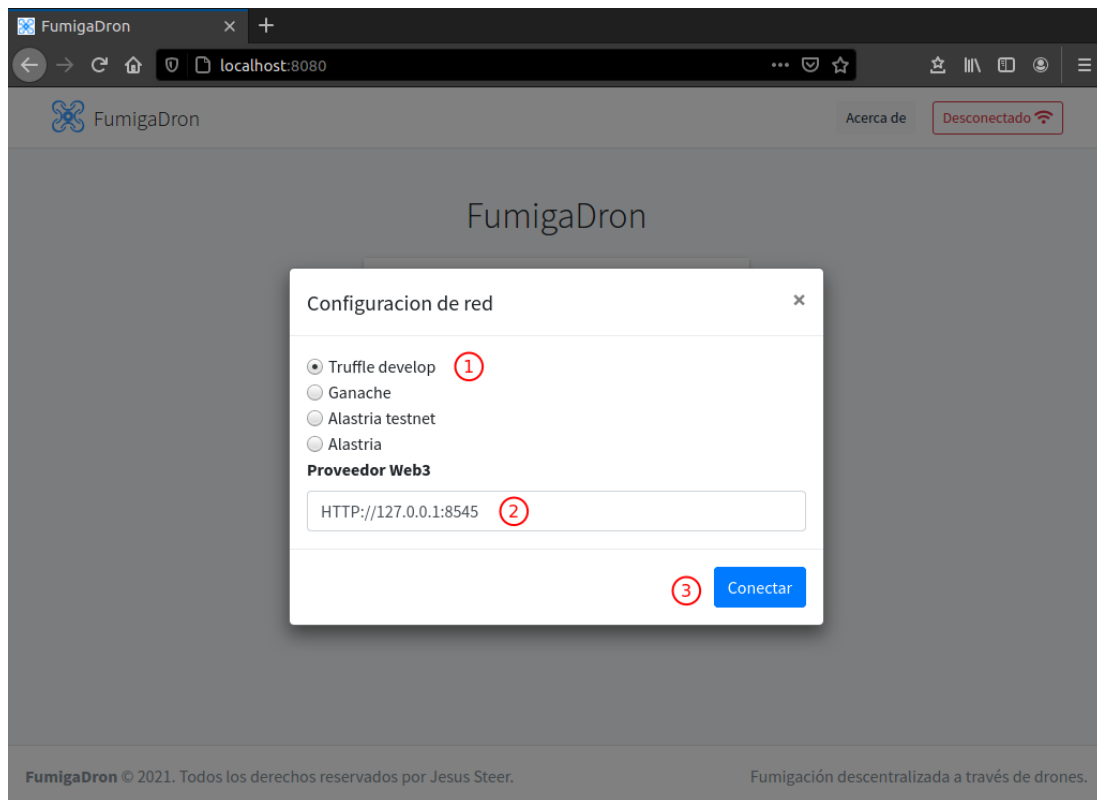
Manual de usuario

Este apartado muestra paso a paso la operación de la solución a través de las funciones de conexión a la red, la interfaz del empresario desde la cual se realiza la gestión de los drones, la interfaz del propietario desde la cual se gestionaran las parcelas, así como las operaciones de solicitud y realización de trabajos por parte de los usuarios.

