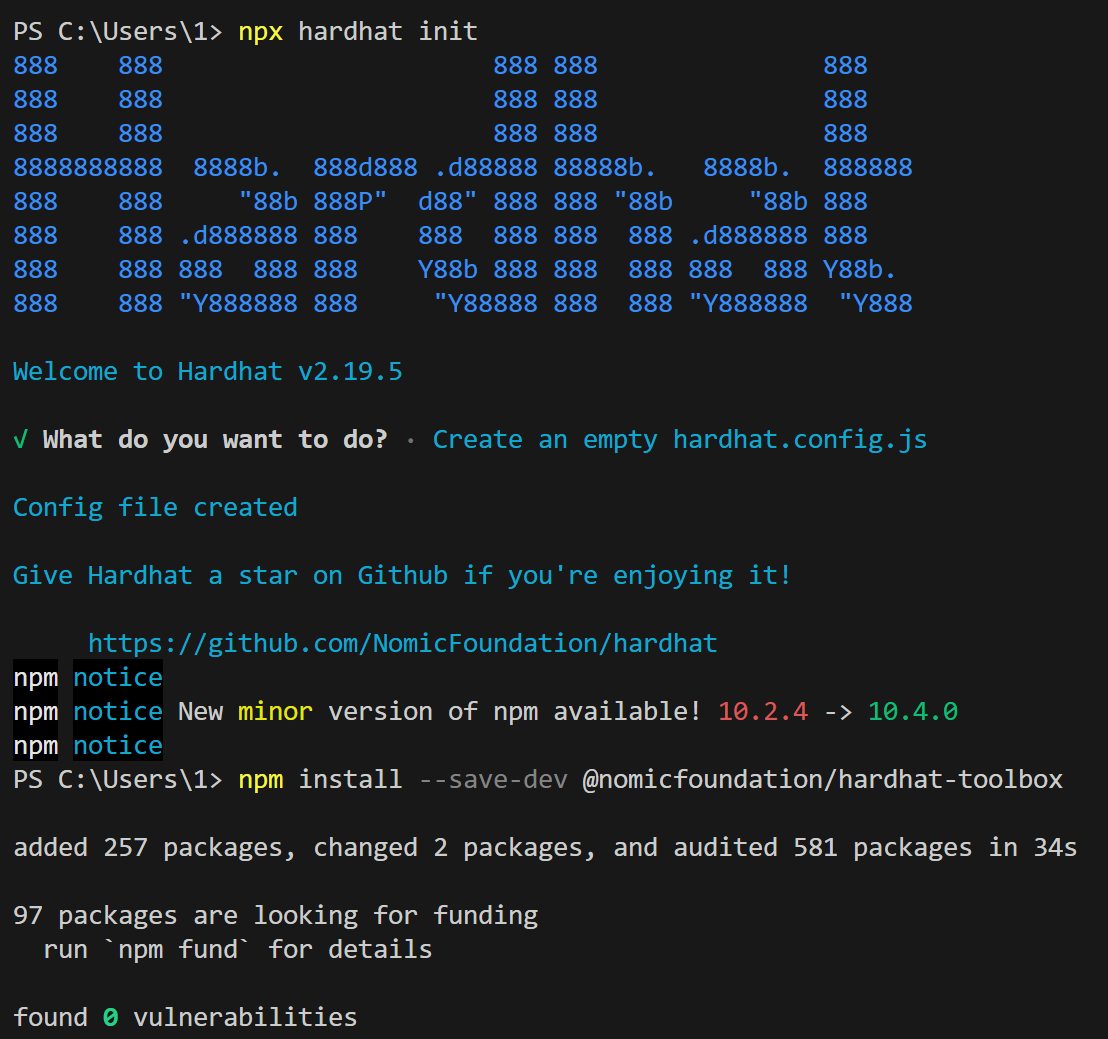
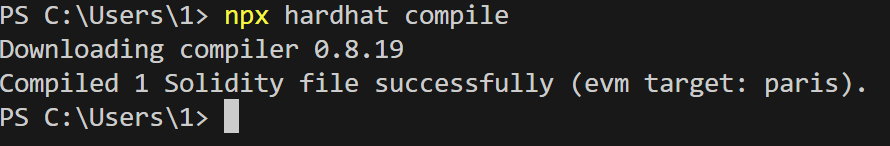
