Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Фізико-технічний інститут

«Блокчейн та децентралізовані системи»

Лабораторна робота №3

Дослідження безпечної реалізації та експлуатації децентралізованих додатків.

Виконали:

студенти групи ФБ-42мп

Андреєв Данило

Косигін Олександр

Зінов’єв Андрій

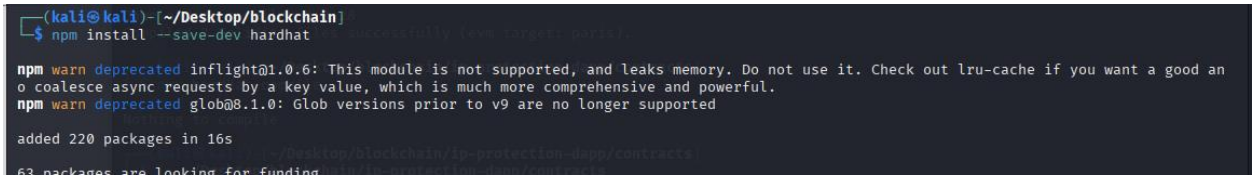
Київ — 2025

Мета роботи: отримання навичок роботи із децентралізованими додатками та оцінка безпеки інформації при їх функціонуванні

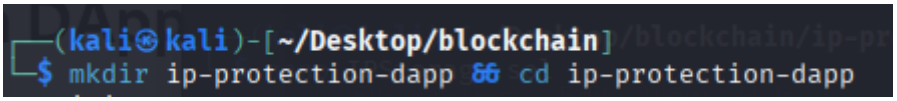
Розробка децентралізованого додатку (наприклад, захисту інтелектуальної власності цифрового контенту) на обраній системі децентралізованих додатків.

Студенти самостійно обирають будь-яку з існуючих систем децентралізованих додатків на базі блокчейн

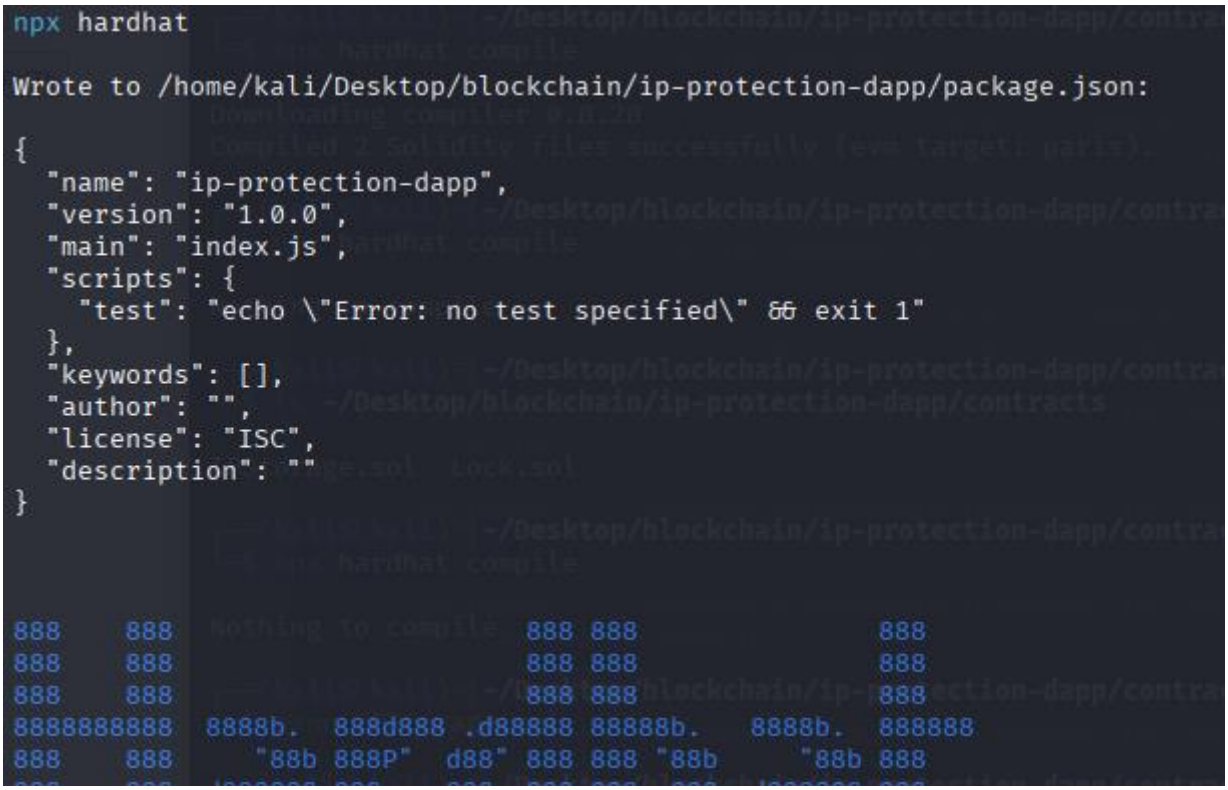
Білдимо Hardhat – утиліта для розробки та впровадження смарт контрактів



