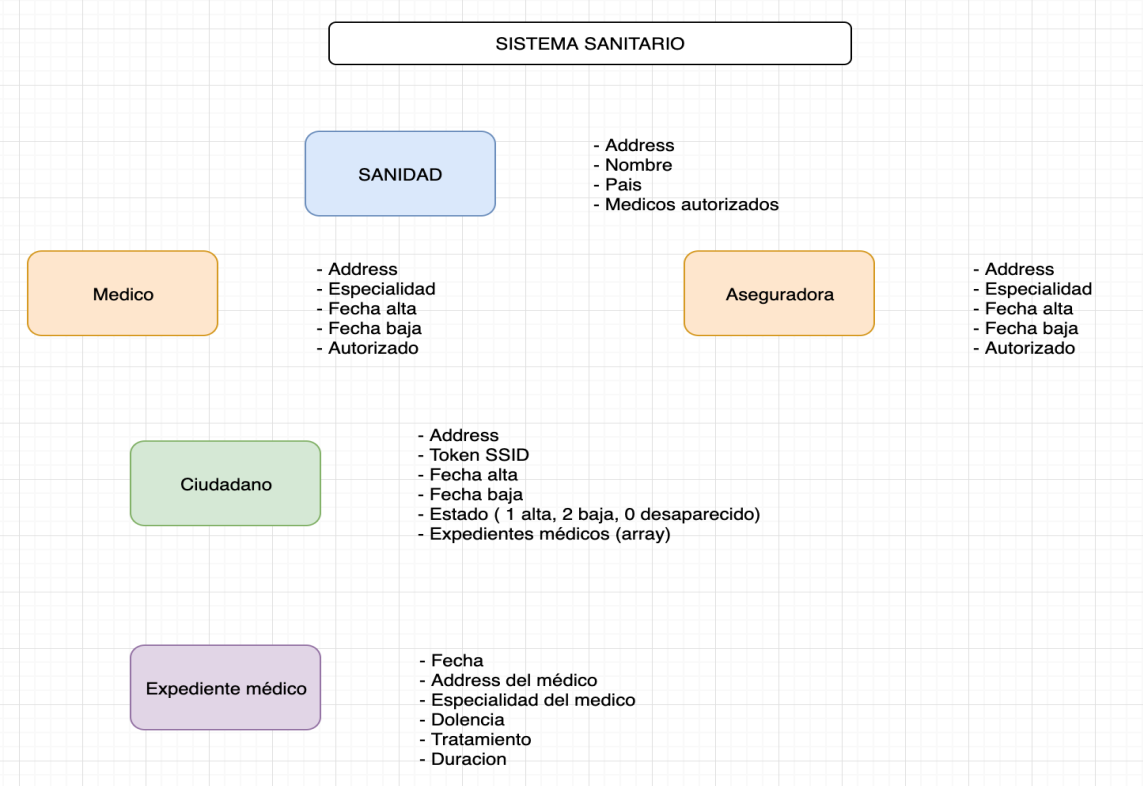
Historia clínica unificada

Introducción

Para dar solución al problema planteado se ha pensado en implementar el siguiente sistema, le llamaremos : Sistema Sanitario

Constará de los siguientes actores:



donde:

- Sanidad será el organismo oficial de cada país quien gestionara el sistema
- Médicos, se darán de alta al sistema Sanidad y esta les podrá autorizar para usar el sistema o no
- Ciudadanos, son los usuarios del sistema sanitario, serán dados de alta por Sanidad, con la opción de diferentes estados y con la asignación de un Token (SSID)
- Expediente, son los datos de cada tratamiento que recibe un ciudadano, los podrán introducir los médicos autorizados y podrán ser consultados por los médicos autorizados
- Aseguradora, si están autorizados por Sanidad, podrán consultar los expedientes de los ciudadanos, cada consulta tendrá un coste

Funcionalidades implementadas

El proyecto esta disponible en github:
https://github.com/sjms00/SSID

A parte de las funcionalidades heredadas del paquete instalado zeppelin-solidity, las funciones del proyecto se han distribuido en dos ficheros .sol.

En el fichero SistemaSanitario.sol se han implementado la funciones:

- altaMedico, los médicos se darán de alta indicando su especialidad
- consultaMedico, solo Sanidad la puede utilizar donde obtendrá la información del médico consultado.
- bajaMedico, solo Sanidad la puede utilizar para dar de baja a un médico
- autorizarMedico, solo Sanidad la puede utilizar para autorizar a un médico a usar el sistema
- desautorizarMedico, solo Sanidad la puede utilizar para desautorizar a un médico a usar el sistema
- altaCiudadano, solo Sanidad la puede utilizar para dar de alta (estado = 1) un ciudadano, asignándole un token SSID (ERC721)
- bajaCiudadano, solo Sanidad la puede utilizar para dar de baja (estado = 2) un ciudadano
- modifCiudadano, solo Sanidad la puede utilizar para modificar el de un ciudadano, por si se había dada de baja por error, o (estado = 0) si está desaparecido
- altaAseguradora, Sanidad da de alta la aseguradora y la autoriza a usar el sistema, solo consulta de expedientes
- insertExpediente, solo un médico autorizado por Sanidad puede insertar expedientes de un ciudadano, indicando dolencia, tratamiento y duración, no se permite que un médico se autodiagnostique. Se emite un evento con información del médico, paciente y expediente.
- consultaExpedienteAseguradora, función solo permitida a las aseguradoras autorizados, permite consultar un expediente en concreto de un ciudadano. . Se emite un evento con información del médico, paciente y expediente.
- consultaExpediente, función solo permitida a los médicos autorizados, permite consultar un expediente en concreto de un ciudadano. . Se emite un evento con información de la aseguradora, paciente y expediente.

