TRABAJO FINAL DE EXPERTO

Plataforma para la contratación de drones de fumigación

UNIR - Experto universitario en desarrollo de aplicaciones blockchain

Autor:

Marc Rodríguez Armengol

Justificación del uso de la tecnología blockchain para resolver el problema propuesto

Los pesticidas son sustancias altamente contaminantes y de uso regulado, es necesario asegurar que se aplican únicamente sobre la parcela correspondiente a la altura indicada por lo que las características inherentes a la tecnología blockchain de seguridad, automatización y trazabilidad garantizan la fiabilidad de todo el proceso.  Además, se evita que los datos puedan ser alterados una vez registrados los drones y las parcelas.

La unidad de pago utilizada es a través de tokens por lo que la gestión del pago no requiere de la confianza entre las partes, el dron ejecuta su trabajo una vez ha recibido la aprobación del presupuesto por parte del propietario de las parcelas y se transfieren los tokens a la empresa cuando el dron ha ejecutado el trabajo encomendado. Al ser el proceso comercial totalmente automatizado se evita el riesgo de estafa por alguna de las partes y al estar su ejecución también controlada por las funciones de los smart contracts se elimina el riesgo de manipulación en la correcta ejecución del servicio contratado.

La trazabilidad e identificación de todas las partes involucradas también elimina cualquier divergencia en la responsabilidad de un mal uso de los drones o de cualquier disputa sobre la carga de los drones permitiendo determinar con rapidez y exactitud la causa del problema.

Es por todo ello que se justifica el uso de la tecnología blockchain sobre cualquier otra tecnología actualmente existente.

Análisis y modelo del sistema propuesto

´

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.

Para la gestión de pagos se debe crear un token propio basado en el estándar ERC 20.

La empresa solicita tener una interfaz web que le permita registrar los drones y asignarles trabajos. A su vez, también se debe proporcionar una interfaz web para que los propietarios de las parcelas las puedan registrar y tengan la posibilidad de contratar un dron con las características que requiere su parcela.

De lo que aquí se pide podemos concluir:

Debemos crear un token ERC20 y un contrato para la gestión de los drones y parcelas sobre alastria mediante solidity.

Debemos crear una interfaz web para interactuar con los contratos, para la web en sí utilizamos HTML y CSS y para lógica de las funciones utilizamos javascript, en concreto la librería web3.js.

El contrato gestión permite:

* Registrar los drones y las parcelas
* Solicitar una fumigación por parte del cliente aprobando un presupuesto determinado
* Confirmación de la fumigación y cobro del servicio

El contrato TokenDron es un token compatible con ERC20 que tendrá la utilidad de pago por el servicio. Este token es una ampliación del original de OpenZeppelin. Utiliza la librería SafeMath para evitar el overflow de números enteros.

Para realizar estas acciones se ha tomado la decisión de estructurar la aplicación de la siguiente manera:

Para tratar los pesticidas, se ha decidido tratarlos como uint256 para asignarles una id a cada uno y a la vez se ha creado la función nuevoPesticida para añadirlos. Esta función realiza 2 mapping para guardar el pesticida y poder buscar su nombre por la id y la id por su nombre con las funciones correspondientes.

En el constructor de la aplicación se crean los 5 pesticidas solicitados llamando a nuevoPesticida con el nombre de cada pesticida como parámetro.

El cliente/empresa registra una parcela/dron con los parámetros de su elección, esta aplicación comprueba que el pesticida seleccionado esté aceptado y si es así guarda un mapping de la id de la parcela/dron a un struct con la información introducida.

Una vez registrada la parcela, el cliente necesita tokens para pagar por el servicio, para este caso se ha añadido una función obtenerBono para recibir 1M de tokens. (Cualquiera pueda solicitar el bono).

La secuencia de la interacción consiste en primero el registro de los drones y parcelas y luego el cliente solicita un trabajo y la empresa lo realiza y cobra.

Cuando el cliente solicita el trabajo lo que está haciendo es aprobar una transferencia de TokenDron a la cuenta de la empresa, para saber la cantidad que hay que aprobar, esta función multiplica el tamaño de la parcela por el precio/m2 de la misma que es un valor máximo que el cliente está dispuesto a pagar por m2.

Esta función aprueba en el contrato TokenDron la transferencia de X tokens de la cuenta del cliente a la cuenta de la empresa mediante la llamada a increaseAllowanceOrigin, (después de hacer pruebas se ha llegado a la conclusión que para que la cuenta de la empresa sea la beneficiaria de los tokens y no la dirección del propio contrato es necesario añadir 2 funciones al contrato TokenDron que son similares a transferFrom y increaseAllowance del estándar de OpenZeppelin pero cambiando el msg.sender por tx.origin). Después emite un evento TrabajoSolicitado con la id de la parcela.

La empresa puede ver el evento y buscar la parcela con la id y obtendrá la información que necesita. Después elige un dron compatible, o registra uno nuevo que se adapte a la parcela en cuestión.