Створюємо папки для наших проектів



Вмикаємо Hardhat, обираємо джаваскрипт



Встановлюємо необхідні ліби для роботи з ефіром

```
┌──(kali㉿kali)-[~/Desktop/blockchain/ip-protection-dapp]
└─$ npm install --save ethers
npm install --save-dev @nomicfoundation/hardhat-toolbox
npm install web3


up to date, audited 577 packages in 1s

103 packages are looking for funding
  run `npm fund` for details

13 low severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

Some issues need review, and may require choosing
a different dependency.
```

Пишемо свій смарт контракт, вказуємо версію і тд

Про всяк випадок чистимо систему від інших контрактів, компілюємо свій

```
┌──(kali㉿kali)-[~/Desktop/blockchain/ip-protection-dapp/contracts]
└─$ npx hardhat clean
npx hardhat compile

Compiled 2 Solidity files successfully (evm target: paris).
```

Пишемо код який автоматично задеплоить контракт у майбутній блокчейн

```
 GNU nano 8.4                                                scripts/deploy.js *
const hre = require("hardhat");

async function main() {
  const [deployer] = await hre.ethers.getSigners();

  console.log("Deploying contract with account:", deployer.address);

  const IPStorage = await hre.ethers.getContractFactory("IPStorage");
  const ipStorage = await IPStorage.deploy();

  await ipStorage.waitForDeployment();

  console.log("IPStorage deployed to:", await ipStorage.getAddress());
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Створюємо та деплоїмо фронтенд частину нашого завдання

```jsx
import { useState } from "react";
import { ethers } from "ethers";
import IPStorageAbi from "./abi/IPStorage.json";
import { CONTRACT_ADDRESS } from "./config";

function App() {
  const [ipfsHash, setIpfsHash] = useState("");
  const [owner, setOwner] = useState("");
  const [timestamp, setTimestamp] = useState("");
  const [status, setStatus] = useState("");

  const handleRegister = async () => {
    try {
      const provider = new ethers.JsonRpcProvider("http://127.0.0.1:8545");
      const signer = await provider.getSigner(0);

      const contract = new ethers.Contract(CONTRACT_ADDRESS, IPStorageAbi.abi, signer);

      const tx = await contract.registerContent(ipfsHash);
      await tx.wait();

      setStatus("Контент зарегистрирован!");
    } catch (err) {
      setStatus("Ошибка: " + err.message);
    }
  };

  const handleFetch = async () => {
    try {
      const provider = new ethers.JsonRpcProvider("http://127.0.0.1:8545");
      const contract = new ethers.Contract(CONTRACT_ADDRESS, IPStorageAbi.abi, provider);

      const content = await contract.getContent(ipfsHash);
      setOwner(content.owner);
      setTimestamp(new Date(content.timestamp * 1000).toLocaleString());
      setStatus("Данные получены");
    } catch (err) {
      setStatus("Ошибка: " + err.message);
    }
  };

```

