

Student name: Orynbay Gani

Group: Csse-241M

Lab Report: Blockchain Marketplace Development

## 9.1 Introduction

## 9.2 Methodology

### Environment Setup

- **Server:** Running a node server on localhost.
- **Interface:** Using Metamask to interact with the test network.
- **Configurations:** Setting up a local server for deploying and testing smart contracts.

### Smart Contract Development

- **Hello World in Solidity:** Implementing basic contract logic to display a welcome message.
- **Marketplace Contract Development:** The contract manages tokens and provides the ability to update and retrieve messages.

### API Implementation

- Several API endpoints were implemented to interact with the contract:
  - **Creating a new message:** Sending data to the contract.
  - **Retrieving messages:** Accessing information stored on the blockchain.

### Testing Procedures

- **Smart Contract Tests:** Using a local environment for unit tests.
- **API Tests:** Making requests through Postman and checking responses. Screenshots of the requests are provided below.

## 9.3 Results

- **Testing with Postman:** All API requests were executed successfully.

- **Screenshots of Results:** Images of interactions with the contract and API requests are attached.

## 9.4 Discussion

### Challenges Faced

- Setting up the local server and deploying the smart contract took time due to configuration errors.
- Metamask required manual configuration of the test network, which led to minor delays.

### Security Considerations

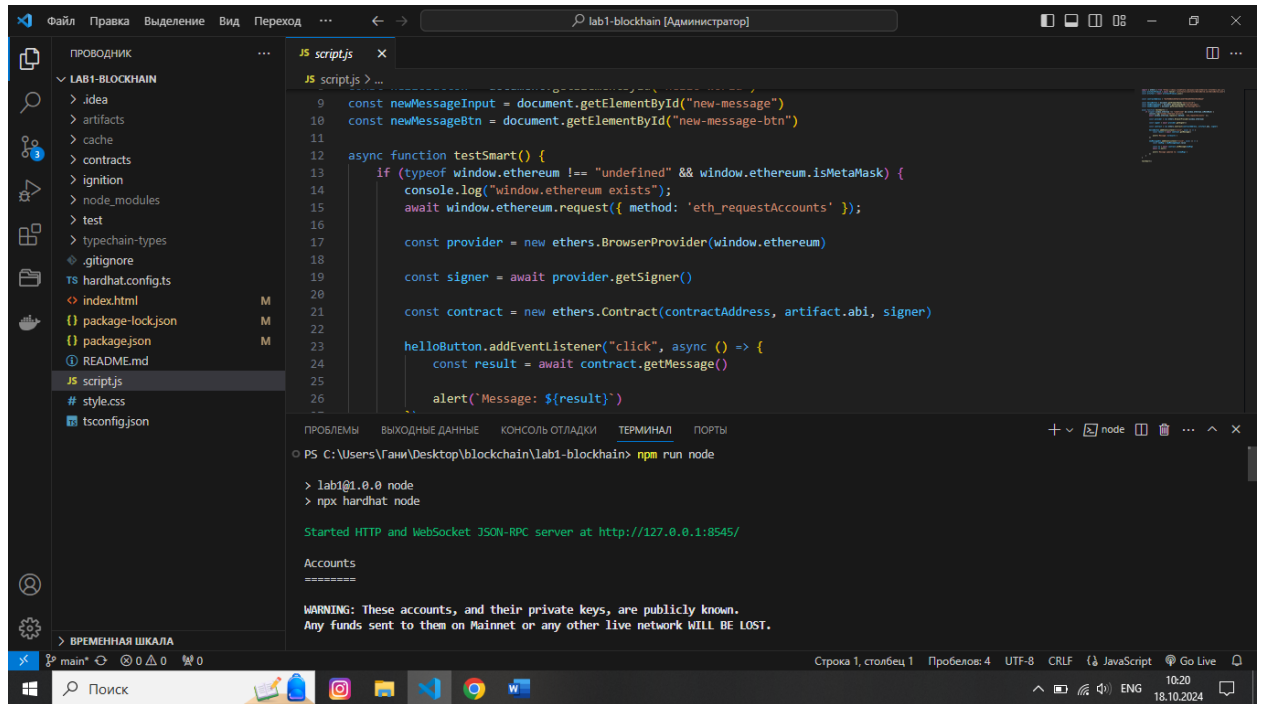
- Basic security measures were implemented: input validation and access restrictions to critical contract functions.

