Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones Blockchain	Apellidos: Macaya Faber	22/02/2021
	Nombre: Iker	

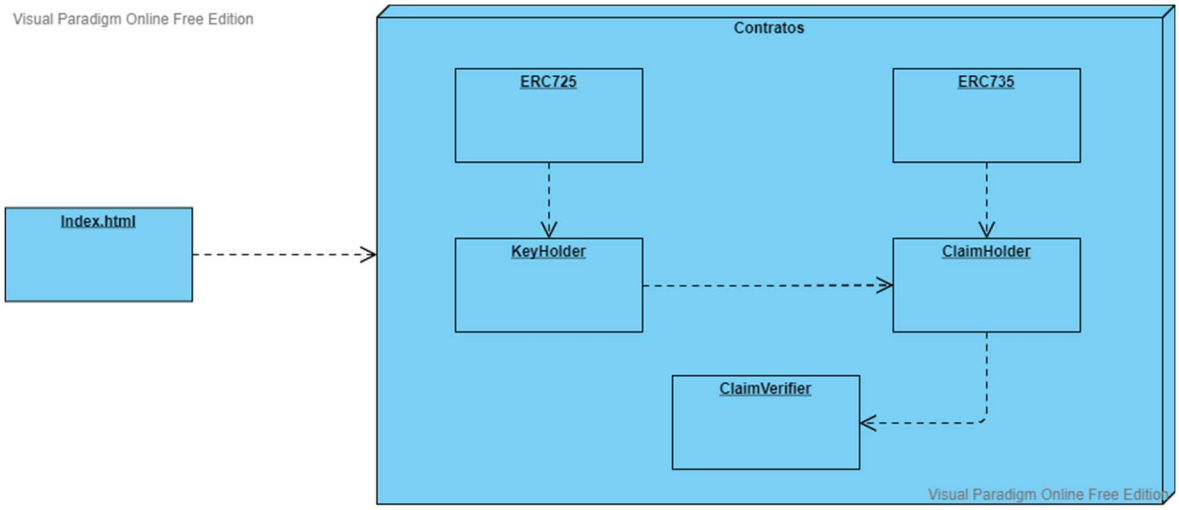
Desarrollo de Aplicaciones Blockchain – Identidad Digital

DAPP que nos va a permitir desde un frontal muy sencillo interactuar con varios Smart contratcs a través de Ganache.

Es una aplicación desarrollada con propósito educativo. Por eso hay una serie de características que están imlementadas de manera muy básica y diversas salidas las tenemos por console.

Diagrama de despliegue:

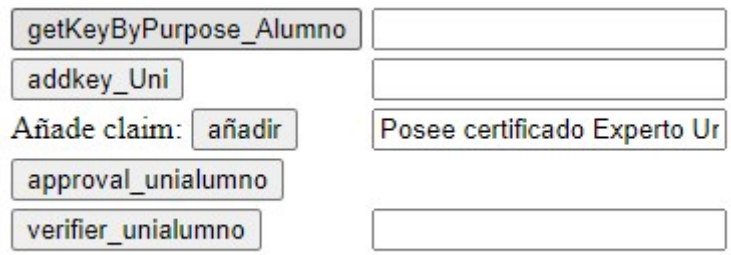
Tenemos el index.html que hace de frontal y 5 contratos: ERC725, ERC735, KeyHolder, ClaimHolder y ClaimVerifier.



Instrucciones de despliegue:

Resumen de instrucciones de despliegue (se ve en detalle en el manual de usuario):

1. Desplegar los Smart contracts en remix.
2. Abrir Ganache.
3. Asignar cada una de las address a modo hardcoded en el index.html.
4. Levantar un servidor web con el comando: python -m http.server
5. Conectarnos a localhost:8000
6. Ejecutar los métodos de la DAPP:



Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones Blockchain	Apellidos: Macaya Faber	22/02/2021
	Nombre: Iker	

Diagrama de secuencia:

Tenemos 3 actores: Alumno, Universidad y Empresa

Y por cada uno de ellos una instancia al contrato correspondinte:

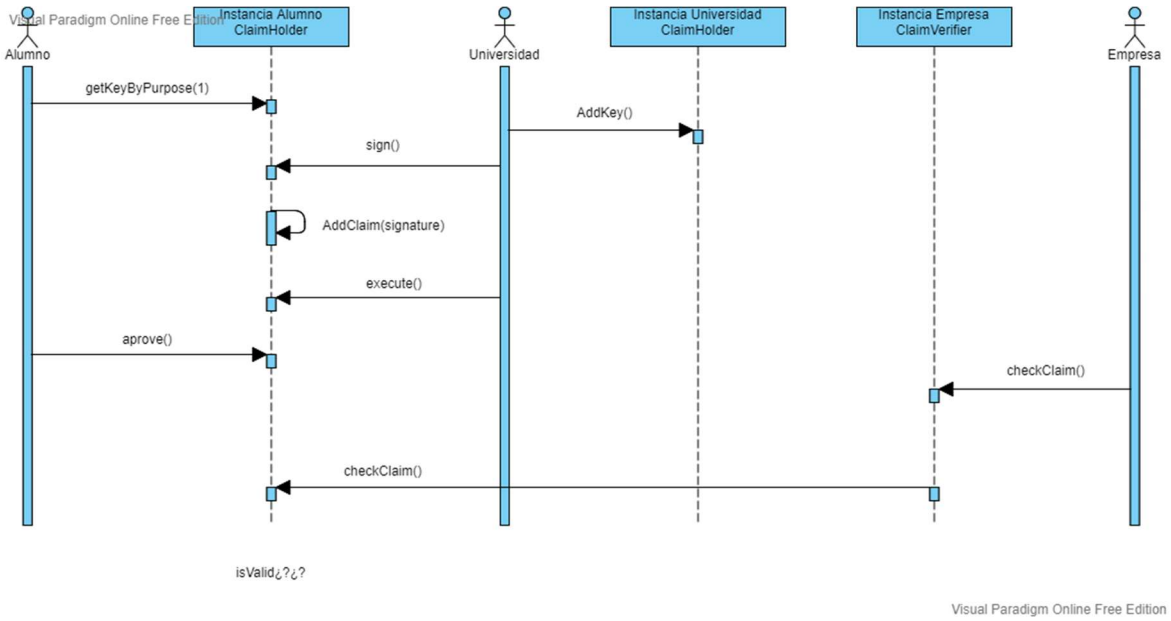
InstanciaAlumno: ClaimHolder

InstanciaUniversidad: ClaimVerifier

InstanciaEmpresa: ClaimVerifier (pasandole el address del contrato de la Universidad)

Resumen de la DAPP:

1. El alumno ejecuta el método getKeyByPurpose(1) para coger una de las keys generadas automáticamente por el contrato claimholder.
2. Añadimos una key a la universidad para firmar alegaciones, usando su instancia del contrato ClaimHolder.
3. Generamos la firma que vamos a usar en el addClaim desde la cuenta de la universidad sobre la instancia del alumno.
4. Lanzamos el addClaim con la firma generada anteriormente.
5. Ejecutamos la alegación el método execute.
6. El alumno aprueba (método aprove) la alegación (claim) ejecutada (execute) por la Universidad.
7. La empresa desde su instancia de contrato Claim Verifier lanza el método checkclaim sobre el contrato del alumno para verificar que la alegación es correcta o no.

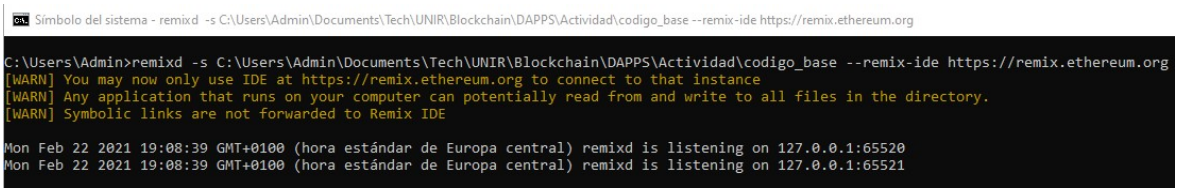


Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones Blockchain	Apellidos: Macaya Faber	22/02/2021
	Nombre: Iker	