```
40  };
41
42  return (
43    <div style={{ padding: 20 }}>
44      <h2>IP Protection DApp</h2>
45      <input
46        type="text"
47        placeholder="IPFS Hash"
48        value={ipfsHash}
49        onChange={(e) => setIpfsHash(e.target.value)}
50      />
51      <div>
52        <button onClick={handleRegister}>Зарегистрировать</button>
53        <button onClick={handleFetch} style={{ marginLeft: 10 }}>Проверить</button>
54      </div>
55      <p>{status}</p>
56      {owner && <p>Владелец: {owner}</p>}
57      {timestamp && <p>Дата регистрации: {timestamp}</p>}
58    </div>
59  );
60 }
61
62 export default App;
63
```

```
┌──(kali㉿kali)-[~/Desktop/blockchain/ip-protection-dapp/frontend]
└─$ npm install ethers dotenv

up to date, audited 1476 packages in 11s

279 packages are looking for funding
  run `npm fund` for details
```

Запускаємо нашу локальну блокчейн-мережу, в якості доказу роботи – вивело гаманці з тестовим балансом

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/blockchain/ip-protection-dapp]
└─$ npx hardhat node

Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/

Accounts
========

WARNING: These accounts, and their private keys, are publicly known.
Any funds sent to them on Mainnet or any other live network WILL BE LOST.

Account #0: 0×f39Fd6e51aad88F6F4ce6aB8827279cffFb92266 (10000 ETH)
Private Key: 0×ac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80

Account #1: 0×70997970C51812dc3A010C7d01b50e0d17dc79C8 (10000 ETH)
Private Key: 0×59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d

Account #2: 0×3C44CdDdB6a900fa2b585dd299e03d12FA4293BC (10000 ETH)
Private Key: 0×5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a

Account #3: 0×90F79bf6EB2c4f870365E785982E1f101E93b906 (10000 ETH)
Private Key: 0×7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6

Account #4: 0×15d34AAf54267DB7D7c367839AAf71A00a2C6A65 (10000 ETH)
Private Key: 0×47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a

Account #5: 0×9965507D1a55bcC2695C58ba16FB37d819B0A4dc (10000 ETH)
Private Key: 0×8b3a350cf5c34c9194ca85829a2df0ec3153be0318b5e2d3348e872092edffba

Account #6: 0×976EA74026E726554dB657fA54763abd0C3a0aa9 (10000 ETH)
Private Key: 0×92db14e403b83dfe3df233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e

Account #7: 0×14dC79964da2C08b23698B3D3cc7Ca32193d9955 (10000 ETH)
```

Далі отримаємо адресу нашого контракту

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[~/Desktop/blockchain/ip-protection-dapp]
└─$ npx hardhat run scripts/deploy.js --network localhost

Deploying contract with account: 0×f39Fd6e51aad88F6F4ce6aB8827279cffFb92266
IPStorage deployed to: 0×5FbDB2315678afecb367f032d93F642f64180aa3

┌──(kali㉿kali)-[~/Desktop/blockchain/ip-protection-dapp]
└─$
```

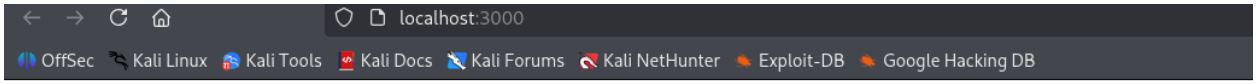
Яку вставляємо у конфіг файл

```
File  Actions  Edit  View  Help
  GNU nano 8.4                                                  config.js
export const CONTRACT_ADDRESS = "0×5FbDB2315678afecb367f032d93F642f64180aa3";
```

Тим часом наш блокчейн реєструє всі запити та зміни,

