НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

БЛОКЧЕЙН ТА ДЕЦЕНТРАЛІЗОВАНІ СИСТЕМИ

Комп'ютерний практикум №1

Розгортання систем Etherium та криптовалют

Варіант 2

Виконали:
Волинець Сергій ФІ-42мн
Сковрон Роман ФІ-42мн
Молдован Дмитро ФІ-42мн

Київ — 2025

1 Мета

Отримання навичок налаштування платформ виконання смартконтрактів та криптовалют.

2 Завдання на лабораторну роботу

Провести налаштування системи Ethereum та виконати тестові операції.

3 Дослідження

Для початку, завантажимо програмний код для роботи з блокчейном.

```
alice@Alices-MacBook-Pro ~ % brew tap ethereum/ethereum
brew install ethereum

==> Downloading https://formulae.brew.sh/api/formula.jws.json
==> Downloading https://formulae.brew.sh/api/cask.jws.json
```

Далі запускаємо тестову мережу, використовуючи команду `geth`.

```
alice@Alices-MacBook-Pro btslab % geth --sepolia --syncmode "full" --http --http.api personal,eth,net,web3
INFO [06-03|00:14:40.229] Starting Geth on Sepolia testnet...
INFO [06-03|00:14:40.231] Maximum peer count                       ETH=50 total=50
INFO [06-03|00:14:40.236] Set global gas cap                       cap=50,000,000
INFO [06-03|00:14:40.237] Initializing the KZG library             backend=gokzg
INFO [06-03|00:14:40.239] Allocated trie memory caches             clean=154.00MiB dirty=256.00MiB
INFO [06-03|00:14:40.239] Using pebble as the backing database
INFO [06-03|00:14:40.239] Allocated cache and file handles         database=/Users/alice/Library/Ethereum/sepolia/geth/chaindata cache=512.
00MiB handles=5120
```

Тепер, ми можемо під'єднатися до мережі за адресою `http://localhost:8545`.

Для простішого використання, створимо `test-node.js` файл та запустимо його. Для цього ми спочатку ініціалізуємо наш проєкт командою `npm init -y` та завантажимо додатковий пакет через `npm install web3`. Щоб запустити код на виконання ми виконаємо команду `node test_node.js`. Сам код у файлі наступний.

```
let Web3;
try {
    Web3 = require('web3').default || require('web3');
} catch (error) {
    console.error('Failed to import Web3:', error.message);
    process.exit(1);
}

async function testWithProvider(providerUrl, providerName) {
    console.log(`\n=== Testing with ${providerName} ===`);
    console.log(`URL: ${providerUrl}`);

    try {
        const web3 = new Web3(providerUrl);

        console.log('\nTesting connection...');
        const isConnected = await web3.eth.net.isListening();
        console.log('Connected:', isConnected);
```

```
    console.log('\nGetting latest block...');
    const latestBlock = await web3.eth.getBlockNumber();
    console.log('Latest block number:', latestBlock);

    console.log('\nGetting network info...');
    const networkId = await web3.eth.net.getId();
    console.log('Network ID:', networkId);

    console.log('\nGetting gas price...');
    const gasPrice = await web3.eth.getGasPrice();
    const gasPriceGwei = web3.utils.fromWei(gasPrice, 'gwei');
    console.log('Gas price:', gasPriceGwei, 'Gwei');

    console.log('\nCreating test account...');
    const account = web3.eth.accounts.create();
    console.log('New account address:', account.address);
    console.log('Private key:', account.privateKey, '\n');

    console.log('Signed transaction:')
    await web3.eth.accounts.signTransaction({
        to: '0xF0109fC8DF283027b6285cc889F5aA624EaC1F55',
        value: '1000000000',
        gas: 2000000,
        gasPrice: '234567897654321',
        nonce: 0,
        chainId: 1
    }, '0x4c0883a69102937d6231471b5dbb6204fe5129617082792ae468d01a3f362318')
    .then(console.log);

    console.log('\nChecking account balance...');
    const balance = await web3.eth.getBalance(account.address);
    const balanceEth = web3.utils.fromWei(balance, 'ether');
    console.log('Account balance:', balanceEth, 'ETH');

    console.log('\nGetting block details...');
    const block = await web3.eth.getBlock('latest');
    console.log('Block hash:', block.hash);
    console.log(
        'Block timestamp:',
        new Date(Number(block.timestamp) * 1000).toISOString()
    );
    console.log('Transaction count:', block.transactions.length);

} catch (error) {
    console.log('Error:', error.message.split('\n')[0]);
    return false;
```