Para la fumigación, la empresa introduce la id del dron (previamente registrado por la empresa) y la id de la parcela a fumigar, la función comprueba la compatibilidad de las alturas entre el dron y la parcela seleccionados mediante “require” lanzando sus correspondientes mensajes de error. Después multiplica los m2 de la parcela por el precio por m2 del dron para obtener el valor de la factura y llama a transferFromOrigin para transferir el pago de la cuenta del cliente a la de la empresa, en caso que el pago no se realice lanza un error. Si todas las condiciones son correctas se realiza una transferencia de tokens del cliente a la empresa y se lanza un evento ParcelaFumigada con la id de la parcela, la id del dron y el valor de la factura.

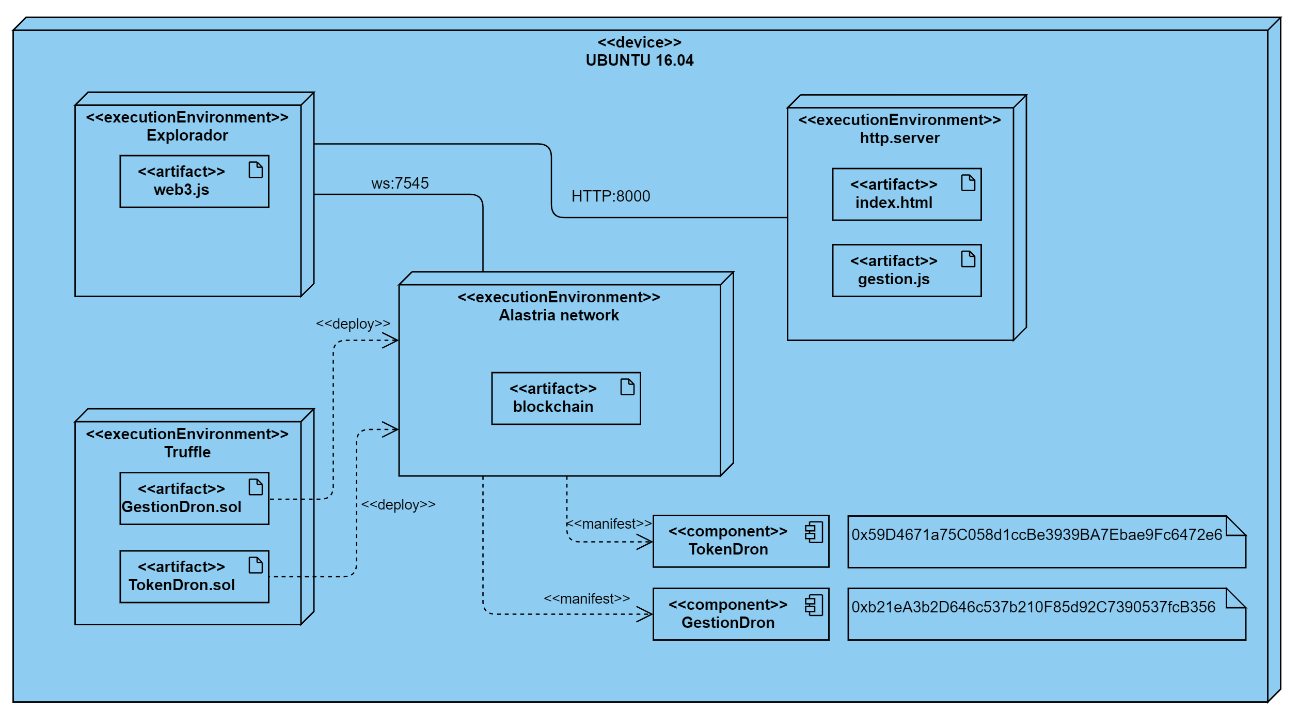
En caso que un cliente haya registrado una parcela y quiera cambiar algún parámetro puede modificarlos con la función modificarParcela introduciendo la id de la parcela a modificar y los nuevos parámetros.