```
  eth_accounts
  hardhat_metadata (20)
  eth_accounts
  hardhat_metadata (20)
  eth_blockNumber
  eth_getBlockByNumber
  eth_feeHistory
  eth_maxPriorityFeePerGas
  eth_sendTransaction
    Contract deployment: IPStorage
    Contract address:    0x5fbdb2315678afecb367f032d93f642f64180aa3
    Transaction:         0x40ce078c2108b43f860b4f1786ebb0f2f60d92581085570153b119016e3bb353
    From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
    Value:               0 ETH
    Gas used:            732648 of 30000000
    Block #1:            0x2880e1999e258ec4fb162462c879aa5c889501e78f5060f847bea8d61781f672

  eth_getTransactionByHash
  eth_getTransactionReceipt
  eth_blockNumber
  eth_chainId
  eth_accounts
  eth_call
    Contract call:       IPStorage#getContent
    From:                0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266
    To:                  0x5fbdb2315678afecb367f032d93f642f64180aa3
```

Запускаємо наш веб інтерфейс, спробуємо зареєструвати\перевірити хеш



IP Protection DApp

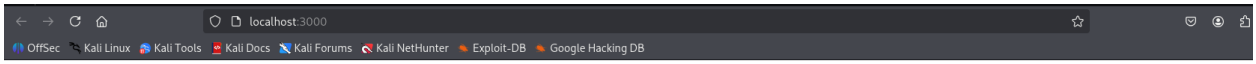
›82f6ec3e0aa620df749b749d

Зарегистрировать Проверить

Контент зарегистрирован!

Владелец: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

Бачимо що зареєструвати повторно не вийшло



IP Protection DApp

›82f6ec3e0aa620df749b749d

Зарегистрировать Проверить

Ошибка: execution reverted: "Already registered" (action="estimateGas", data="0x08c379a0000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000012416c726561647920726569737465 reason="Already registered", transaction={ "data": "0xac4d9cb2000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000406435353739633436646663633764306430643 "from": "0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266", "to": "0x5FbDB2315678afecb367f032d93F642f64180aa3" }, invocation=null, revert={ "args": ["Already registered"], "name": "Error", "signature": "Error(string)" }, code=CALL_EXCEPTION, version=6.14.1)

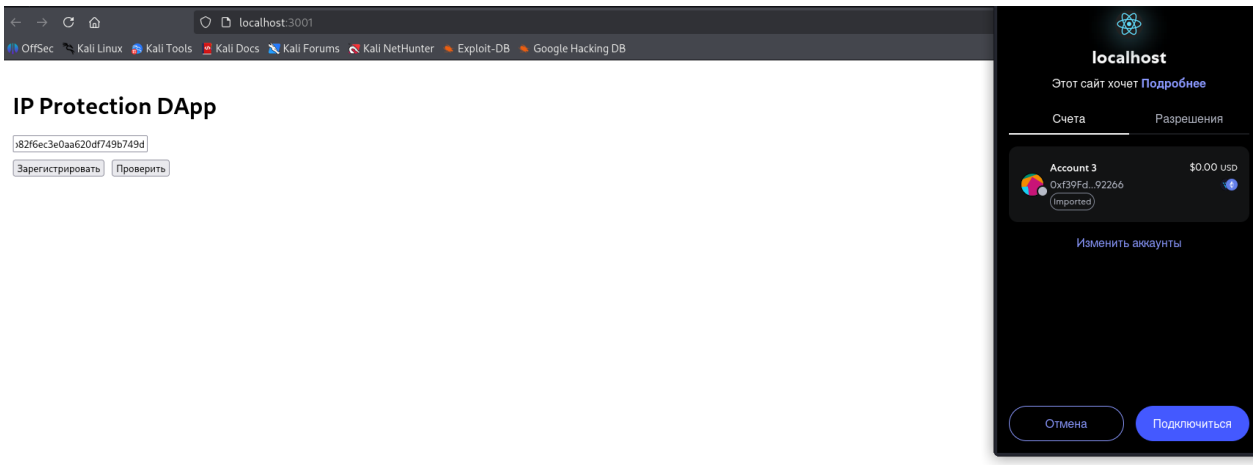
Владелец: 0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266

Консоль також про це нам повідомила

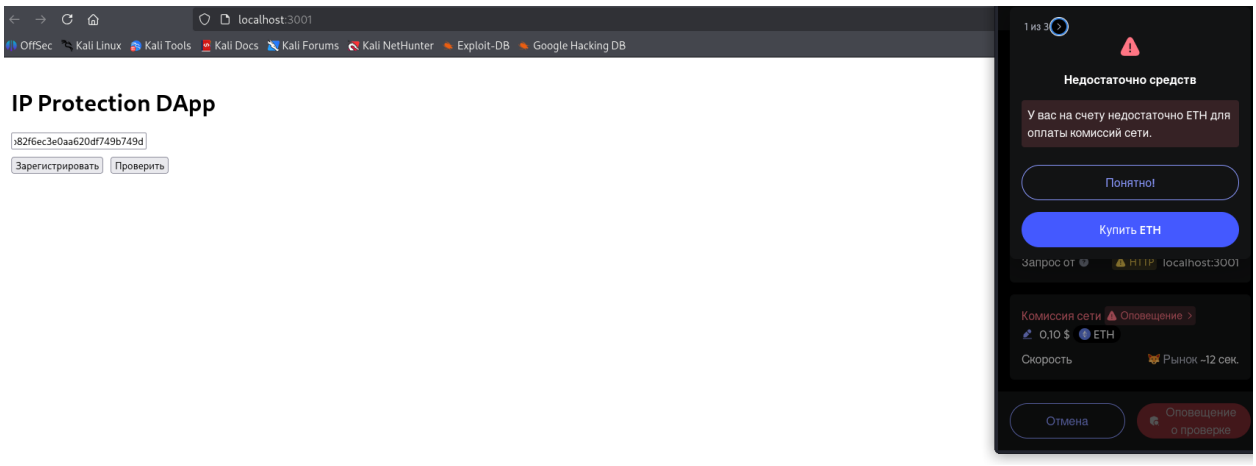
```
eth_chainId
eth_accounts
eth_blockNumber
eth_estimateGas
eth_sendTransaction
  Contract call:       IPStorage#registerContent
  Transaction:         0×e32ade15d72d4ba043f74432b742ded84ca3aafc628e1940e1189a4f63650032
  From:                0×f39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0×5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH
  Gas used:            138837 of 138837
  Block #2:            0×c94d3ba67f8394d0f241cfff08733821cfbdd3747d022b877818ed9d5e92c94c