Conectar

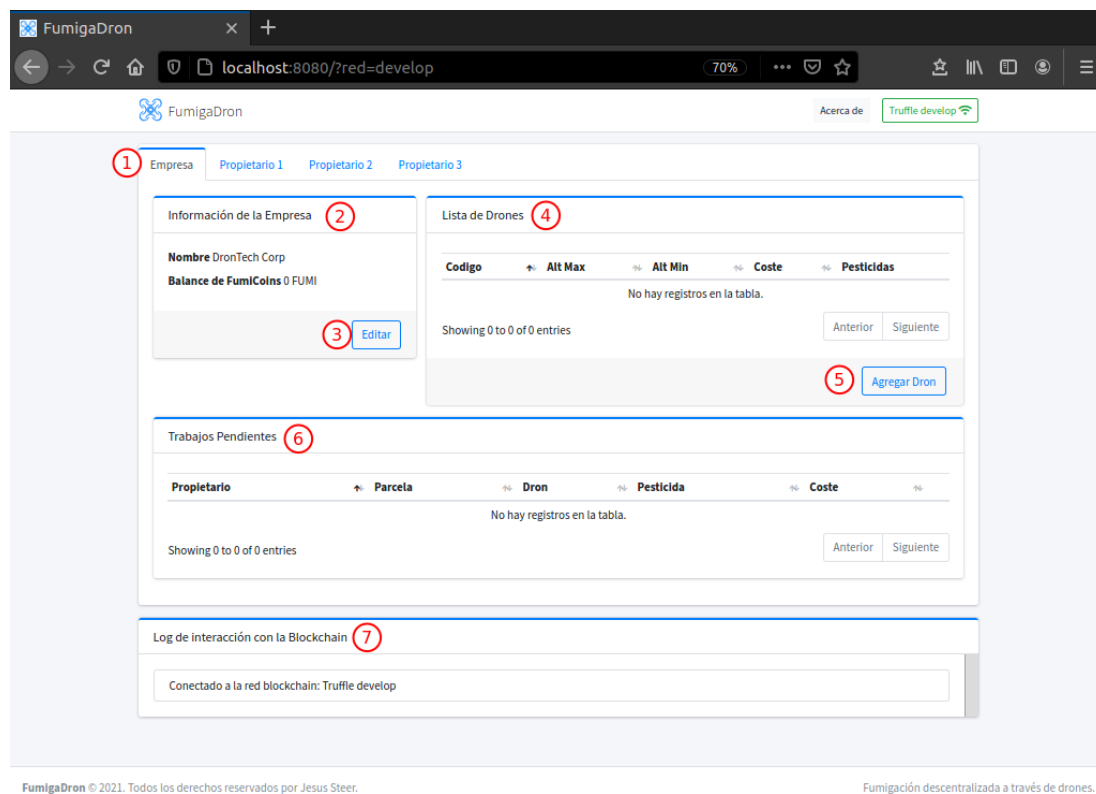


1. Botón de conectar, al hacer clic podemos elegir a que red blockchain queremos conectarnos.
2. Botón de estado de la conexión, el cual también nos permite cambiar de red en cualquier momento.
3. Botón con información sobre el proyecto.