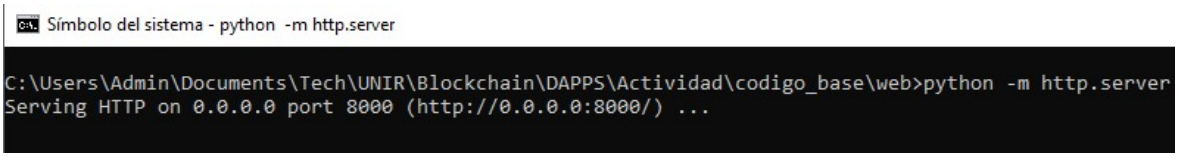
Manual de usuario:

1. Activar el demonio de remixd con el comando:

   remixd -s Ruta_a_los_ficheros\codigo_base --remix-ide https://remix.ethereum.org



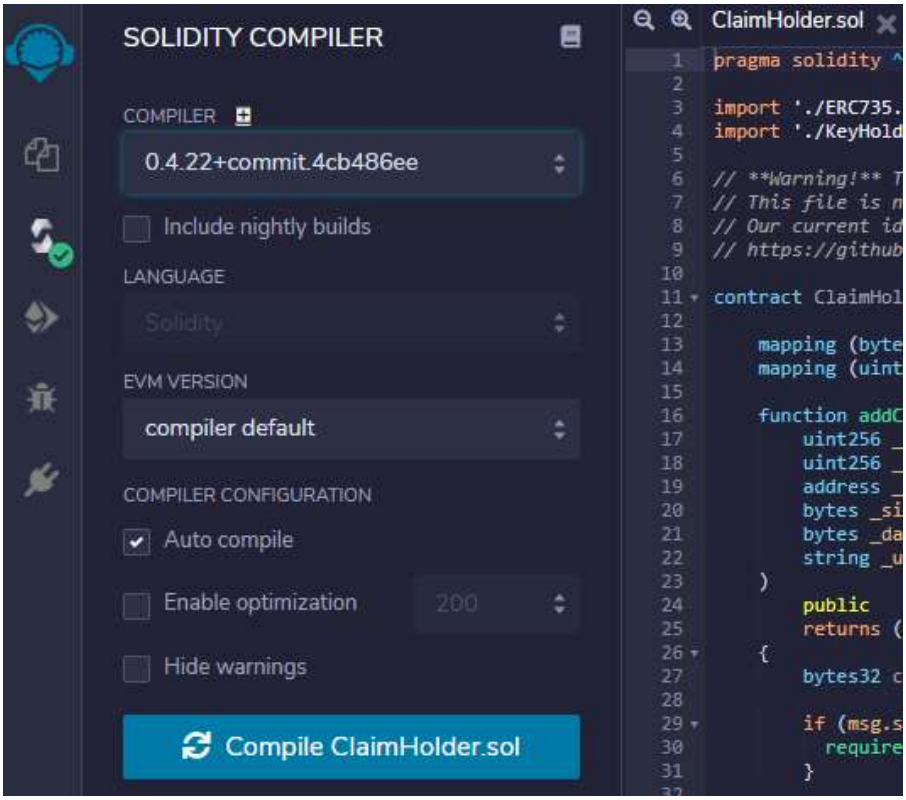
2. Levantar el servidor web: python -m http.server



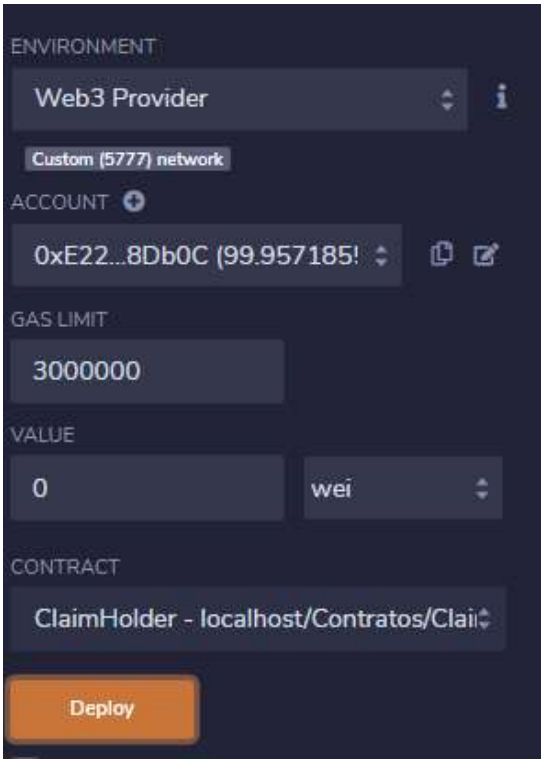
3. Abrir Ganache:

Asignatura	Datos del alumno	Fecha
Desarrollo de Aplicaciones Blockchain	Apellidos: Macaya Faber	22/02/2021
	Nombre: Iker	

4. Compilamos los contratos en remix:



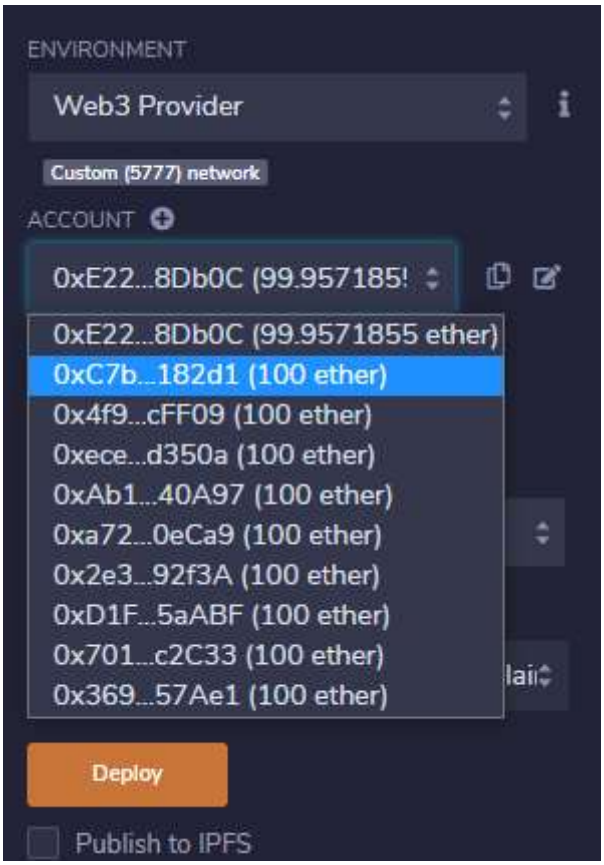
5. Desplegamos el contrato claim holder, con la primera address que nos proporciona ganache:



6. Hardcode el address del contrato en el index.html:


```
var contrato_alumno = "0xC626E18E92510cAfE623eD253818c5304F58D33E";
var contrato_uni = "";
var contrato_empresa = "";
```

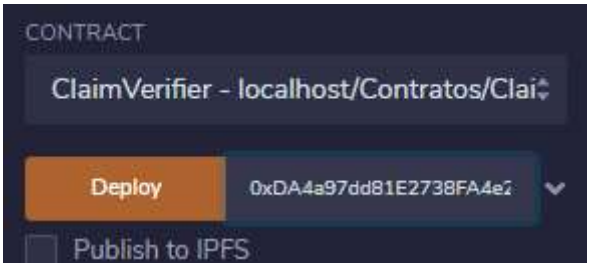
7. Mismo proceso para la instancia de la Universidad. Pero tenemos que seleccionar el segundo address que nos proporciona Ganache:



Una vez desplegado harcode el address al index.html:

```
var contrato_alumno = "0xC626E18E92510cAfE623eD253818c5304F58D33E";
var contrato_uni = "0xDA4a97dd81E2738FA4e2B9850e3b23C050e9740d";
var contrato_empresa = "";
```