**9.5 Conclusion** During the lab work, a platform for interacting with the blockchain contract was successfully created, and APIs for interaction with it were implemented. Testing demonstrated the correctness of the contracts and APIs.

## 9.6 References

- Solidity Documentation
- Metamask Setup Guides
- Instructions for Working with Node.js and APIs

## Running node server on localhost



The screenshot shows the Visual Studio Code editor with the 'lab1-blockchain' project open. The file explorer on the left shows the project structure, including 'script.js'. The main editor displays the content of 'script.js', which contains JavaScript code for interacting with a smart contract. The terminal at the bottom shows the command 'npm run node' being executed, resulting in the output: 'lab1@1.0.0 node', 'npm: hardhat node', and 'Started HTTP and WebSocket JSON-RPC server at http://127.0.0.1:8545/'. A warning message is also displayed: '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.'

```
const newMessageInput = document.getElementById("new-message")
const newMessageBtn = document.getElementById("new-message-btn")

async function testSmart() {
  if (typeof window.ethereum !== "undefined" && window.ethereum.isMetaMask) {
    console.log("window.ethereum exists");
    await window.ethereum.request({ method: 'eth_requestAccounts' });

    const provider = new ethers.BrowserProvider(window.ethereum)

    const signer = await provider.getSigner()

    const contract = new ethers.Contract(contractAddress, artifact.abi, signer)

    helloButton.addEventListener("click", async () => {
      const result = await contract.getMessage()

      alert(`Message: ${result}`)
    })
  }
}
```

```
PS C:\Users\Гани\Desktop\blockchain\lab1-blockchain> npm run node

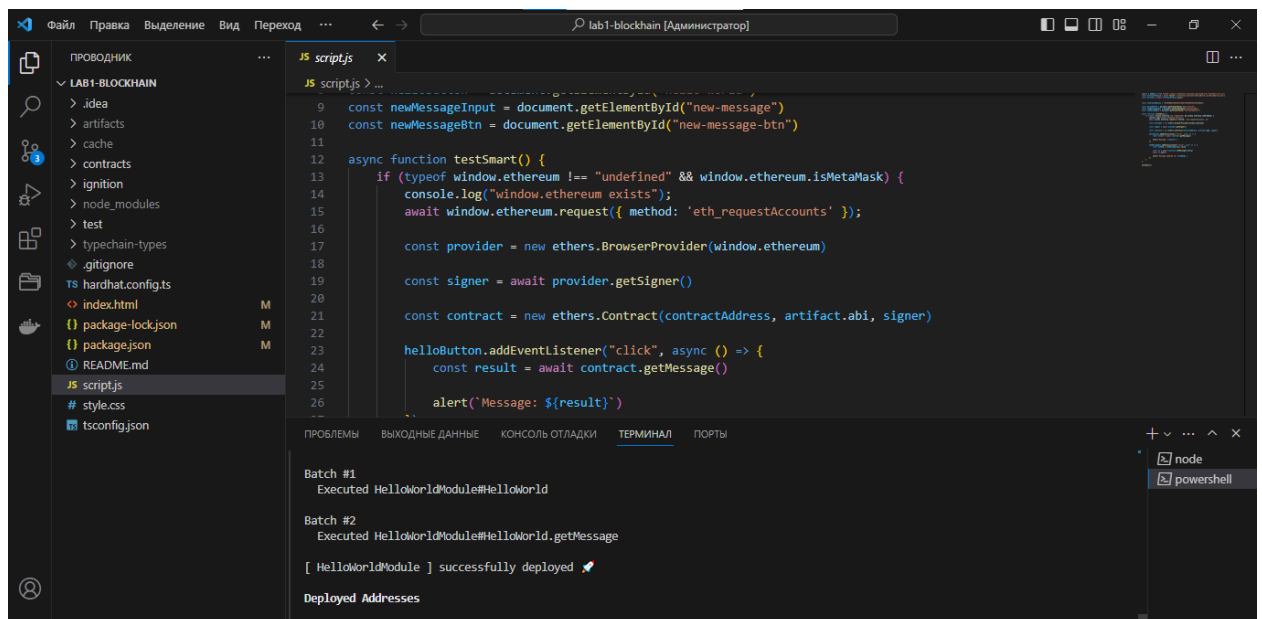
> lab1@1.0.0 node
> npx hardhat node

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.
```