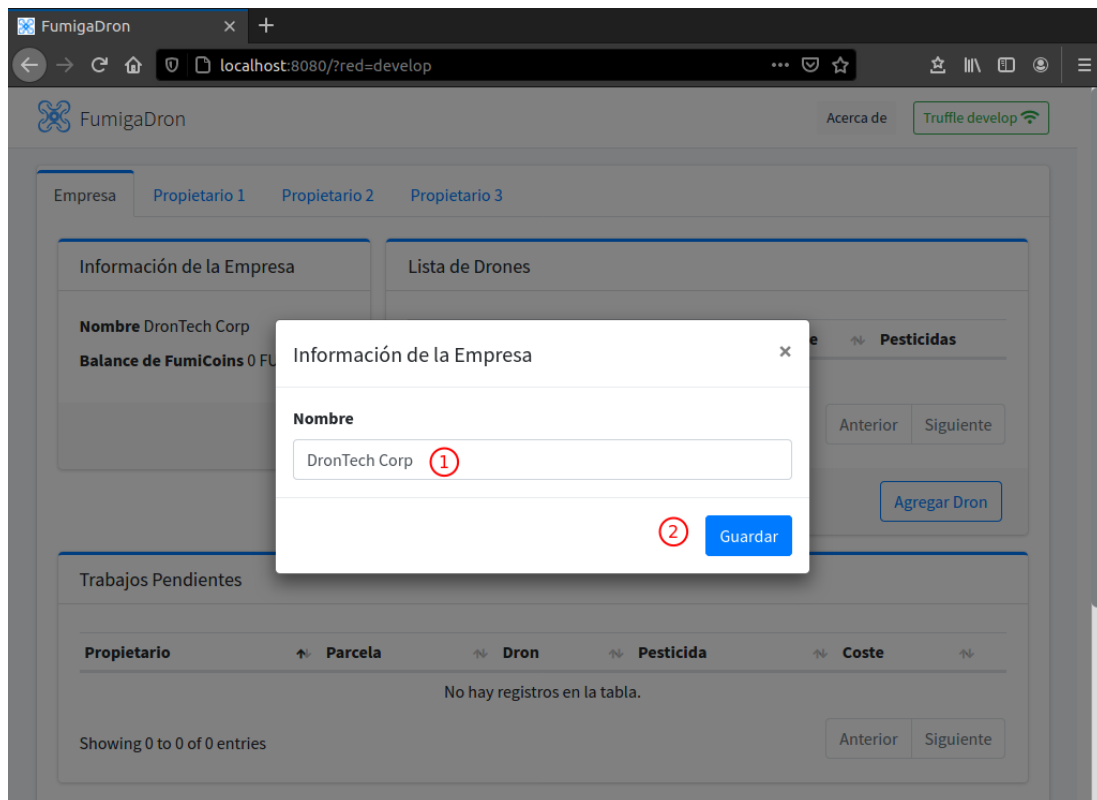


1. Al elegir cada uno de los elementos se cargara la configuración por defecto para conectar con cada una de estas redes en el campo de entrada Proveedor Web3.
2. Este campo permite editar la cadena de conexión en caso de que se quiera hacer un ajuste sobre la configuración por defecto.
3. Este botón intenta realizar la conexión con los datos suministrados.

Interfaz de la Empresa



1. Pestaña para cambiar entre los usuarios del sistema. Esto se incorpora para dar agilidad y facilitar la labor de prueba del sistema al no necesitar abrir varias pestañas o ventanas distintas del navegador. Cada pestaña aísla la cuenta y los datos del usuario en cuestión.
2. Tarjeta de información básica de la empresa.
3. Cambiar el nombre de la empresa.
4. Lista de tokens no fungibles drones creados.
5. Botón para agregar un nuevo dron.
6. Lista de trabajos pendientes.
7. Log de interacción con la blockchain, registra la información relacionada con las diferentes transacciones, y eventos que suceden al momento de interactuar con la red.



1. Formulario para cambiar el nombre de la empresa.
2. Botón para guardar los cambios.

The screenshot shows a web browser window with the URL `localhost:8080/?red=develop`. The application is titled 'FumigaDron'. A modal window titled 'Información del Dron' is open, displaying the following form fields:

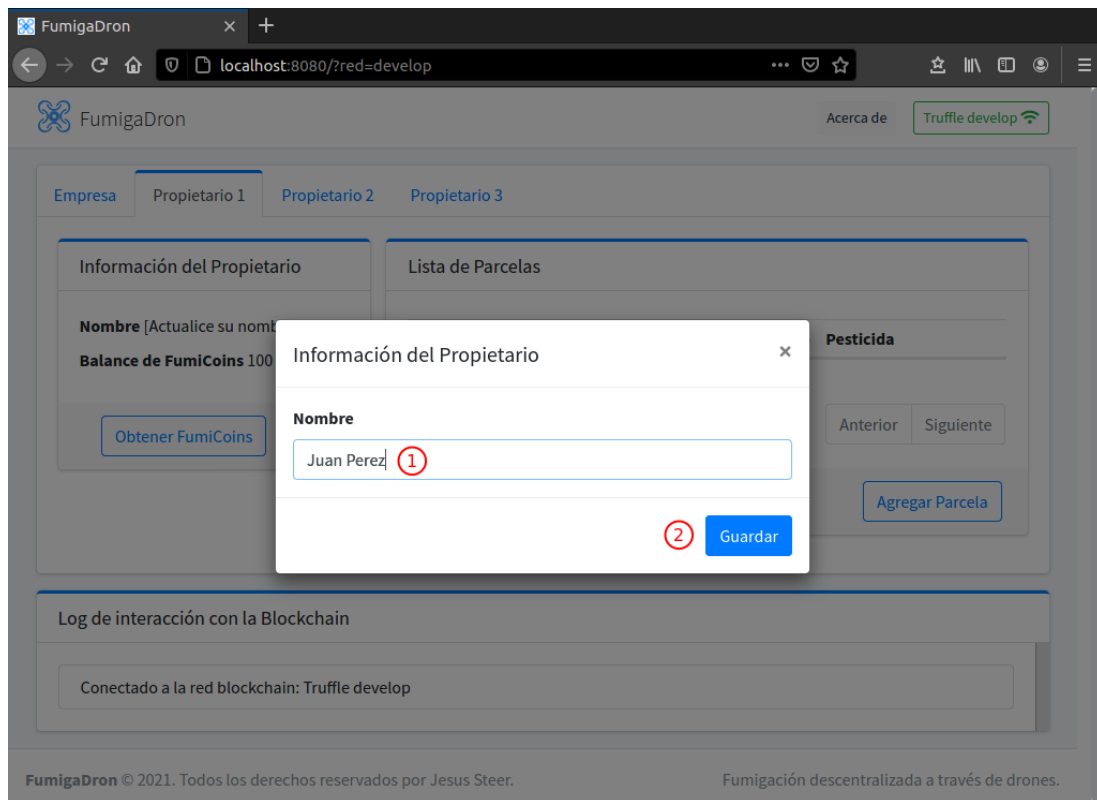
- Codigo:** A text input field containing 'DRX-001'.
- Altitud maxima:** A text input field.
- Altitud minima:** A text input field.
- Coste:** A text input field.
- Pesticidas aceptados:** A group of five checkboxes labeled A, B, C, D, and E.
- Guardar:** A blue button at the bottom right of the modal.