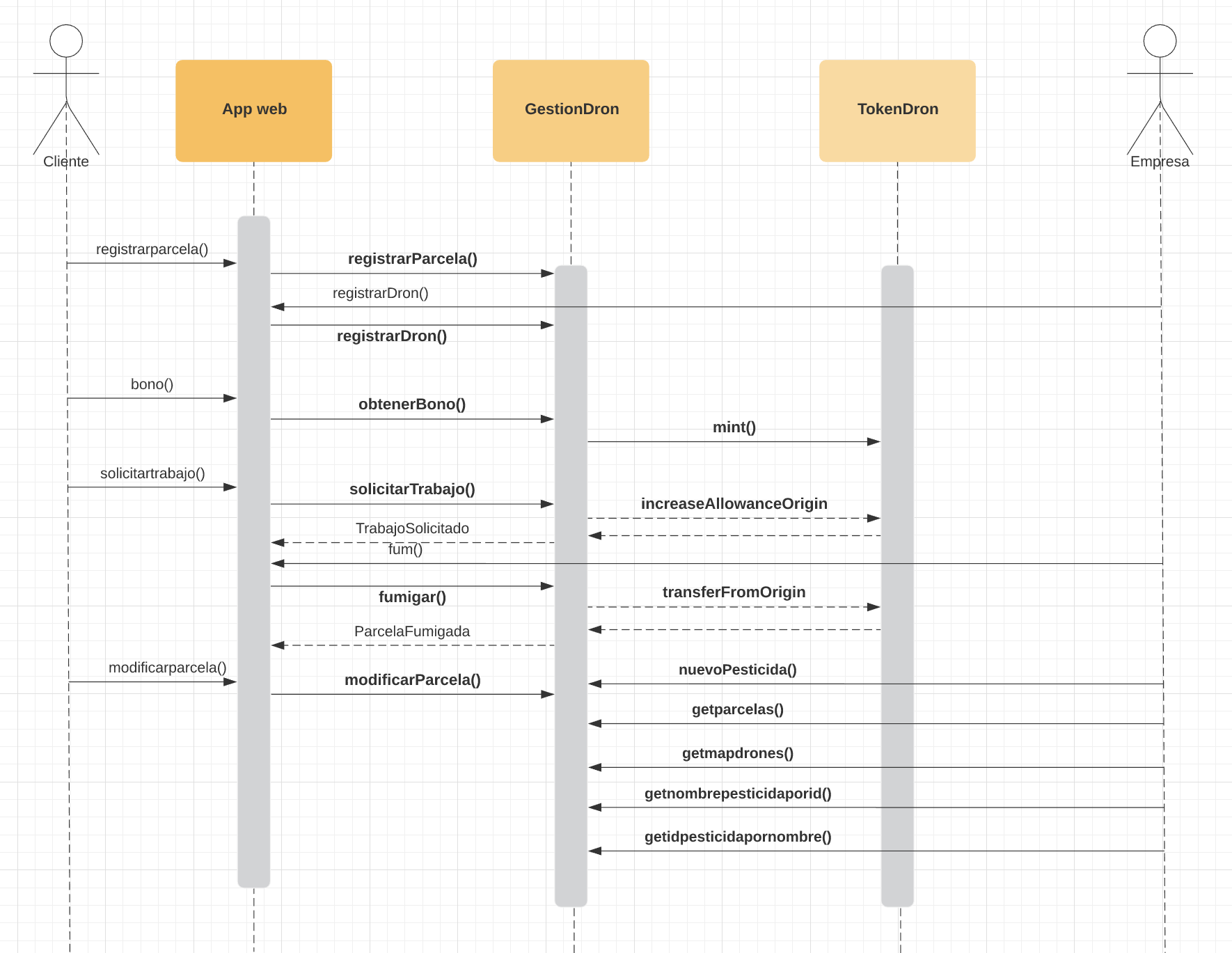
Descripción del entorno de desarrollo utilizado

Para realizar el desarrollo de esta aplicación se ha empleado el lenguaje de programación Solidity para los smartcontracts y HTML y javascript para la interfaz web.

Para hacer pruebas y comprobar las funcionalidades se ha utilizado el IDE de remix.ethereum.org.

Para desplegar el contrato en la red se ha hecho uso de truffle, que contiene las herramientas necesarias para compilar, desplegar y hacer el testing de la solución.

El testing se ha hecho utilizando truffle y mocha.



