dit-upm

proyecto 1

smart contracts
solidity

José A. Mañas < http://www.dit.upm.es/~pepe/>
Dep. de Ingeniería de Sistemas Telemáticos
E.T.S. Ingenieros de Telecomunicación
Universidad Politécnica de Madrid

19.9.2018

objetivos

- desarrollar un smart contract
  - codificar en solidity
  - probar
  - desplegar en la red de pruebas

enunciado

- Desarrollar un sistema de cursos
  - 1 contract en solidity
  - el profesor
    - pone el nombre del curso
    - puede poner otros subcursos
    - admite a los alumnos
    - evalúa a los alumnos (nota)
  - el alumno
    - se registra
    - puede ver su evaluación (nota)

estructura del contrato

- estado interno; es opinable

```
pragma solidity ^0.4.0;

contract Course {
    address owner;
    string name;
    Course[] previous;

    enum State {Registered, Accepted, Evaluated}
    struct Student {
        State state;
        uint score;
    }
    mapping(address => Student) students;
```

ciclo de vida: comienzo

- el profesor usa un constructor
  - marca el nombre del curso
  - y fija al profesor como owner
  - constructor(string _name) public { }
- tenemos un getter público
  - function getName() public returns(string)
- opcionalmente el profesor puede asociar otros cursos componentes (contratos desplegados, identificados por su dirección
  - function addCourse(address _addr) public { }
  - esto solo puede hacerlo el profesor

ciclo de vida /registro

- el alumno puede pedir registrarse en el curso
  - function register() public { }
  - si hay subcursos, se le registra en los subcursos
  - el alumno pasa a estado State.Registered
  - si ya está en el curso, no hace nada
- el profesor puede aceptar un alumno
  - function accept(address _student) public { }
  - solo puede aceptar el owner
  - el alumno pasa a estado State.Accepted

ciclo de vida /evaluación

- el profesor puede evaluar a un alumno
  - function eval(address _student, uint _score) public
  - solo puede evaluar el owner
  - la nota es entre 0 .. 100
  - el alumno pasa a State.Evaluated
- se puede consultar la nota
  - function getScore() public returns(uint) { }
  - el alumno debe estar Accepted o Evaluated
  - cada alumno solo puede ver su propia nota
  - si el alumno está evaluado, se devuelve su nota en este curso
  - si no está evaluado, se le calcula la media de los subcursos
  - si no hay subcursos, la nota es 0

trabajo

- escriba el contrato en solidity
  - el estado interno es opinable
  - los nombres de las funciones deben respetarse
  - puede haber funciones privadas
- prepare 5 casos de prueba
- despliegue el curso en la red de pruebas
  - si el grupo son N alumnos, debe haber N cursos, 1 referenciando a los otros N-1

trabajo

- entregue (email a jmanas@dit.upm.es)
  - código del contrato
  - pruebas
  - dirección del despliegue de los contratos
- fecha
  - < miércoles 26.9.2018

proyecto avanzado

OPCIONAL

- objetivo: trabajar con dinero
- cada curso tiene un coste
  - constructor(string _name, uint _cost) public { }
  - function getCost() public returns(uint) { }
  - el dinero pasa directamente a la dirección del owner
- el alumno tiene que abonar el curso al registrarse
  - function register(address _student) public payable { }
  - debe ser suficiente para cubrir el coste del curso al que se apunta
    - obviando los subcursos
    - si paga de más, se siente