Red circles with numbers 1 through 6 are placed over the form elements to indicate specific points of interest for the user instructions.

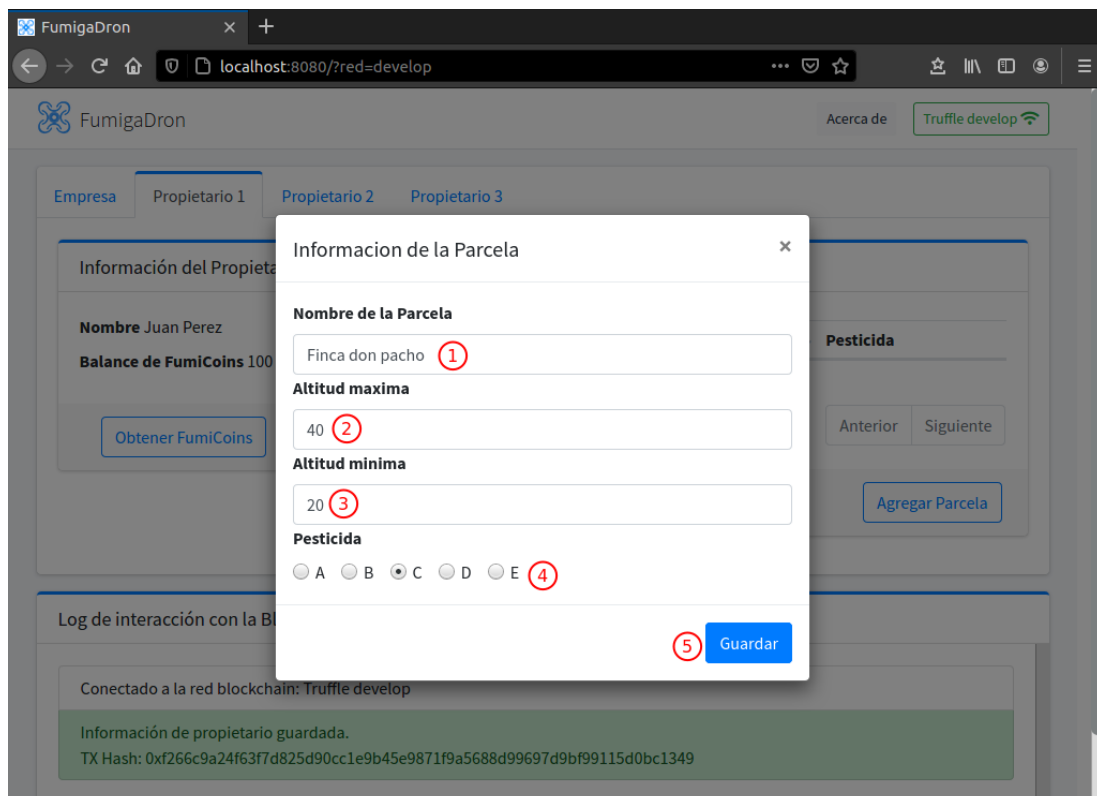
1. Código del dron, por defecto asigna un nombre pre construido con el id del dron.
2. Altura máxima, que debe ser mayor a la altura máxima de la parcela para ser compatible con ella.
3. Altura mínima, que debe ser menor a la altura mínima de la parcela para ser compatible con ella.
4. Coste expresado en unidades del token FUMI.
5. Cajas de chequeo para elegir los pesticidas compatibles.
6. Botón para ejecutar la operación guardar.

Propietario

1. Tarjeta de información básica del propietario.
2. Cambiar el nombre de la empresa.
3. Botón para obtener automáticamente 100 tokens FUMI.
4. Lista de tokens no fungibles de tipo parcela creados.
5. Botón para agregar una nueva parcela.