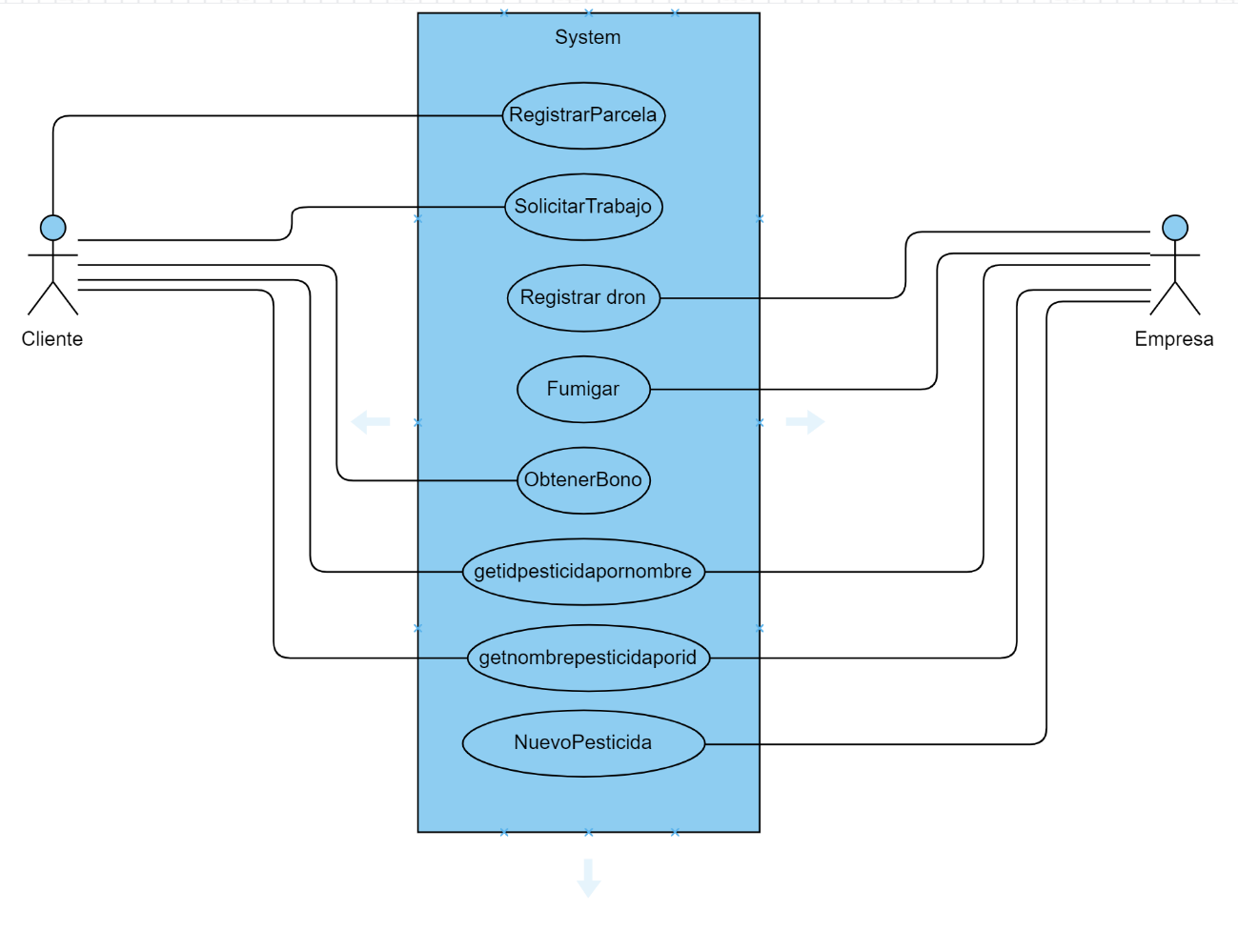
Diagrama de despliegue

Diagrama de casos de uso

Instrucciones de despliegue

Para realizar el despliegue de esta aplicación se siguen estos pasos:

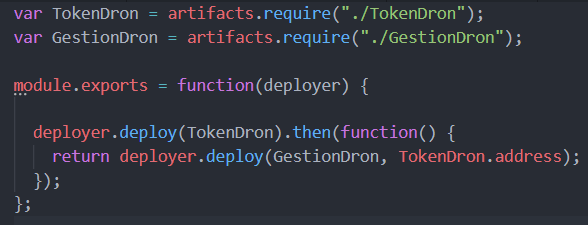
1. Instalar truffle

npm install -g truffle

1. Crear una carpeta nueva y desde ella ejecutar truffle init

truffle init

1. Añadir a esa carpeta los contratos, el index.html y la lógica javascript
2. Crear un archivo 2\_deploy.js en la carpeta migrations para desplegar los contratos (Este código despliega el contrato TokenDron y recibe la dirección para pasarla como constructor al desplegar el contrato GestionDron)



1. Configurar el archivo truffle-config.js con los parámetros de nuestro proyecto (red, compilador)
2. Ejecutar desde la carpeta truffle migrate (compila los contratos y los despliega en la red seleccionada del archivo truffle-config

