POZNAN UNIVERSITY OF TECHNOLOGY

Simple Blockchain Implementation

Eng. Mateusz Szuda (index 144379)
Eng. Mikita Pilinka (index 152153)

June 15, 2023

Objective:

- The objective of this project is to implement a basic blockchain structure that can be used to store and verify transactions. The blockchain should include the necessary components such as blocks, transactions, and cryptographic mechanisms to ensure data integrity and security.

Contents

1 Introduction

Blockchain technology has gained significant attention in recent years due to its potential to revolutionize various industries. This document presents a brief overview of a blockchain implementation using Node.js. The implementation includes the creation of blocks, mining, transaction handling, and integration with React as the front end and Express framework on the backend. Additionally, the implementation incorporates a Proof-of-Work consensus mechanism.

2 Implementation Details

The blockchain implementation uses Node.js as the runtime environment, React as the front-end framework, and Express as the backend framework. The provided code snippets showcase the integration of the blockchain functionality with the Express router.

2.1 Front-end Integration with React

The front-end integration involves building user interfaces using React components to interact with the blockchain. These components can include forms for user registration, authentication, and transaction creation. The React components will communicate with the backend server via API calls to perform blockchain operations.

2.2 Back-end Integration with Express

The Express framework is used on the backend to handle HTTP requests and responses. The provided code snippets show how the blockchain functionality is integrated into the Express router. The router handles routes for user authentication, registration, transaction verification, and block creation.

2.3 Transaction Verification

The verify route in the Express router is responsible for verifying signed transactions. It retrieves the necessary transaction data, including the sender, recipient, amount, and signature. The implementation uses the crypto module to verify the transaction's signature using the sender's public key. If the verification is successful, the transaction is considered valid.

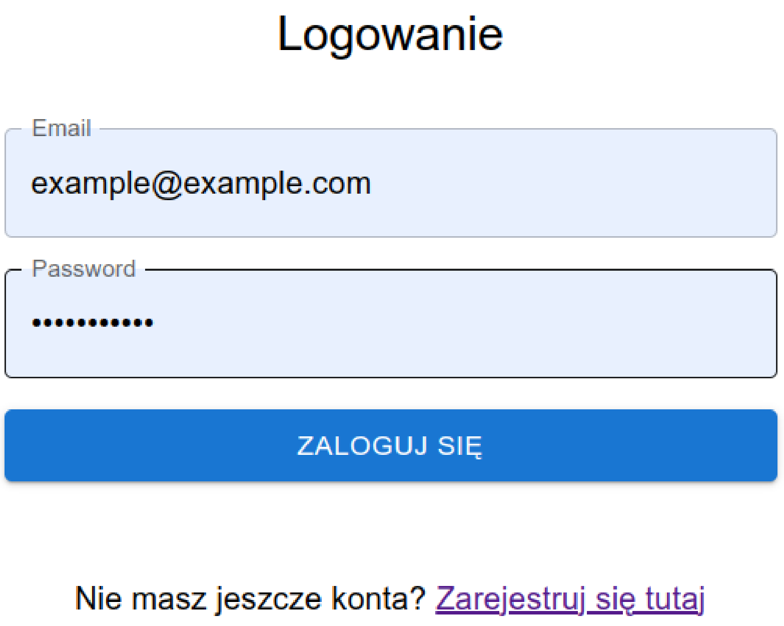
2.4 Other Components

The implementation includes other components such as the BlockChain class, Block class, and Transaction class. These components handle the core functionality of the blockchain, including block creation, mining, and transaction handling.