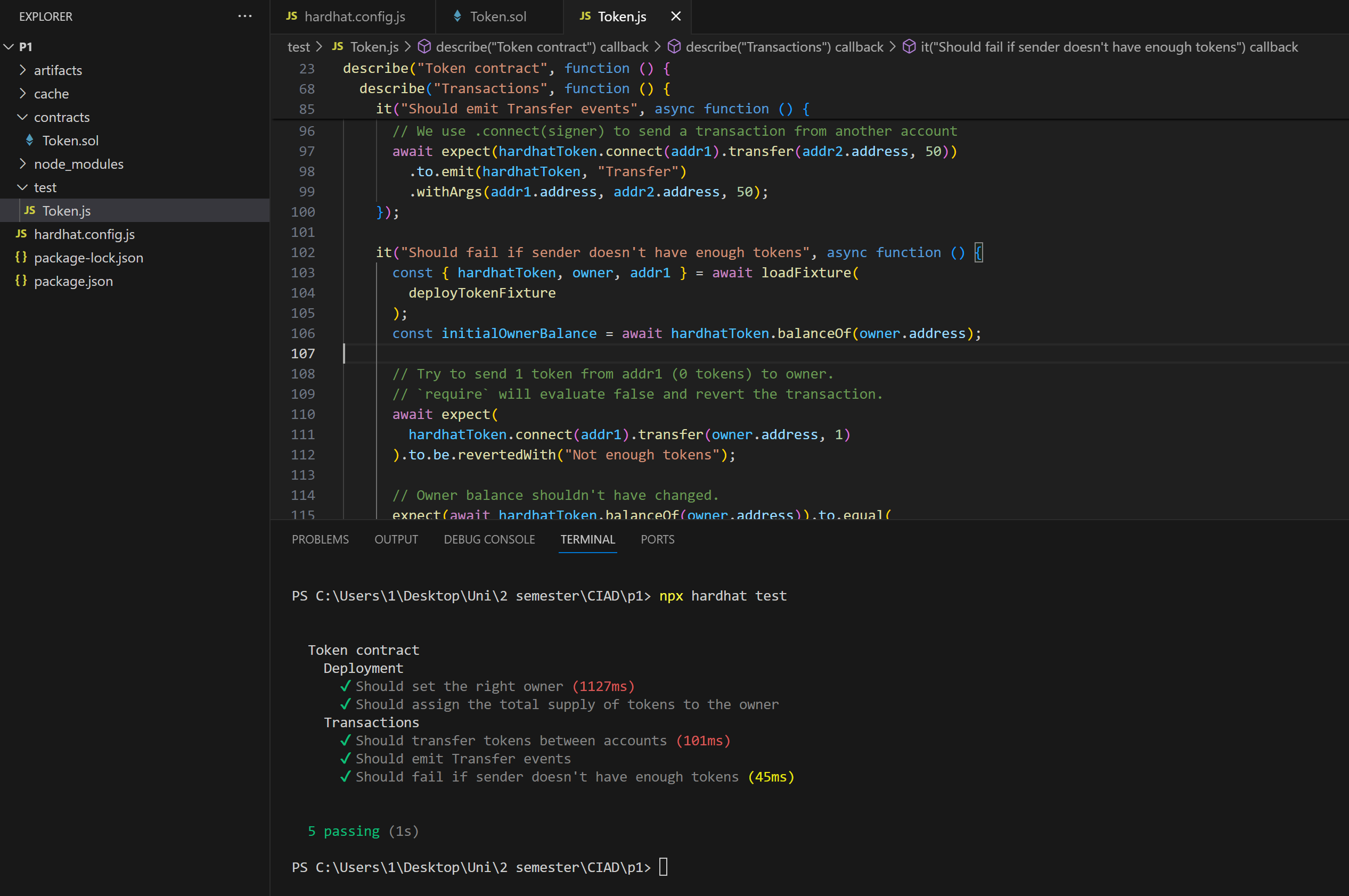
Enclace a github: <https://github.com/Skey3n/CIADlabs>