truffle init

1. Iniciar un servidor web desde la carpeta donde se aloja el archivo index.html

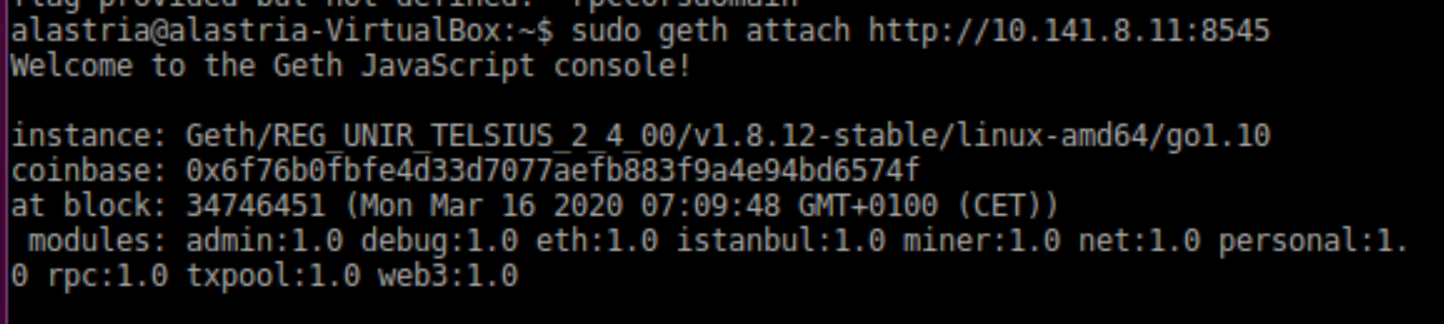
python -m http.server 8000

1. Interactuar con los contratos desde el navegador (accedemos a la web desde la dirección IP definida)

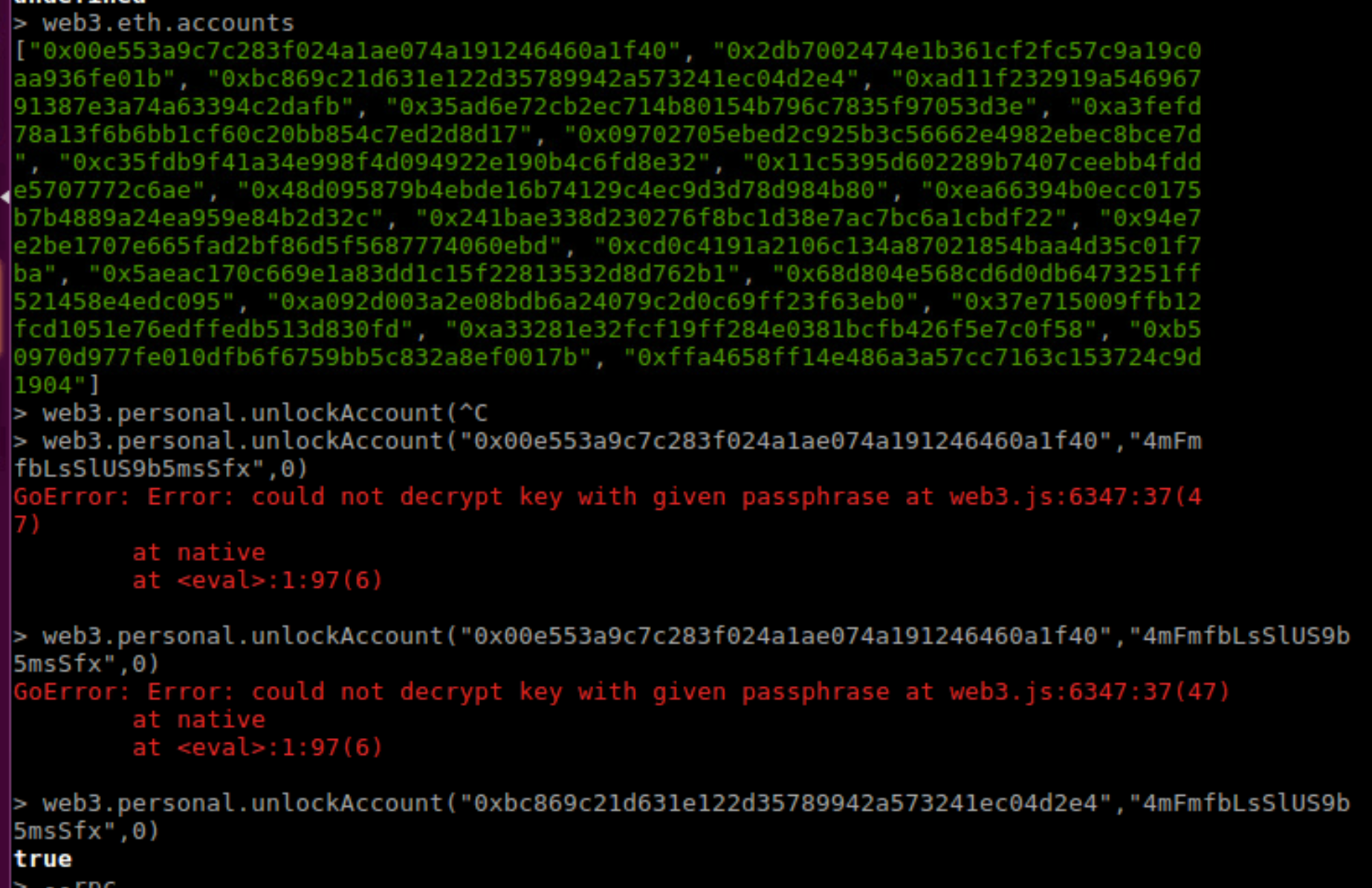
**Manual de usuario**

Para utilizar esta aplicación primero hay que desplegar los smartcontracts en la red TELSIUS de Alastria.