3 App walkthrough

3.1 Login screen

At first we need to provide credentials in order to login into our account. We also have a possibility to register a new account


Logowanie

Email: example@example.com

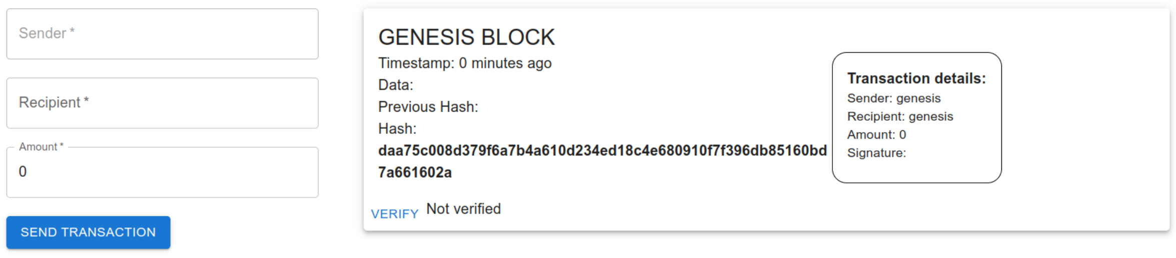
Password: •••••••••••

Nie masz jeszcze konta? Zarejestruj się tutaj

Figure 1: Login screen

3.2 Genesis block

Initial screen after logging in or registering is presented as follows.



Amount: 0

GENESIS BLOCK
Timestamp: 0 minutes ago
Data:
Previous Hash:
Hash:
daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

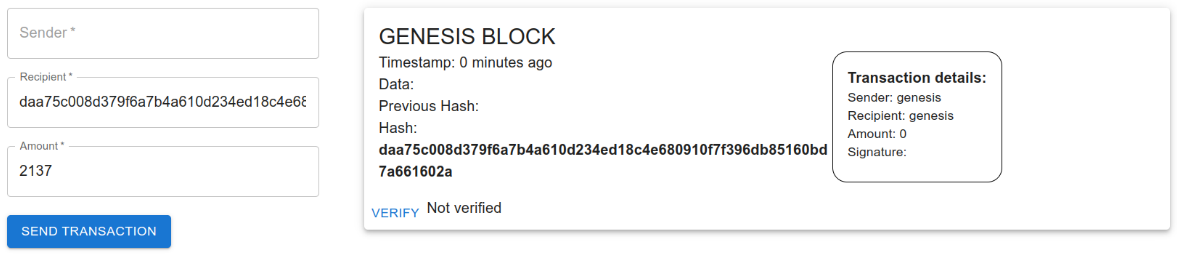
Not verified

Transaction details:
Sender: genesis
Recipient: genesis
Amount: 0
Signature:

Figure 2: Genesis block used for the start of the blockchain of transactions

3.3 Adding new transactions

3.3.1 Filling out the form



Recipient *: daa75c008d379f6a7b4a610d234ed18c4e6ε

Amount *: 2137

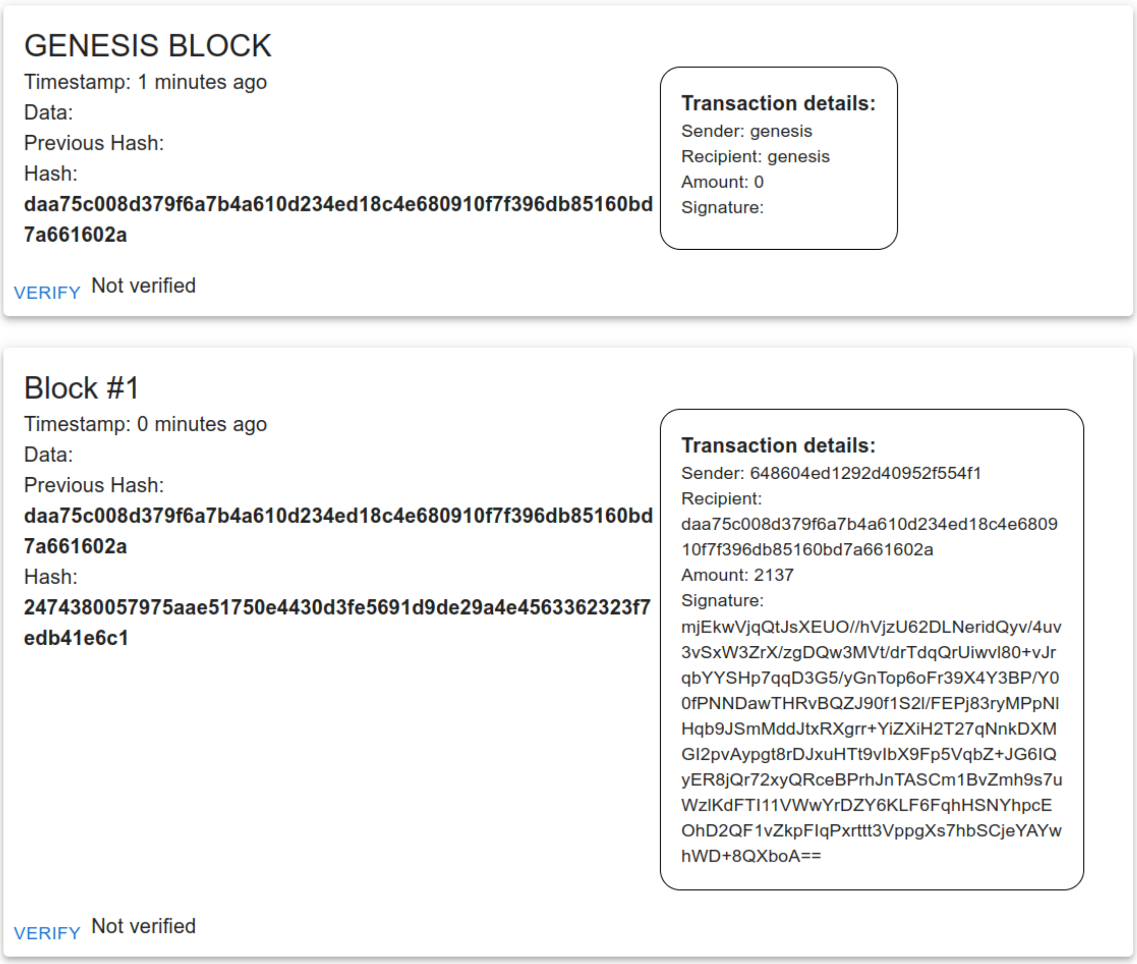
GENESIS BLOCK
Timestamp: 0 minutes ago
Data:
Previous Hash:
Hash:
daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Not verified

Transaction details:
Sender: genesis
Recipient: genesis
Amount: 0
Signature:

Figure 3: Filling out the form of creating new transaction

3.3.2 Transaction added



GENESIS BLOCK

Timestamp: 1 minutes ago

Data:

Previous Hash:

Hash:

daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Transaction details:

Sender: genesis

Recipient: genesis

Amount: 0

Signature:

VERIFY Not verified

Block #1

Timestamp: 0 minutes ago

Data:

Previous Hash:

daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Hash:

2474380057975aae51750e4430d3fe5691d9de29a4e4563362323f7edb41e6c1

Transaction details:

Sender: 648604ed1292d40952f554f1

Recipient:

daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Amount: 2137

Signature:

mjEkwVjqQtJsXEUO//hVjzU62DLNeridQyv/4uv3vSxW3ZrX/zgDQw3MVt/drTdqQrUiwvl80+vJrqbYYSHp7qqD3G5/yGnTop6oFr39X4Y3BP/Y00fPNNDawTHRvBQZJ90f1S2l/FEPj83ryMPpNlHqb9JSmMddJtxRXgrr+YiZXiH2T27qNnkDXMGl2pvAypgt8rDJxuHTt9vIbX9Fp5VqbZ+JG6IQyER8jQr72xyQRceBPrhJnTASCm1BvZmh9s7uWzlKdFTI11VWwYrDZY6KLF6FqhHSNYhpcEOhD2QF1vZkpFIqPxrttt3VppgXs7hbSCjeYAYwhWD+8QXboA==