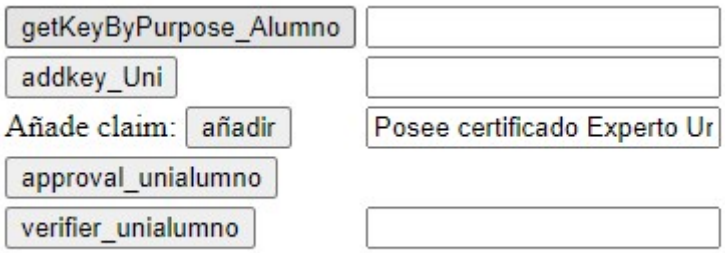
8. Desplegamos el contrato Claim Verifier como instancia de la Empresa. Para ello tenemos que coger el address 3 que nos proporciona Ganache y pasarle al contrato el address del contrato de la Universidad:



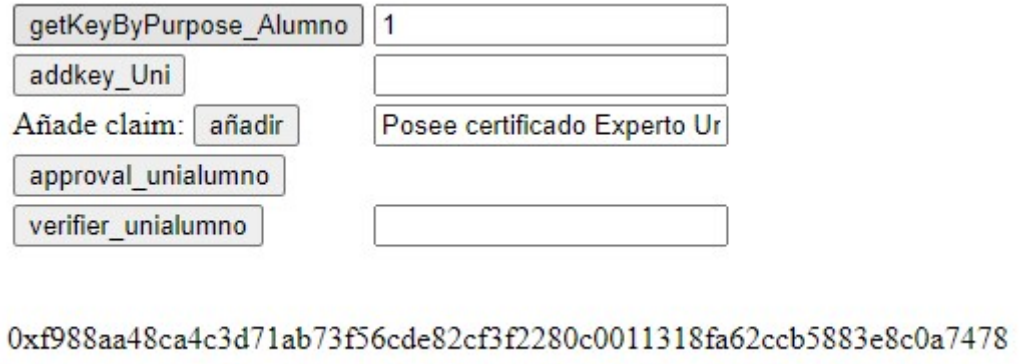
El addrees del contrato lo copiamos en el html:

```
//Contratos
var contrato_alumno = "0xC626E18E92510cAfE623eD253818c5304F58D33E";
var contrato_uni = "0xDA4a97dd81E2738FA4e2B9850e3b23C050e9740d";
var contrato_empresa = "0x53c876a54666184A7789A76d8701DFaB8Efec86f";
```

9. Ya tenemos los contratos desplegados. Podemos interactuar con ellos desde el index.html. Abrimos en el explorador: localhost:8000

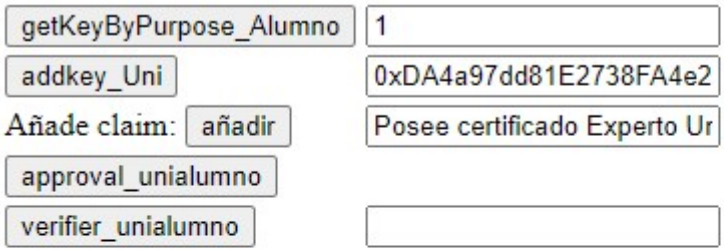


10. Comprobamos la key del alumno:



11. Añadimos la key a la Universidad pasándole el address de su contrato:

12. Con el botón claim podemos añadir las alegaciones que queramos:



0xf988aa48ca4c3d71ab73f56cde82cf3f2280c0011318fa62ccb5883e8c0a7478
0x11c06111fbc3d87f6ef535d749a6a471226512b0bc2a8443a73c3cf5cba5d1b9
0xeb857deb2c4998d32506babd33ceae5c03006a4f4767a97d347e7933f0b1b486
0x0f9206d647a5c65832e1c1bf2048f986f44ca11a9da923384a98b1f0a622f49d

13. Con el botón approval_unialumno comprobamos si el alumno aprueba la alegación. La salida esta por console:

```
0x74b5c53b9d7c6d2495b5154db3a20a3e0625afeed6922821abdb6943d7ed75c1
Executed -> OK
```

14. Con el botón verifier_unialumno comprobamos si la alegación es válida:


```
0xc207e5bc6fa16c4ead93966a07a9320534fa3dd8c86538cb117a1df8bc9e08cb
Claim -> OK
```

Si la verificación no es correcta también nos lo indica:


```
0x1f7c7dd2b0f28d1182753c05e1cc54df52e2f08fcd316360592b7931a6ba9211
Claim -> FAIL
```