Instalación de Hardhat: Hardhat se instaló utilizando npm en un nuevo directorio. Se ejecutaron los siguientes comandos en la terminal:





1. **Creación del Directorio y Archivo del Contrato**:
   * Cree un nuevo directorio llamado **contracts**.
   * Dentro del directorio, cree un archivo llamado **Token.sol**.
2. **Código del Contrato**:
   * Pegue el siguiente código en el archivo **Token.sol**. Tómese un momento para leer el código, ya que está lleno de comentarios explicando los conceptos básicos de Solidity.



1. **Uso de Hardhat Network**:

Vamos a utilizar Hardhat Network, una red Ethereum local diseñada para desarrollo. Viene integrada con Hardhat y se utiliza como la red predeterminada. No es necesario configurar nada para usarla.

1. **Creación de Directorio para Pruebas**:
   * Cree un nuevo directorio llamado **test** dentro del directorio raíz del proyecto.
   * Dentro de ese directorio, cree un nuevo archivo llamado **Token.js**.

**Explicación del Código**

Fixture (deployTokenFixture):

Hemos creado una función deployTokenFixture que realiza la configuración necesaria y devuelve los valores que utilizaremos en las pruebas.

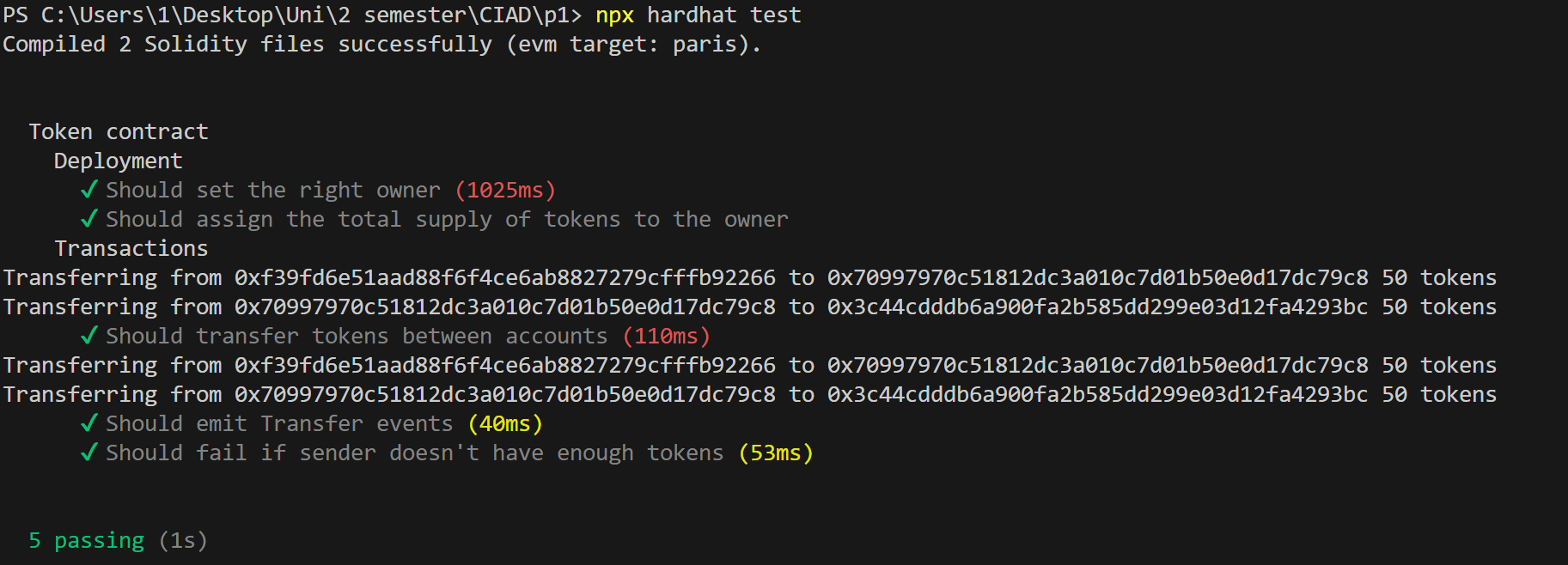
Prueba 1 (Should assign the total supply of tokens to the owner):

Verifica que el saldo del propietario sea igual al suministro total de tokens.