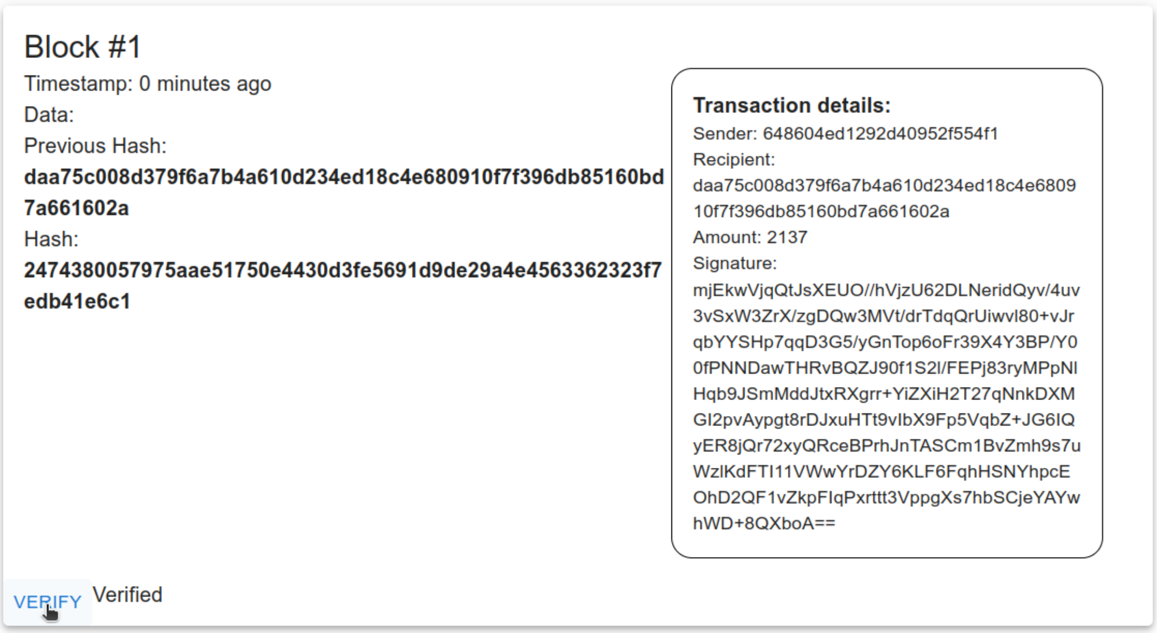
VERIFY Not verified

Figure 4: Caption

3.4 Verification

3.4.1 Correctly verified

If verified transaction is coherent, then the verification that was carried out on the backend side is successfull.



Block #1

Timestamp: 0 minutes ago

Data:

Previous Hash:

daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Hash:

2474380057975aae51750e4430d3fe5691d9de29a4e4563362323f7edb41e6c1

Transaction details:

Sender: 648604ed1292d40952f554f1

Recipient:
daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Amount: 2137

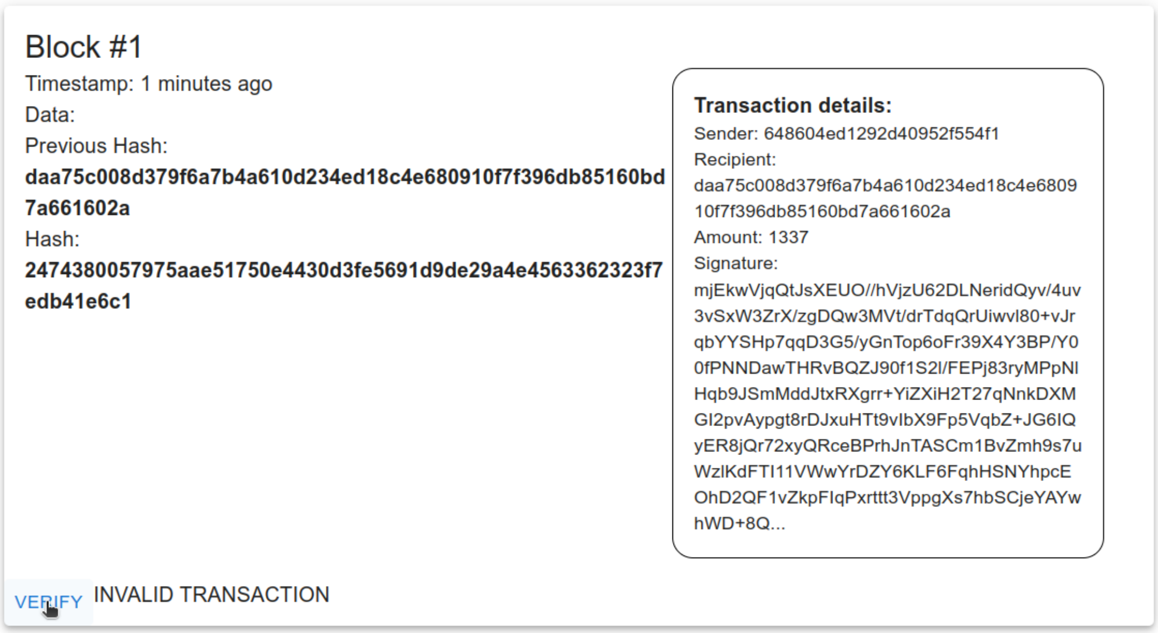
Signature:
mjEkwVjqQtJsXEUO//hVjzU62DLNeridQyv/4uv3vSxW3ZrX/zgDQw3MVt/drTdqQrUiwvl80+vJrqbYYSHp7qqD3G5/yGnTop6oFr39X4Y3BP/Y00fPNNDawTHRvBQZJ90f1S2l/FEPj83ryMPpNlHqb9JSmMddJtxRXgrr+YiZXiH2T27qNnkDXMGl2pvAypgt8rDJxuHTt9vIbX9Fp5VqbZ+JG6IQyER8jQr72xyQRceBPrhJnTASCm1BvZmh9s7uWzlKdFTI11VWwYrDZY6KLF6FqhHSNYhpcEOhD2QF1vZkpFIqPxrttt3VppgXs7hbSCjeYAYwhWD+8QXboA==

Figure 5: Transaction verified successfully

3.4.2 Verification failure

If the verification failed, the user will be prompted with a proper message.

- On the figure 6 there is a visible change of transaction amount, which was also verified by a backend after user verification.



Block #1

Timestamp: 1 minutes ago

Data:

Previous Hash:

daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Hash:

2474380057975aae51750e4430d3fe5691d9de29a4e4563362323f7edb41e6c1

Transaction details:

Sender: 648604ed1292d40952f554f1

Recipient:

daa75c008d379f6a7b4a610d234ed18c4e680910f7f396db85160bd7a661602a

Amount: 1337

Signature:

mjEkwVjqQtJsXEUO//hVjzU62DLNeridQyv/4uv3vSxW3ZrX/zgDQw3MVt/drTdqQrUiwvl80+vJrqbYYSHp7qqD3G5/yGnTop6oFr39X4Y3BP/Y00fPNNDawTHRvBQZJ90f1S2l/FEPj83ryMPpNlHqb9JSmMddJtxRXgrr+YiZXiH2T27qNnkDXMGl2pvAypgt8rDJxuHTt9vIbX9Fp5VqbZ+JG6IQyER8jQr72xyQRceBPrhJnTASCm1BvZmh9s7uWzlKdFTI11VWwYrDZY6KLF6FqhHSNYhpcEOhD2QF1vZkpFIqPxrttt3VppgXs7hbSCjeYAYwhWD+8Q...

INVALID TRANSACTION

Figure 6: Verification failure

4 Conclusion

This document provided a brief overview of a blockchain implementation using Node.js, React, and Express. The implementation showcases the integration of the blockchain functionality with the front-end and back-end frameworks. It includes transaction verification and provides a foundation for implementing Proof-of-Work consensus. Developers can build upon this implementation to create decentralized applications and explore the potential of blockchain technology.