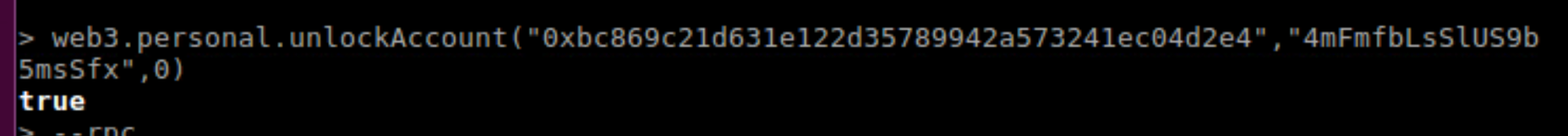
Para desplegar en la red primero debemos conectarla con geth con el siguiente comando:

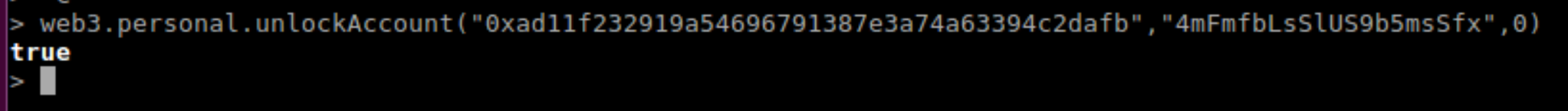


Una vez tenemos geth conectado, podemos utilizar sus funcionalidades, como por ejemplo mostrar la cuentas y desbloquarlas para su uso.

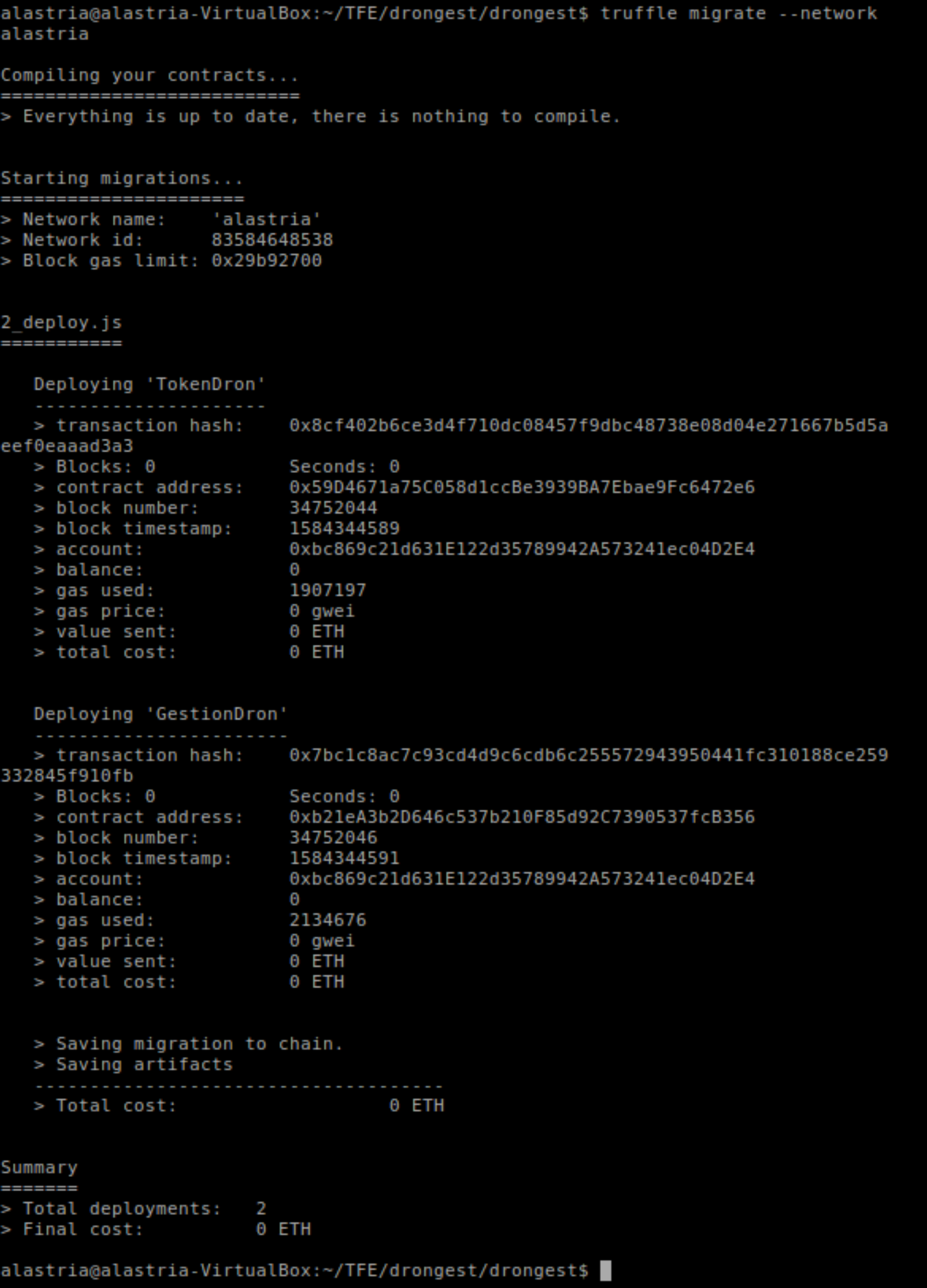


Geth utiliza la primera cuenta por defecto [0] pero en este caso vemos que no disponemos de la contraseña, si es así con las cuentas 2 y 3.