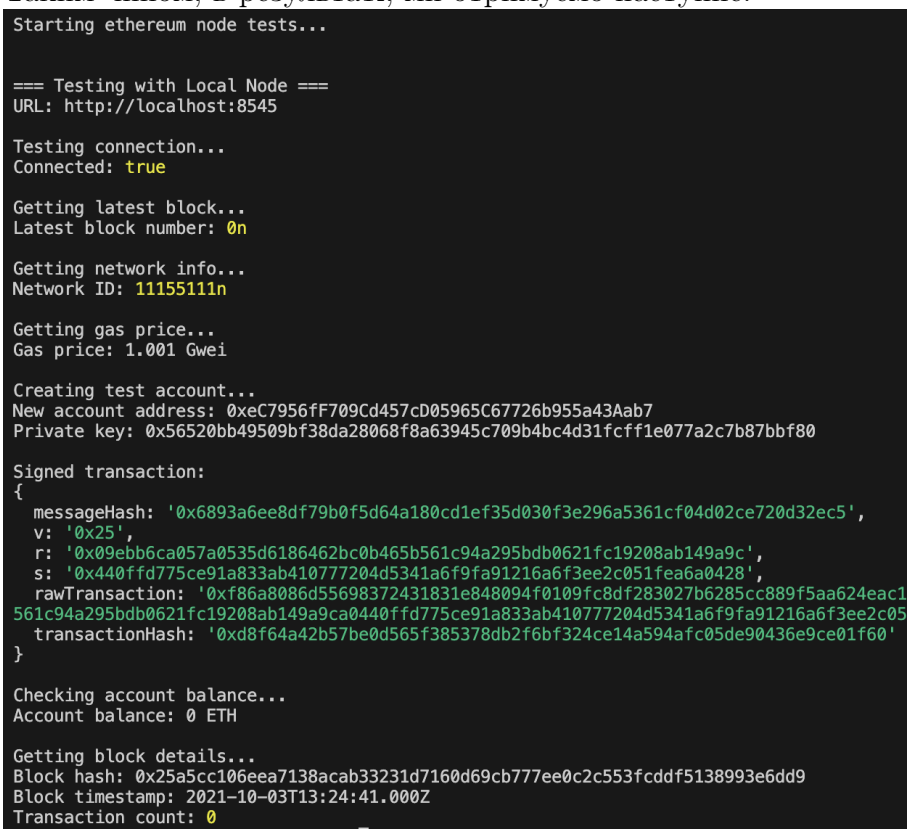
```
    }
}

async function main() {
    const providerUrl = 'http://localhost:8545';
    const providerName = 'Local Node';

    console.log('Starting ethereum node tests...\n');
    const success = await testWithProvider(providerUrl, providerName);
}

main().catch(console.error);
```

Таким чином, в результаті, ми отримуємо наступне:



4 Висновки

У процесі виконання комп'ютерного практикуму було успішно налаштовано локальне середовище для роботи з блокчейном Ethereum. Зокрема, було розгорнуто тестову мережу за допомогою `geth`, встановлено з'єднання через `Web3.js`, створено новий акаунт, перевірено баланс, зчитано інформацію про останній блок, а також здійснено підпис транзакції.

Загалом, виконання практикуму дозволило отримати практичні навички із запуску та підключення до Ethereum ноди, роботи з бібліотекою Web3.js, створенням акаунтів та взаємодії з блокчейном.