## Then deploying smart contracts for unit test



The screenshot shows the Visual Studio Code editor with the 'lab1-blockchain' project open. The file explorer on the left shows the project structure, including 'script.js'. The main editor displays the content of 'script.js', which contains JavaScript code for interacting with a smart contract. The terminal at the bottom shows the command 'npm run deploy' being executed, resulting in the output: 'Batch #1', 'Executed HelloWorldModule#HelloWorld', 'Batch #2', 'Executed HelloWorldModule#HelloWorld.getMessage', '[ HelloWorldModule ] successfully deployed', and 'Deployed Addresses'.

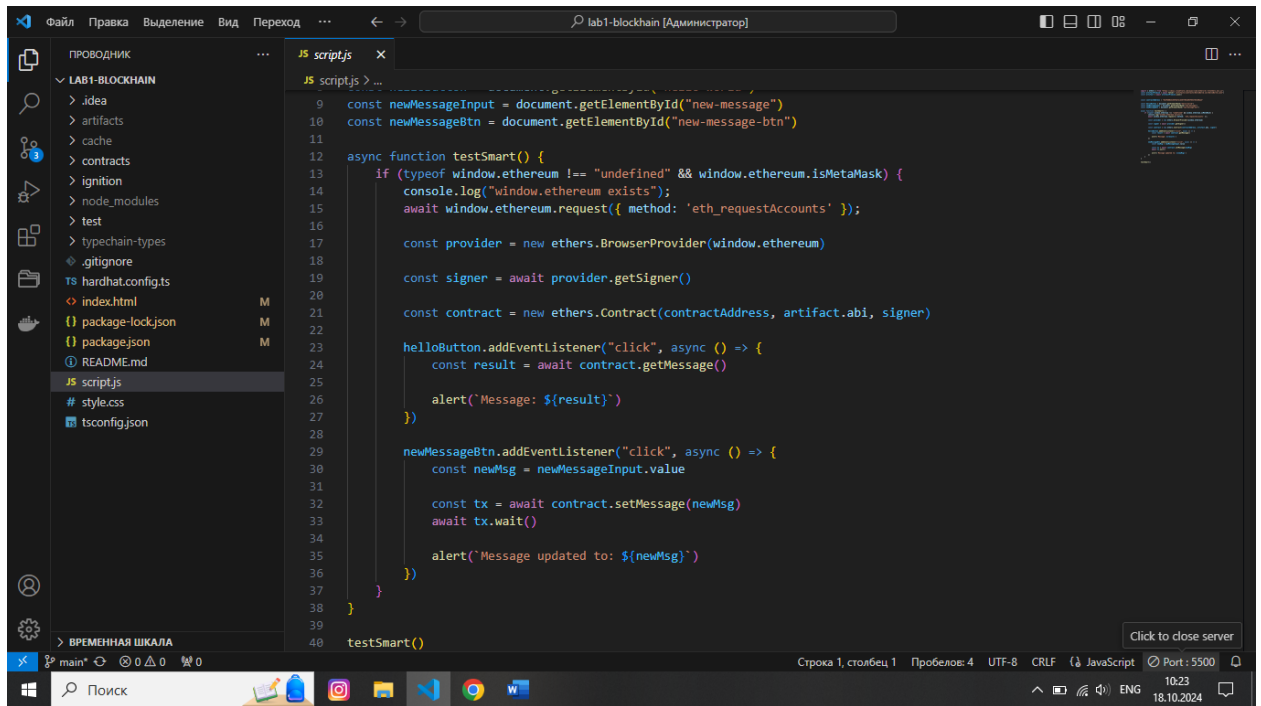
```
Batch #1
  Executed HelloWorldModule#HelloWorld

Batch #2
  Executed HelloWorldModule#HelloWorld.getMessage

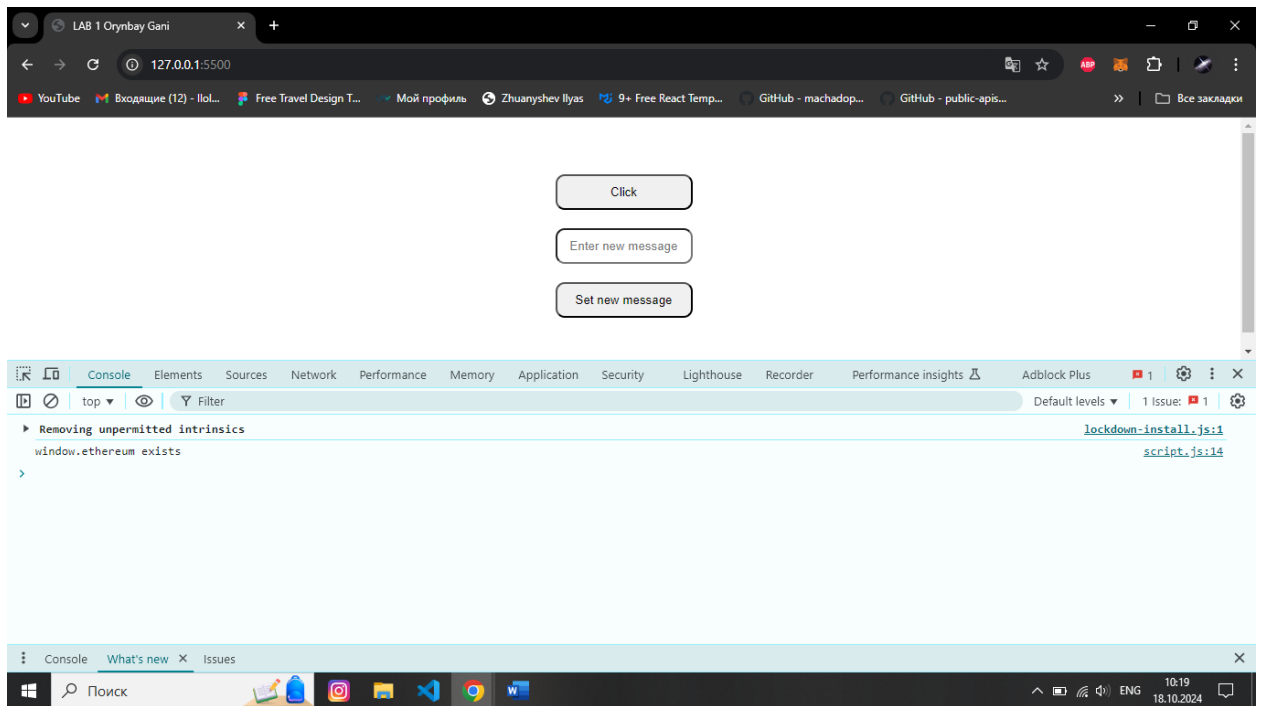
[ HelloWorldModule ] successfully deployed

Deployed Addresses
```