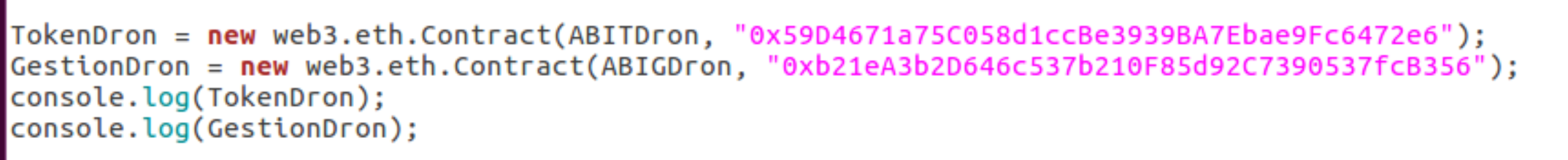




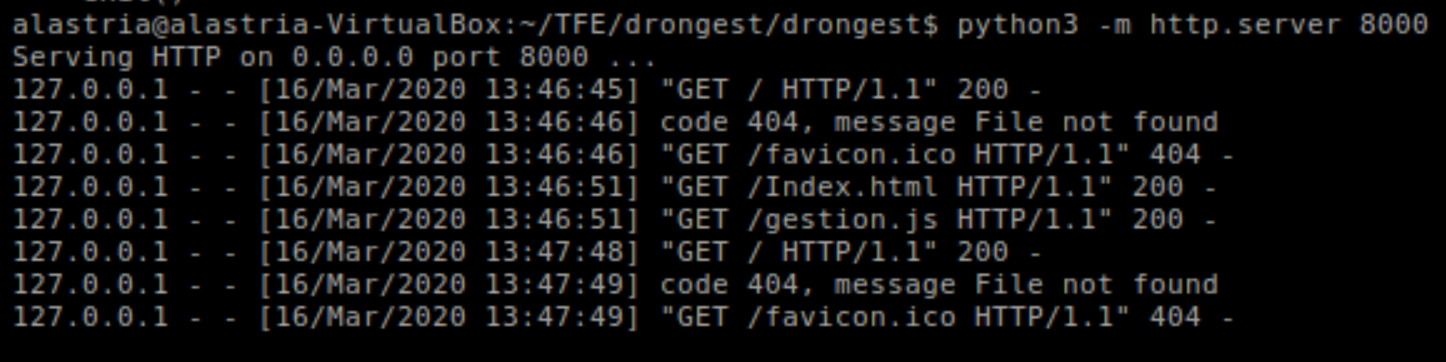
En esta captura apreciamos el despliegue de los contratos en la red de alastria. Se especifica en el archivo truffle-config.js. Puede haber otras redes así que tenemos que elegir la que nosotros queramos a la hora de desplegar con el comando --network.