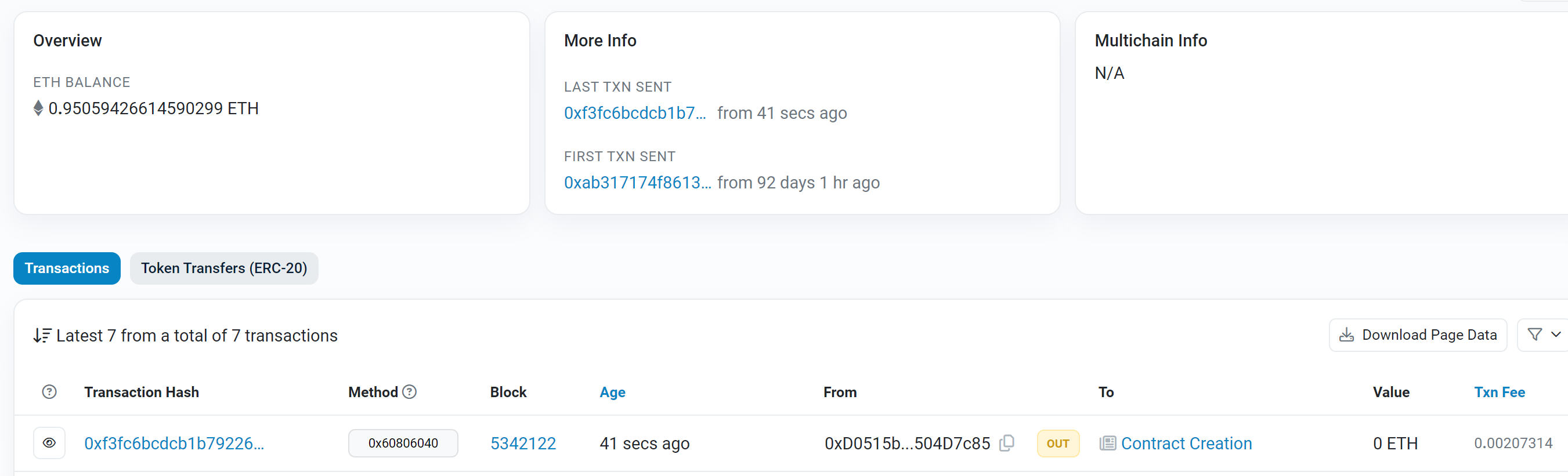
Prueba 2 (Should transfer tokens between accounts):

Verifica que la transferencia de tokens entre cuentas se realiza correctamente.