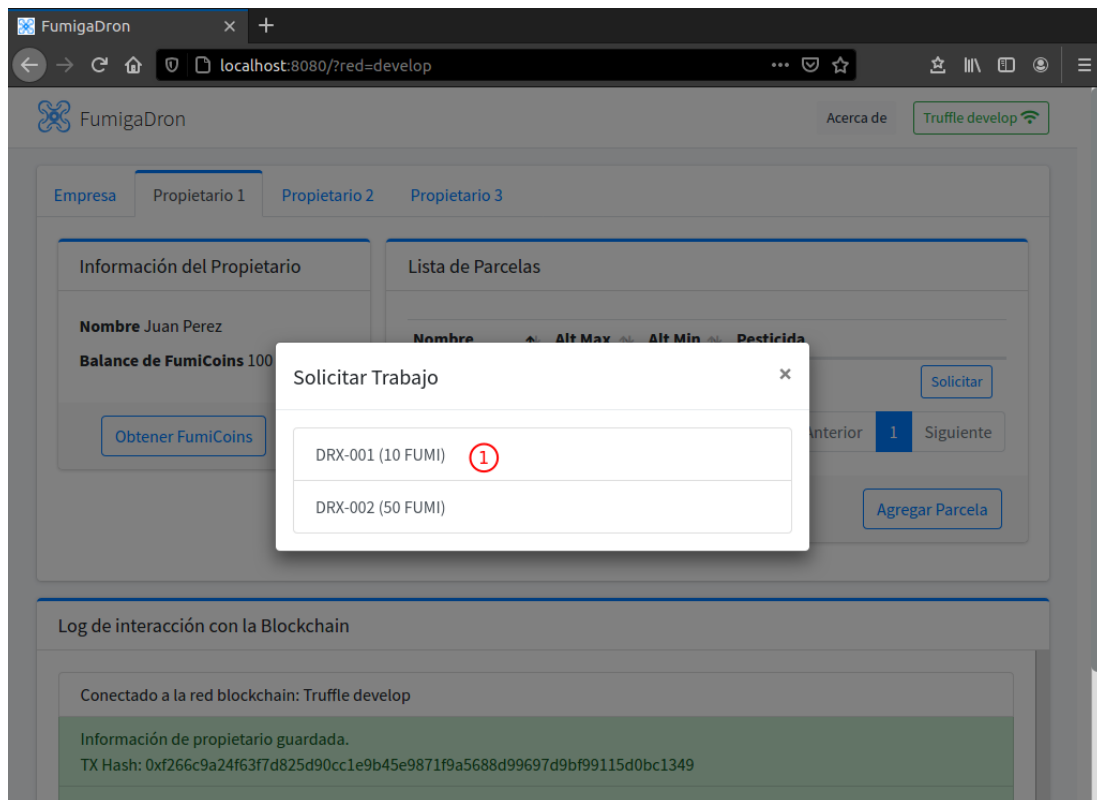
1. Formulario para cambiar el nombre del propietario.
2. Botón para guardar los cambios.



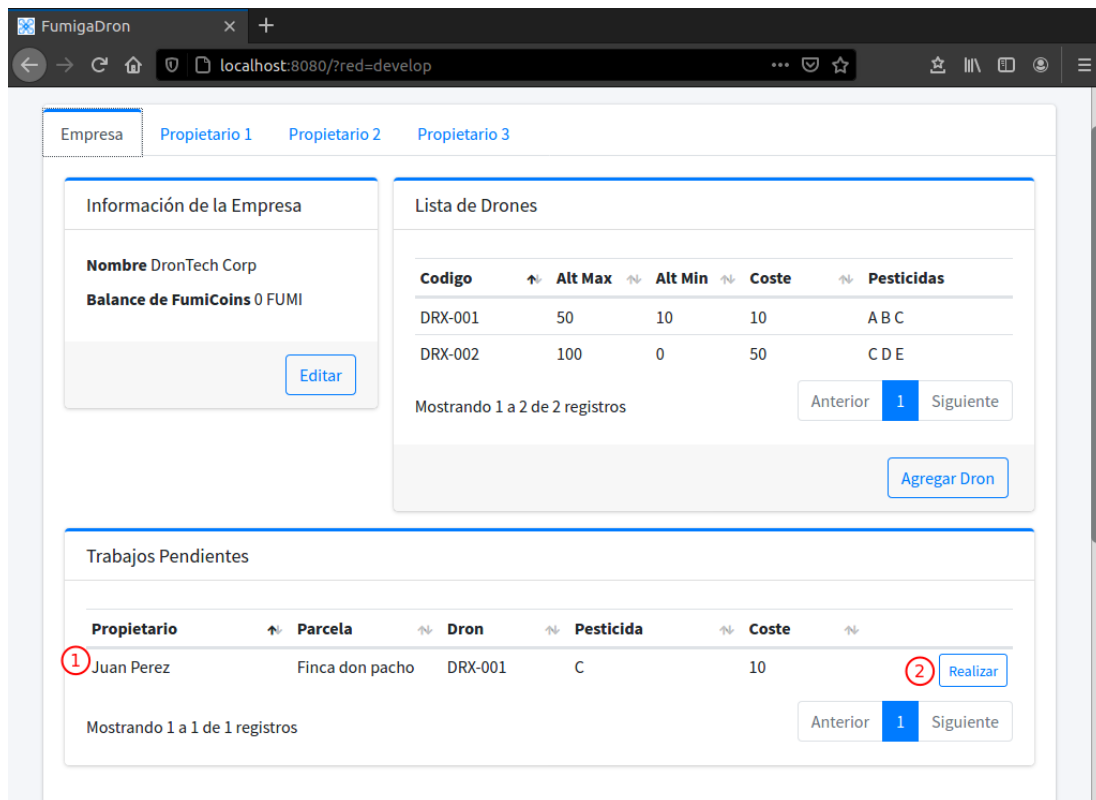
1. Nombre de la parcela.
2. Altura máxima, que debe ser menor a la altura máxima del dron para ser compatible con él.
3. Altura mínima, que debe ser mayor a la altura mínima del dron para ser compatible con él.
4. Cajas de selección única para elegir el pesticidas aceptado para esa parcela
5. Botón para ejecutar la operación guardar.