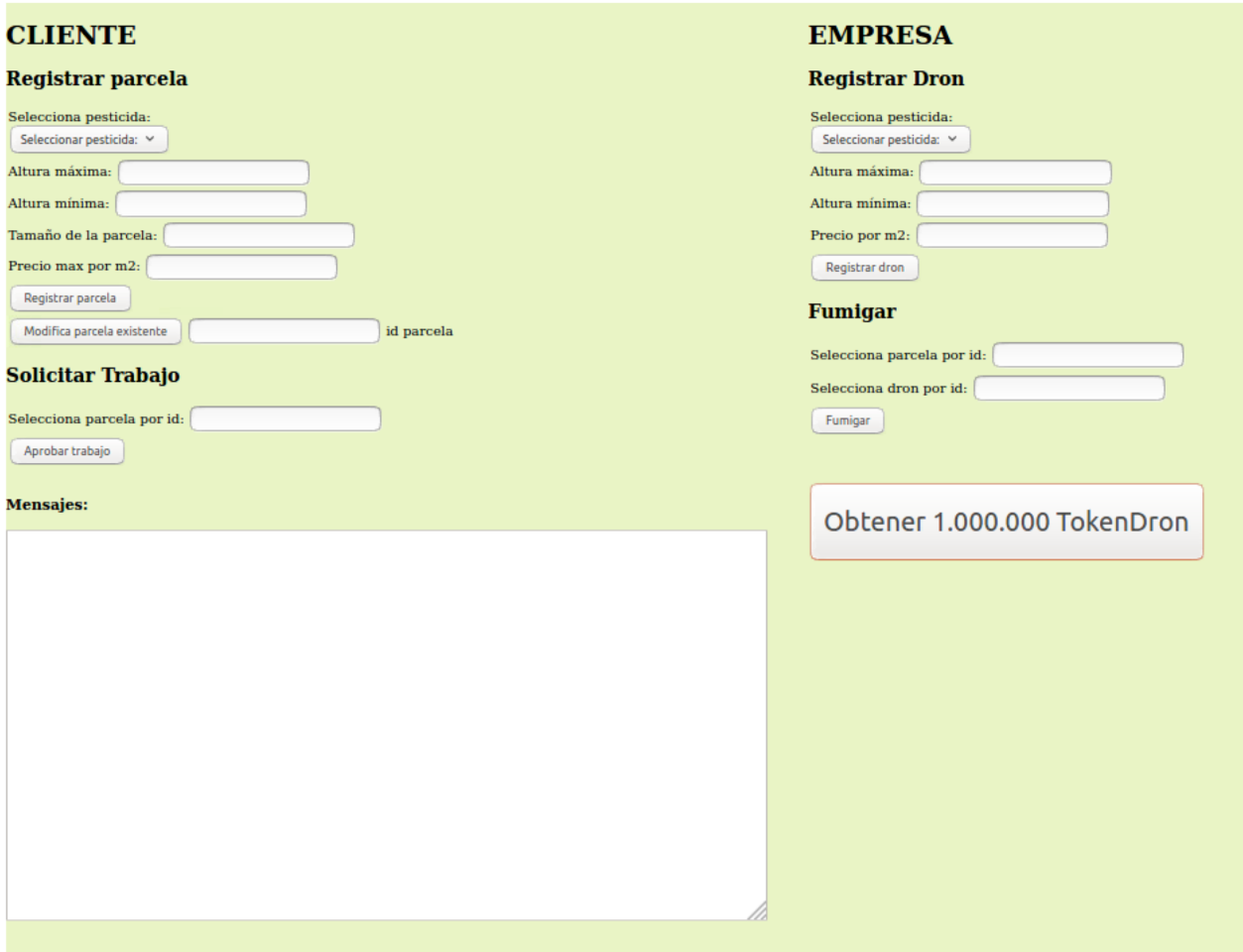


Una vez tenemos los smartcontracts desplegados, copiamos las direcciones de esos contratos y las introducimos en sus respectivas variables en el archivo gestión.js que se encarga de almacenar la lógica javascript de nuestra interfaz web. Esto crea una instancia de nuestro contrato para poder interactuar con ella con el resto de funciones javascript que llaman a los métodos de los contratos.



El siguiente paso es, desde la carpeta donde se aloja nuestro index.html, lanzar un servidor web de manera que podamos acceder a la interfaz desde un navegador web. Lo podemos hacer sencillamente con el siguiente comando de Python especificando el puerto en el que vamos a conectarnos, en este caso el 8000.



La dirección será localhost:8000. Accedemos a ella con el navegador y podemos observar el archivo html funcionando:

Se divide la web en 4 apartados, uno para que el cliente registre la parcela y solicite la fumigación, uno para la empresa para registrar drones y fumigar y los otros 2 son para mostrar los mensajes devueltos de la blockchain y para obtener tokens gratis para probar las funciones de la plataforma.

Para registra un dron o una parcela, el usuario debe introducir los valores correspondientes en cada casilla y, al hacer click en el botón se envía una transacción a la blockchain con los parámetros introducidos en las cajas de texto.

El cliente puede solicitar un trabajo introduciendo la id de su parcela.

La empresa puede fumigar introduciendo el id de la parcela a fumigar y el id del dron que cumpla con esas características concretas de la parcela.

**Conclusiones**

La tecnología blockchain ofrece un gran abanico de posibilidades para crear aplicaciones y automatizar sistemas de bancarios, de identidad, registro y auditoría entre otros. Aún nos encontramos en las fases iniciales de lo que será una revolución mundial y esta tecnología refleja esta fase temprana, con diversidad de versiones incompatibles entre sí, fallos de seguridad en el código y otros tantos obstáculos que se interponen en el camino de una sociedad digital. Cada vez más rápido vemos proyectos que utilizan esta tecnología para resolver problemas de la vida cuotidiana y hacer la vida más fácil que es al final en lo que consiste el desarrollo. Todos estos problemas se irán resolviendo gracias al trabajo y la cooperación descentralizada que origina internet.

Una de las conclusiones que hemos sacado al realizar este curso es que la descentralización de sistemas puede originar caos en un principio y teniendo en cuenta que como sociedad nos dirigimos cada vez más al mundo digital es importante tomar responsabilidad y hacer que esta transición sea lo mas suave posible ayudando a diseñar sistemas de información óptimos. Es muy importante optimizar como lo hace la naturaleza y en redes descentralizadas aún más pues el coste se multiplica.

En un futuro la gente va a dejar de emplear tiempo y recursos en contabilidad y auditoría para dedicarse a otras cosas más importantes ya que los ordenadores pueden hacer el laborioso trabajo que antes hacíamos nosotros, pero es muy importante no sólo automatizar procesos sino hacerlo de manera correcta y segura.

Otra conclusión que hemos sacado es que las redes permisionadas como Alastria suponen un cambio radical en comparación a las no permisionadas como Ethereum de cara al usuario medio que no entiende de blockchain y no está dispuesto a pagar gas para ejecutar transacciones, pero aún falta más colaboración entre los integrantes del consorcio para crear soluciones basadas en esta red. Gracias a este curso y otros que se están dando en paralelo vamos a ver un crecimiento exponencial de aplicaciones creadas en Alastria en los próximos años.