After importing console.log and editing the code, we run test



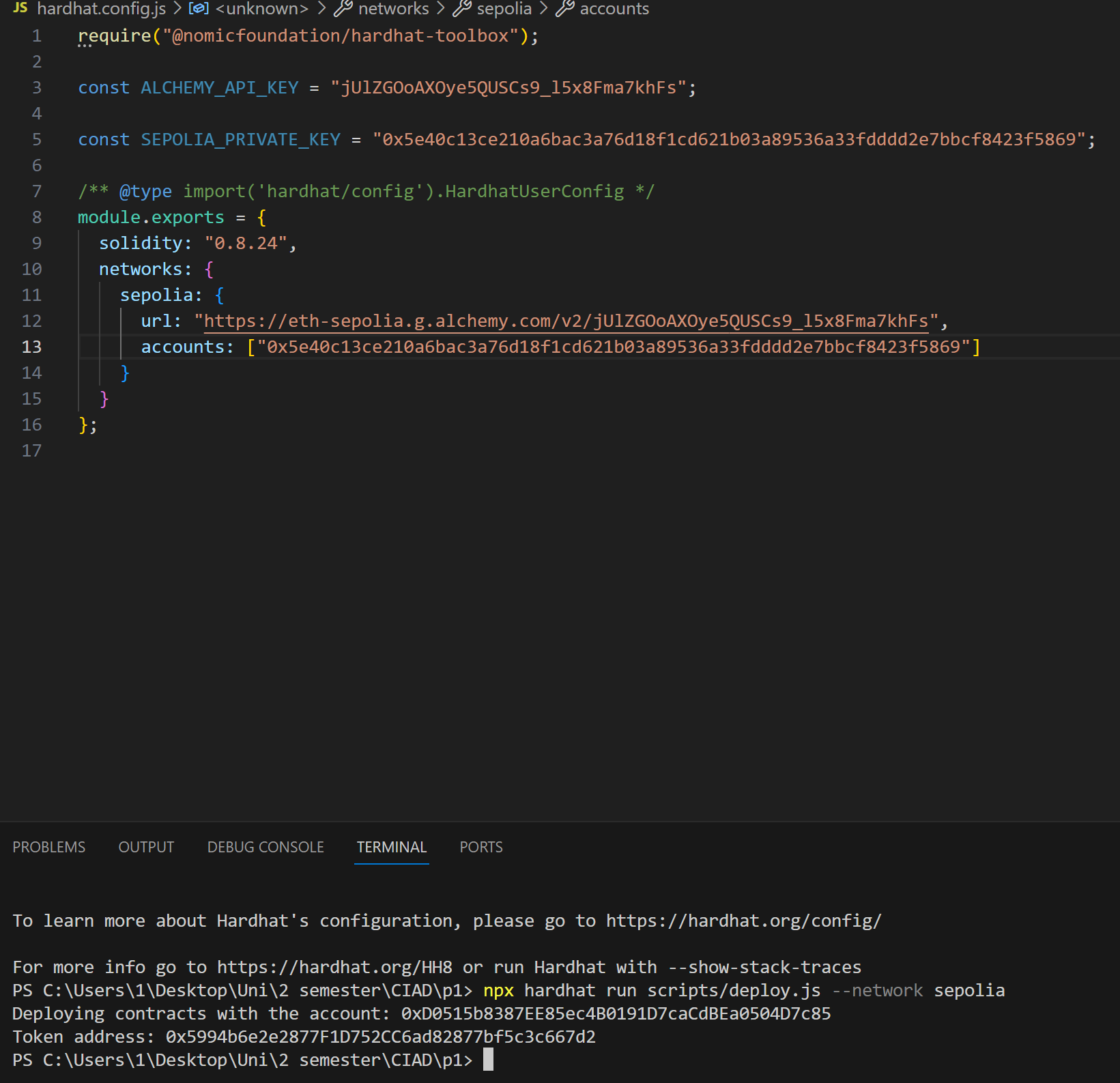
Antes de ejecutar el script de despliegue, necesito tener algo de ether de Sepolia para realizar la transacción. Obtuve ether de prueba de un faucet de Sepolia.



Ejecutando nuevamente el siguiente comando para desplegar el contrato en la red Sepolia:

npx hardhat run scripts/deploy.js --network sepolia

lo Podemos comporbar



Prueba de aplicacion de hardhat.

Iniciamos una red local de Hardhat con el comando npx hardhat node.

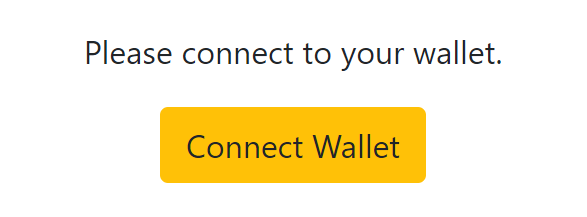
En otra terminal en el mismo directorio, implementamos el contrato en la red local de Hardhat con el comando npx hardhat --network localhost run scripts/deploy.js.

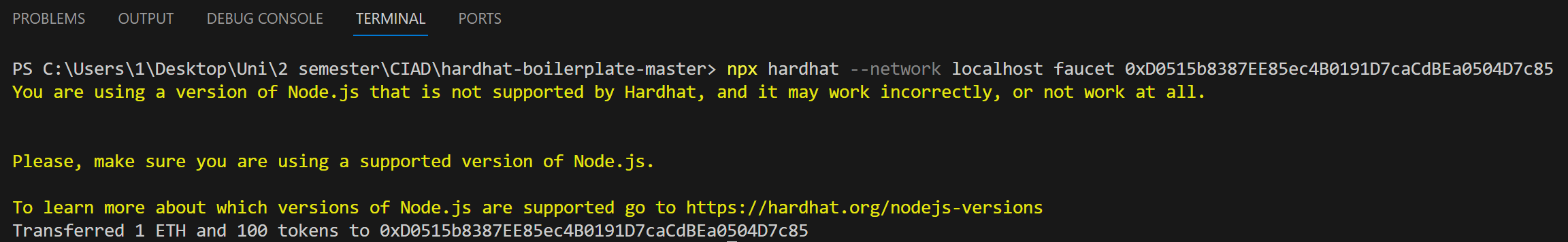
Iniciamos la aplicación web React para interactuar con el contrato, yendo al directorio frontend, instalamos las dependencias con npm install y ejecutamos la aplicación con npm run start.