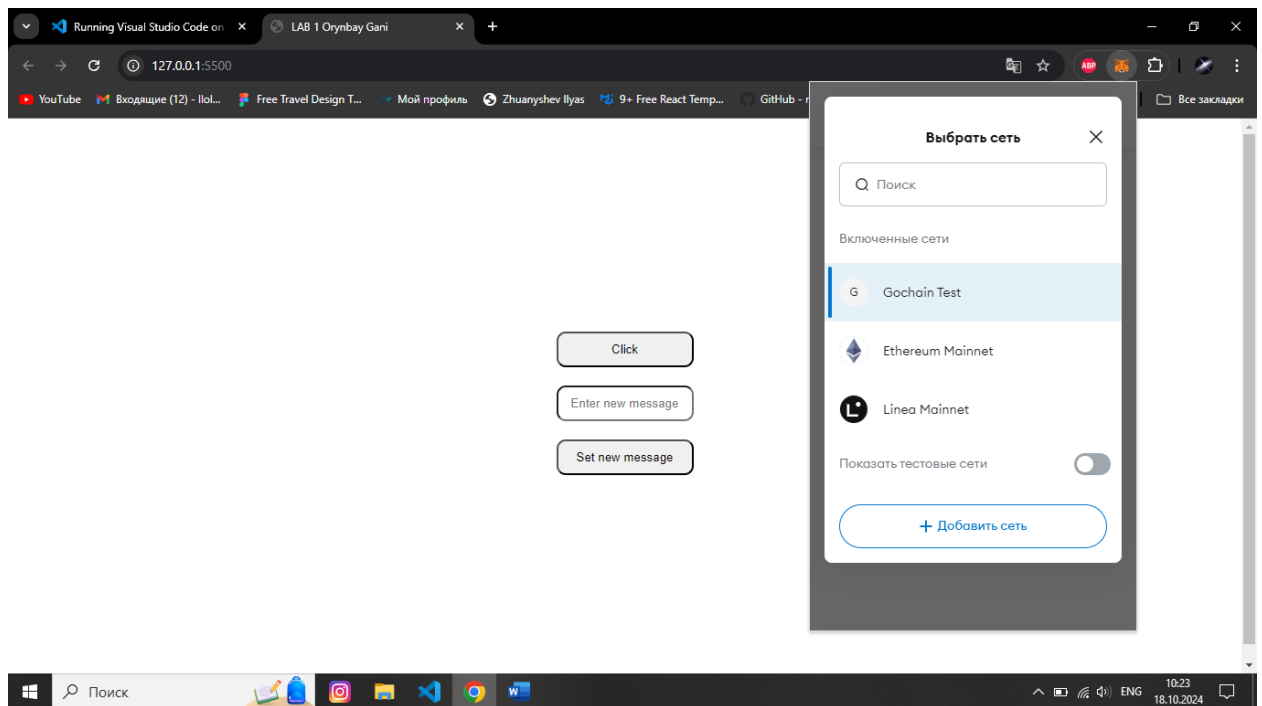
## After that turning on local server



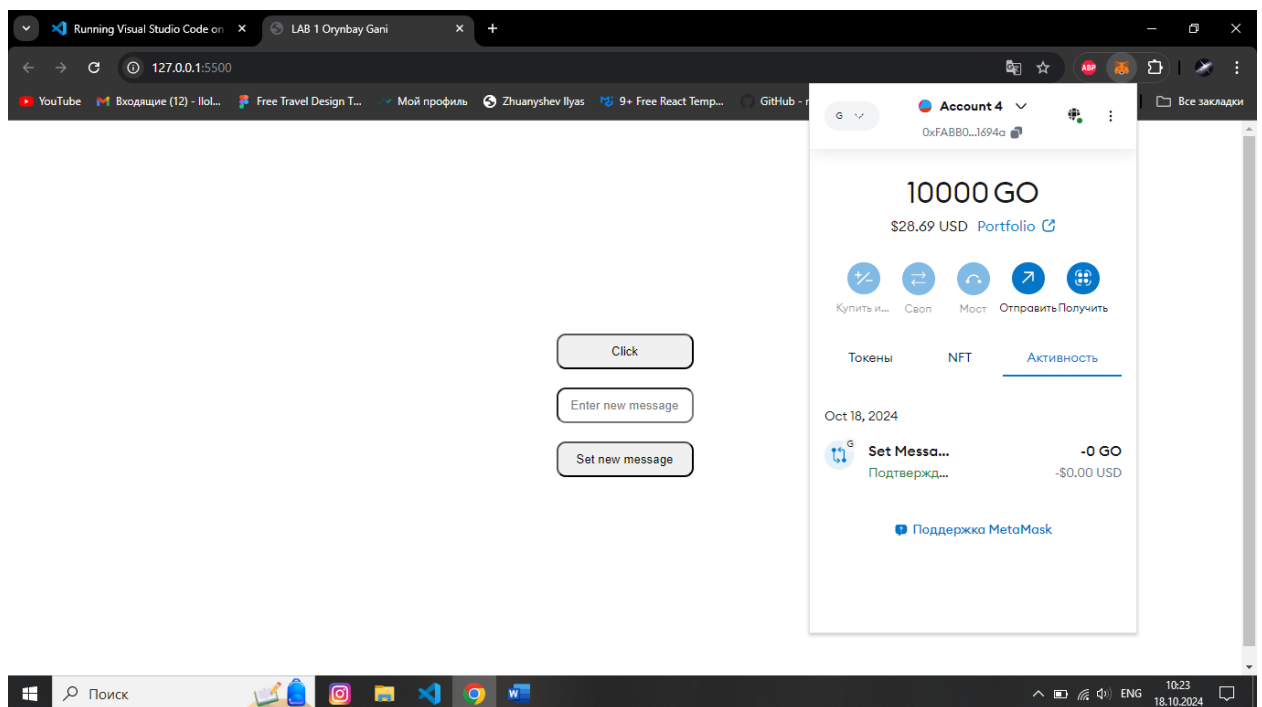