eth_chainId
eth_getTransactionByHash
eth_chainId
eth_getTransactionReceipt
eth_chainId (2)
eth_accounts
eth_blockNumber
eth_estimateGas
  Contract call:       IPStorage#registerContent
  From:                0×f39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0×5fbdb2315678afecb367f032d93f642f64180aa3
  Value:               0 ETH

  Error: reverted with reason string 'Already registered'

eth_chainId
eth_call
  Contract call:       IPStorage#getContent
  From:                0×f39fd6e51aad88f6f4ce6ab8827279cfffb92266
  To:                  0×5fbdb2315678afecb367f032d93f642f64180aa3

eth_chainId
eth_call
  Contract call:       IPStorage#getContent
  From:                0×f39fd6e51aad88f6f4ce6ab8827279cfffb92266
```

Альтернатива

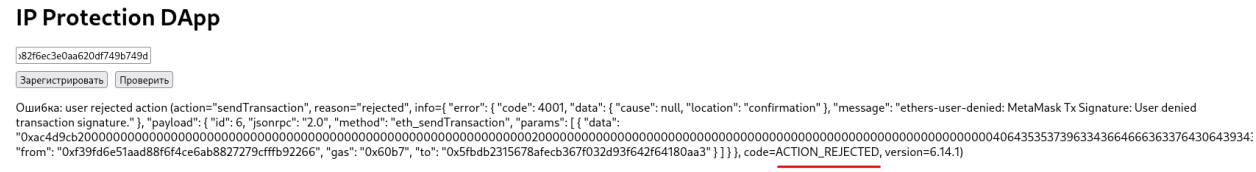
Також була спроба задеплоїти з використанням реальних, не локальних блокчейнів, і як приклад була використана утиліта MetaMask. Як ми можемо побачити на скріншоті при спробі зареєструвати чи перевірити наш хеш ми підключаємося до MetaMask (завчасно налаштувавши потрібні адреси смарт контрактів і гаманця)



Але після підключення від нас просять гроші за транзацію



І якщо ми відовляємося то отримаємо помилку “Action_Rejected”



Тобто загалом нас глобальна мережа бачить, але за рахунок відсутності грошей не виконує операцію

Також наша консоль виводить інформацію про операцію