Desarrollo Blockchain Ethereum con Solidity

Módulo 6 - Implementación con Solidity

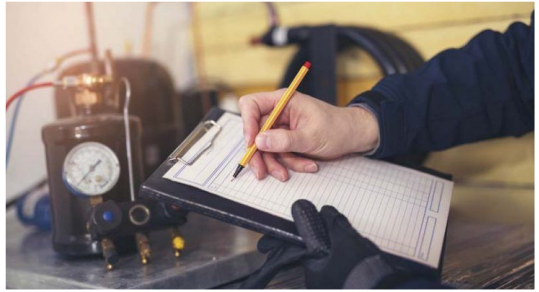
Implementación con Solidity

Solidity para Testing

- Respeta la sintaxis de solidity que utilizamos para escribir los contratos.
- Las pruebas en sí van a estar escritas como funciones dentro de los contratos.
- Tiene las mismas características que los test en Javascript.
- Crea un entorno limpio para cada ejecución de los test.
- Permite acceder a contratos ya implementados e implementar nuevos.

Consideraciones

- Los contratos de prueba no deben heredar de otros contratos, garantizando así que su código está minimizado.
- La función de los contratos de prueba tiene que ser probar, ni más ni menos.
- No agregar ninguna librería externa para probar, utilizar las funciones de Assert provistas por la plataforma.
- Puede ejecutarse el ambiente de pruebas en cualquier entorno de Ethereum, sin embargo se recomienda su ejecución en entornos locales por una cuestión de performance.



Estructura

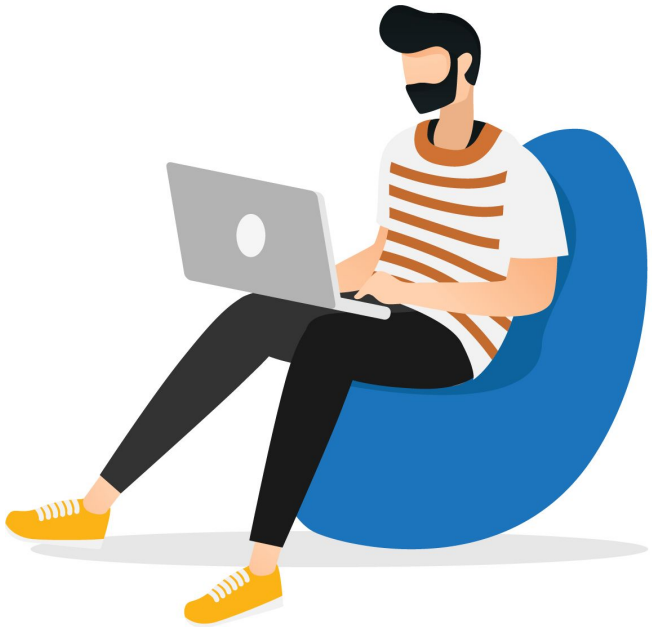
- **Assertions:** a través de la librería Assert se realizan las evaluaciones de los resultados esperados.
- **Deployed Addresses:** colección que contiene las direcciones de los contratos que ya están implementados en la red. Se accede por medio del nombre del contrato.
- El nombre del contrato de prueba tiene que empezar con la palabra Test con la T mayúscula.
- El nombre de las funciones tienen que empezar con la palabra test en minúscula.
- Existen funciones predefinidas que se ejecutan ante o después de las pruebas y que nos permiten reutilizar código como beforeEach, beforeAll, afterEach y afterAll.
- Se puede fondear al contrato automáticamente creando una variable llamada *“initialBalance”* de tipo uint y seteándole el valor de fondos deseado.

Estructura - Ejemplo

```
pragma solidity >=0.4.25 <0.6.0;

import "truffle/Assert.sol";
import "truffle/DeployedAddresses.sol";
import "../contracts/MetaCoin.sol";

contract TestMetaCoin {
  function testInitialBalanceUsingDeployedContract() {
    MetaCoin meta = MetaCoin(DeployedAddresses.MetaCoin());

    uint expected = 10000;

    Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
  }

  function testInitialBalanceWithNewMetaCoin() {
    MetaCoin meta = new MetaCoin();

    uint expected = 10000;

    Assert.equal(meta.getBalance(tx.origin), expected, "Owner should have 10000 MetaCoin initially");
  }
}
```

Ejercitación

1. Armar un entorno de pruebas para el contrato de Storage de ejemplo.
2. Considerar agregar permisos de acceso de Owner al contrato para probar no sólo el almacenamiento si no también que sólo los usuarios permitidos puedan transaccionar.
3. Probar los escenarios en contratos ya implementados y también en contratos nuevos.
4. No olvidar probar el valor por defecto del contrato.



¡Muchas gracias!

¡Sigamos trabajando!