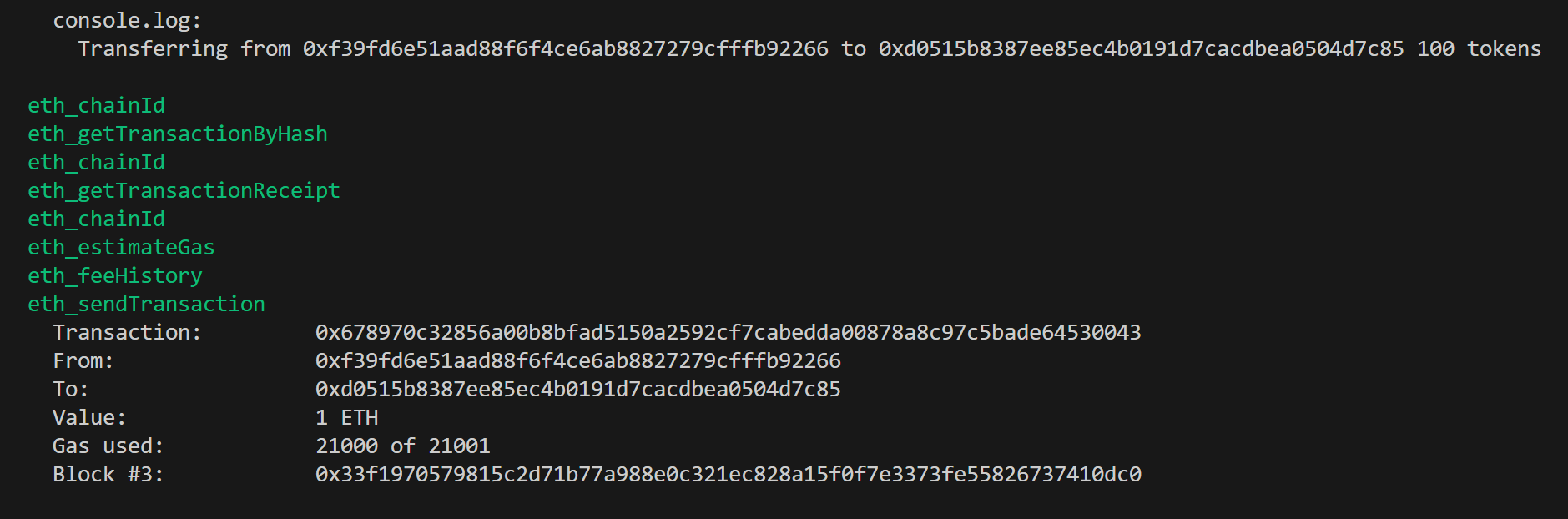
Conectamos una billetera a la aplicación web y obtuvimos acceso a la funcionalidad del contrato.

Recargamos el saldo de la dirección de la billetera con el comando npx hardhat --network localhost faucet, para tener la posibilidad de enviar tokens a otra dirección.

Verificamos que la transferencia de tokens se realizó con éxito, observando la información de la transferencia en la consola y viendo el saldo actualizado en la aplicación web.