Realización de Trabajos de Fumigación

1. En la lista se muestra el registro de la parcela ingresada.
2. El botón de solicitar es utilizado para lanzar una solicitud de trabajo para la parcela. Solo será visible si no existe una solicitud activa para dicha parcela.



1. Se muestran los drones compatibles con la parcela. Se observa que por sus características es compatible con los dos drones creados, por lo tanto ambos se muestran como opciones. La cantidad de drones varía dependiendo de la compatibilidad con la parcela. Al hacer clic sobre el dron lanza la transacción de solicitud de trabajo, junto con el evento TrabajoSolicitado y la pre-aprobación de la transferencia.



1. En la interfaz del Empresario se puede ver el registro que aparece en la tabla de solicitud de trabajos, donde nos muestra el propietario, la parcela, el dron elegido y algunos datos adicionales de la operación.
2. Al presionar el botón realizar, se ejecuta la operación, lanzando el evento TrabajoRealizado, y realizando la transferencia pre-aprobada de tokens FUMI.

FumigaDron

Acerca de Truffle develop

Empresa Propietario 1 Propietario 2 Propietario 3

Información de la Empresa

Nombre DronTech Corp

Balance de FumiCoins 10 FUMI **1**

Editar

Lista de Drones

Código	Alt Max	Alt Min	Coste	Pesticidas
DRX-001	50	10	10	A B C
DRX-002	100	0	50	C D E

Mostrando 1 a 2 de 2 registros

Anterior 1 Siguiete

Agregar Dron

Trabajos Pendientes

Propietario	Parcela	Dron	Pesticida	Coste
No hay registros en la tabla.				

Showing 0 to 0 of 0 entries

Anterior Siguiete

1. Finalmente se puede confirmar la operación revisando como cambia el balance de los usuarios implicados en la operación.

Conclusiones

La primera conclusión obtenida del desarrollo de este trabajo es que a través de un ejemplo práctico se puede comprender mejor el papel de la tecnología blockchain en proveer herramientas y soluciones que permitan generar confianza entre individuos que no tienen una relación previa. Este fenómeno es muy interesante debido a que puede generar un impacto enorme en la sociedad, permitiendo a la humanidad avanzar de forma coordinada e inclusiva, requiriendo cada vez menos confianza en personas e instituciones, y de esta forma minimizando las injusticias o casos de corrupción que pueden generar algunos entornos.

La segunda conclusión a la que llego es que teniendo unas buenas bases conceptuales y las herramientas de las que disponemos hoy en día es posible hacer aplicaciones descentralizadas de calidad, seguras, estables y que generen un gran valor.

Finalmente concluyo que se hace necesario promover la tecnología blockchain, tanto en su uso y adopción, como facilitar la entrada de nuevos profesionales en la industria, ya que existe resistencia por el estado de la tecnología y las barreras que puede generar. Sin embargo, al realizar este trabajo puedo comprobar que si es posible aprender a hacer buenas aplicaciones descentralizadas, siempre que se tenga constancia y persistencia en este objetivo.