En el fichero TokenSSID.sol se han implementado la funciones:

- mintTo, assigna un nuevo token a la address que se le pase.
- _getNextTokenId, devuelve el siguiente TokenId para assignar

Configuración del entorno

Se ha decidido utilizar el Framework Truffle para configurar el entorno, este nos aporta funcionalidades que nos ayuda al desarrollo com es la resolución de las dependencias de los diferentes contratos utilizados, así como la preparación del entorno para posibles migraciones de los contratos.
El detalle para su implementación y configuración es:

>yarn add global truffle
>yarn init
>truffle init
>yarn add zeppelin-solidity (par añadir las funcionalidades de un token ERC721)
>yarn add global ganache-cli
modificamos truffle-config.js para utilizarlo
lo dejamos arrancado en local ganache-cli -p 7545

```
networks: {
  // Useful for testing. The `development` name is special - truffle uses it by default
  // if it's defined here and no other network is specified at the command line.
  // You should run a client (like ganache-cli, geth or parity) in a separate terminal
  // tab if you use this network and you must also set the `host`, `port` and `network_id`
  // options below to some value.
  //
   development: {
    host: "127.0.0.1",     // Localhost (default: none)
    port: 7545,            // Standard Ethereum port (default: none)
    network_id: "*",       // Any network (default: none)
   },
```

Para poder hacer el deploy en la red de test " rinkeby ", instalamos el soporte a la wallet:
>npm install truffle-hdwallet-provider
modificamos truffle-config.js para utilizarlo, las 12 palabras de mnemonic están en una variable de entorno "MNEMONIC" del ordenador

```
 * You'll also need a mnemonic - the twelve word phrase the wallet uses to generate
 * public/private key pairs. If you're publishing your code to GitHub make sure you load this
 * phrase from a file you've .gitignored so it doesn't accidentally become public.
 *
 */
const fs = require('fs');
const HDWalletProvider = require('truffle-hdwallet-provider');
const mnemonic = process.env.MNEMONIC;
```

```
 rinkeby: {
   provider: () => new HDWalletProvider(mnemonic, `https://rinkeby.infura.io/v3/7046305a9eb0
   network_id: 4,       // Ropsten's id
   gas: 5500000,        // Ropsten has a lower block limit than mainnet
   confirmations: 2,    // # of confs to wait between deployments. (default: 0)
   timeoutBlocks: 200,  // # of blocks before a deployment times out  (minimum/default: 50)
   skipDryRun: true     // Skip dry run before migrations? (default: false for public nets )
 }
```

Tests implementados

Una vez desplegado el contrato se realizan los siguientes tests con esta relación de Accounts:

// acounts[0] sera Sanidad

// acounts[1] sera medico 1 autorizado

// acounts[2] sera medico 2 NO autorizado

// acounts[3] sera ciudadano 1

// acounts[4] sera ciudadano 2
// acounts[6] sera Aseguradora 1

1- Comprobar que existe la dirección del contrato desplegado
2- Comprobar que la dirección que lo ha desplegado es la de Sanidad
3- Dar de alta una aseguradora
4- Dar de alta médico 1
5- Dar de alta médico 2
6- Consulta médico 1
7- Autorizar médico 1
8- Autorizar médico 2 por parte de médico 1, debe de dar KO
9- Dar de alta ciudadano 1
10- dar de alta ciudadano 2 por parte de médico 1, debe de dar KO
11- Insertar expediente a ciudadano 1 por parte de médico 1 (autorizado)
12- Insertar expediente a ciudadano 1 por parte de médico 2 (No autorizado), debe de dar KO
13- 14 – Insertar un expediente a un ciudadano que es médico por parte del mismo médico, debe de dar KO
15- Consulta de un expediente por un médico autorizado
16- Consulta de un expediente por un No médico autorizado, debe de dar KO
17- Consulta de un expediente por parte de una aseguradora con saldo suficiente
18- Consulta de un expediente por parte de una aseguradora sin saldo suficiente, debe de dar KO

ejecutamos :
>truffle test

para realizar los tests

Deploy del proyecto

Primero revisamos si compila y verificamos que no hay errores:

>truffle compile

Para deployar el proyecto sobre la red local de desarrollo:

>truffle migrate

Para el despliege sobre la red Rinkeby

>truflle migrate –network rinkeby

En el fichero deploy_rinkeby.txt esta la salida del deploy

Para reiniciar el proyecto o deploy inicial:
>truffle migrate –reset
>truffle migrate -network rinkeby –reset