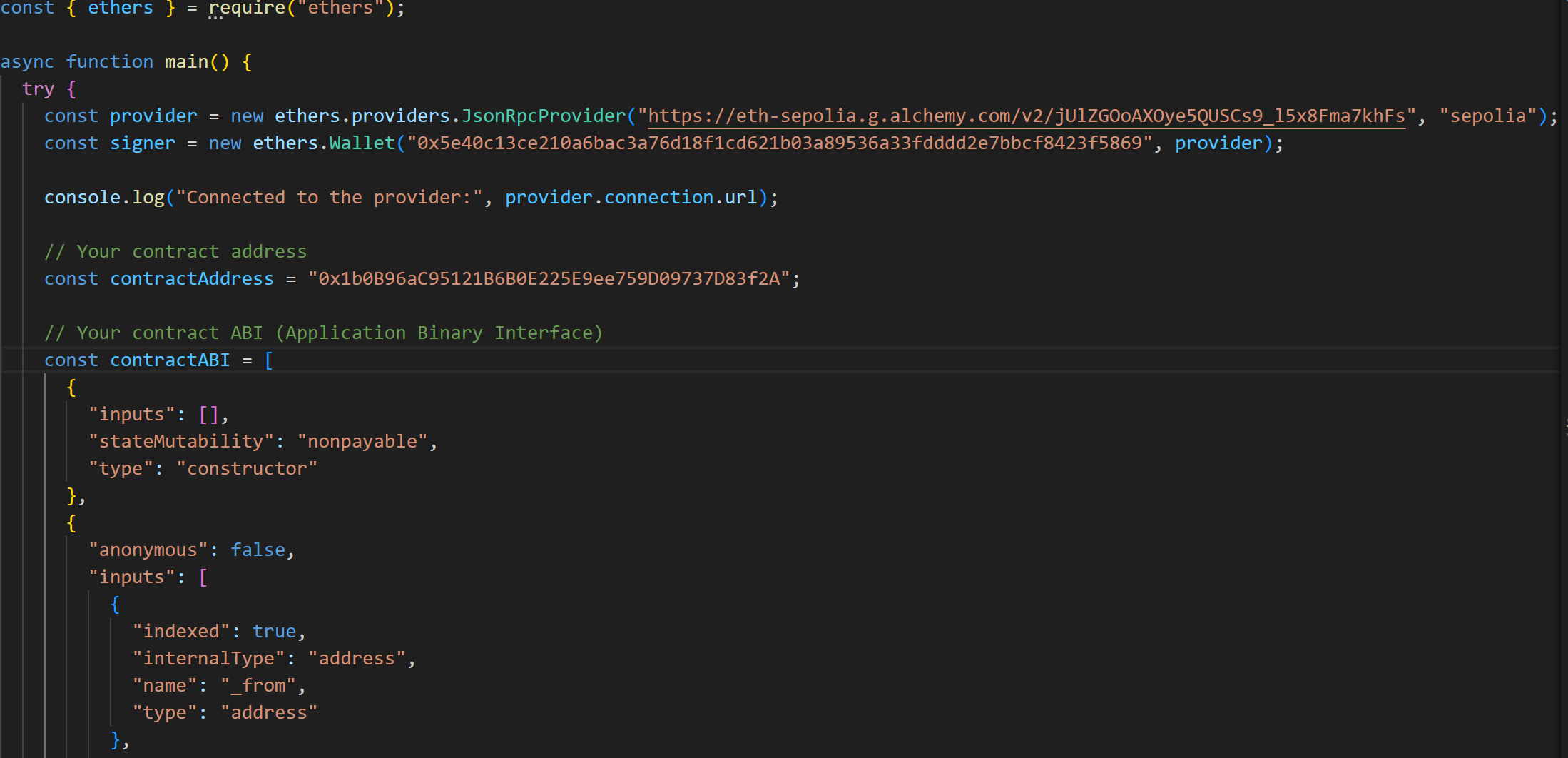


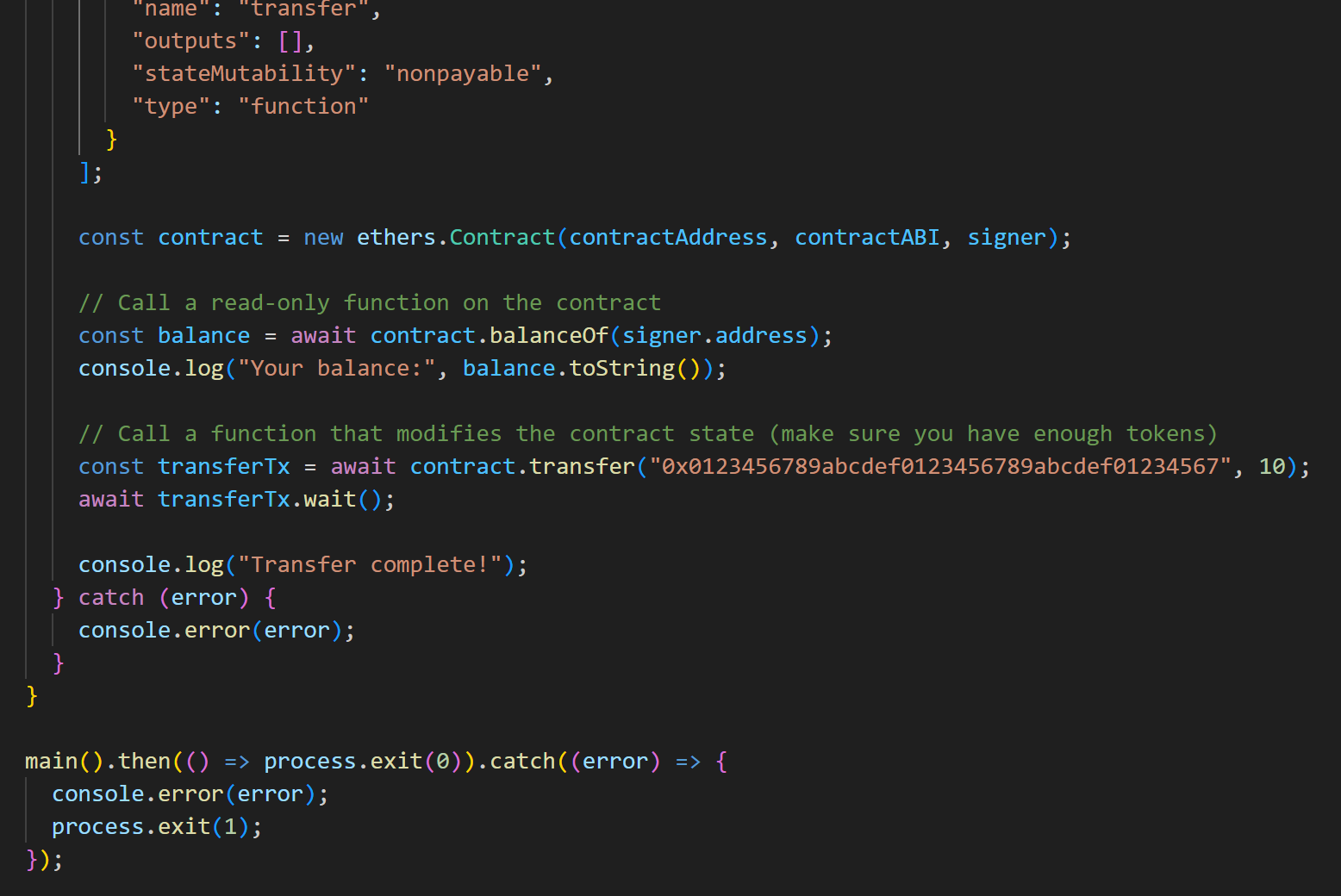


# Mi propia aplicacion.

Yo he creado el archivo cli.js para desarrollar una aplicación de línea de comandos que interactúa con un contrato inteligente en la red Ethereum. En este archivo, he utilizado la biblioteca ethers.js para establecer una conexión con un proveedor de nodos de Ethereum. He especificado la URL del proveedor como "https://eth-sepolia.g.alchemy.com/v2/jUlZGOoAXOye5QUSCs9\_l5x8Fma7khFs". Además, he proporcionado la dirección del contrato inteligente (contractAddress) y su ABI (contractABI), que define la interfaz del contrato.

En la función main(), he creado una instancia de la clase ethers.Contract para interactuar con el contrato en la red. He utilizado un ejemplo de dirección de destinatario ("0x0123456789abcdef0123456789abcdef01234567") y he llamado a la función transfer del contrato para realizar una transacción de tokens.





Y el resultado nos demuestra la linea de codigo.