## interface



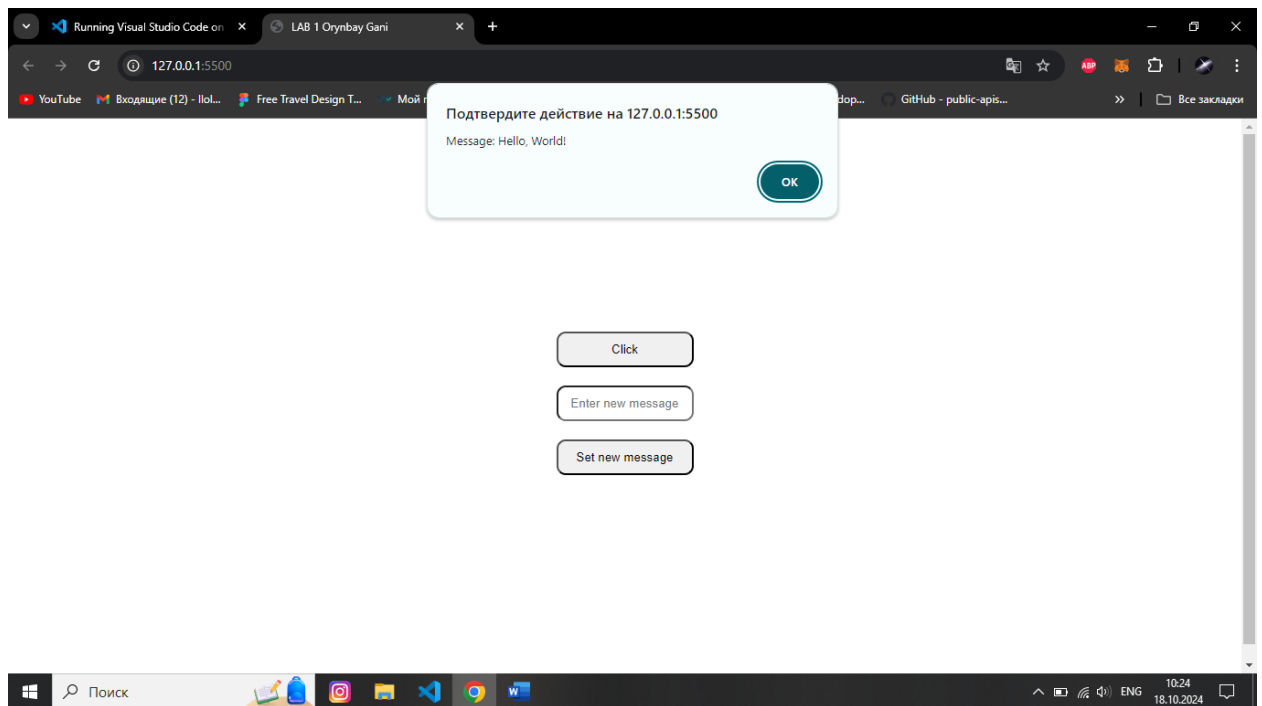
## Metamask test server



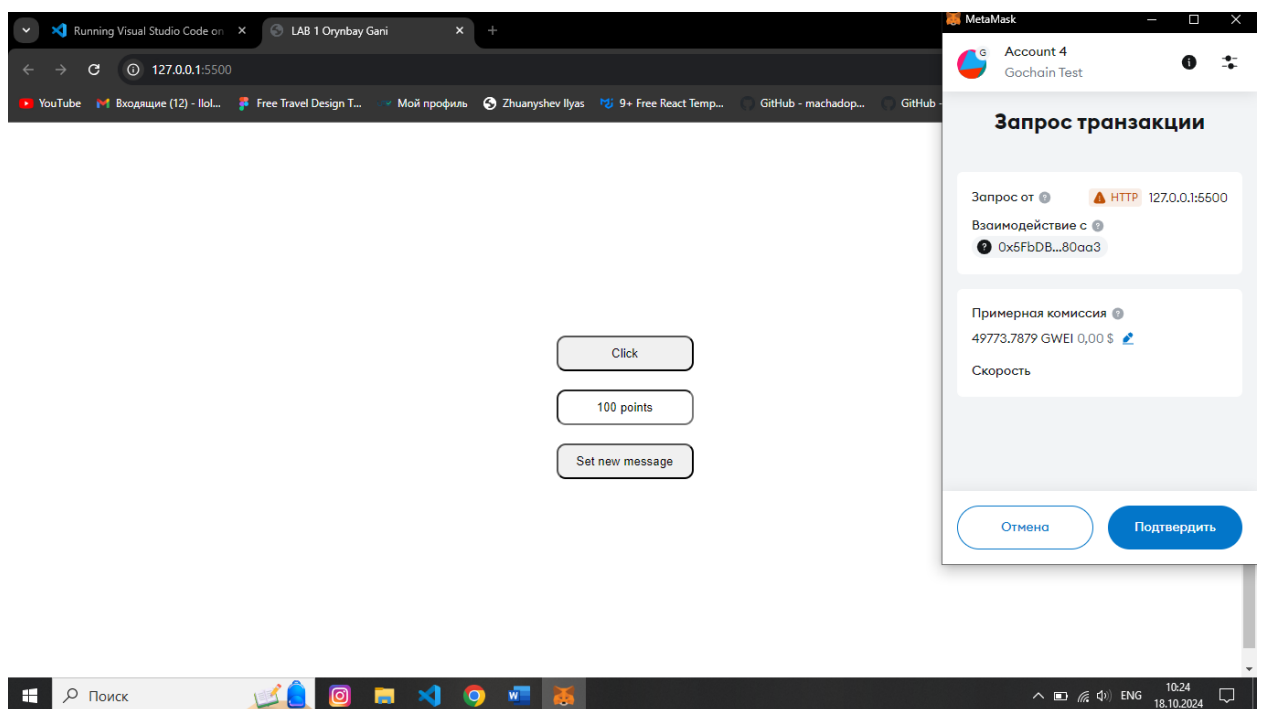
## Tokens



## Hello World on solidity



## Setting new alert message



## Getting new message

