**RIGA TECHNICAL UNIVERSITY**

Faculty of Computer Science, Information Technology and Energy

Institute of Applied Computer Systems

**Alina Verkholomova**

Academic Master Study Programme “Business Informatics”

**Assignment for blockchain development**

**Assignment on Introduction to Blockchain Technology**

Scientific adviser Arnis Staško

RIGA 2024

# TASK DESCRIPTION

1. Code and test a blockchain:
   1. Write a code
   2. Add advanced blockchain features
   3. Create a blockchain with a minimum of 10 blocks. Visualize the result.
   4. Try to make an unauthorized change
      1. Change a block in the middle of the blockchain
      2. Change the last block of the blockchain
      3. Add another block at the end of the blockchain
      4. Is it possible to detect a fake? How?
2. Upload the code you have created, tests, samples and description summarized in a document.

# Summary

A code was developed to build a blockchain chain with 10 blocks and advanced features (proof-of-work).

const SHA256 = require("crypto-js/sha256");

class Block {

constructor(index, timestamp, data, previousHash = '') {

this.index = index;

this.previousHash = previousHash;

this.timestamp = timestamp;

this.data = data;

this.hash = this.calculateHash();

this.nonce = 0;

}

calculateHash() {

return SHA256(this.index + this.previousHash + this.timestamp + this.nonce + JSON.stringify(this.data)).toString();

}

mineBlock(difficulty) {

while (this.hash.substring(0, difficulty) !== Array(difficulty + 1).join("0")){

this.nonce++;

this.hash = this.calculateHash();

}

console.log("BLOCK MINED: " + this.hash);

}

}

class Blockchain {

constructor() {

this.chain = [this.createGenesisBlock()];

this.difficulty = 2;

}

createGenesisBlock() {

return new Block(0, "09/03/2025", "Genesis block", "0");

}

getLatestBlock(){

return this.chain[this.chain.length - 1];

}

addBlock(newBlock) {

newBlock.previousHash = this.getLatestBlock().hash;

newBlock.mineBlock(this.difficulty);

this.chain.push(newBlock);

}

isChainValid() {

for (let i = 1; i < this.chain.length; i++) {

const currentBlock = this.chain[i];

const previousBlock = this.chain[i - 1];

if (currentBlock.hash !== currentBlock.calculateHash()) {

return false;

}

if (currentBlock.previousHash !== previousBlock.hash) {

return false;

}

}

return true;

}

}

console.log("Hello, Blockchain!");

let myBlockchain = new Blockchain();

for (let i = 1; i <= 10; i++) {

myBlockchain.addBlock(new Block(i, "09/03/2025", { amount: i \* 10 }));

}

console.log(JSON.stringify(myBlockchain, null, 4));

console.log("Blockchain validity: ", myBlockchain.isChainValid());

In this code, blocks are created and added to myBlokchain by using for loop. Then, myBlockhain is visualized by using the console.log output command and JSON.stringify. Afterward, myBlockchain is validated.

The output:

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

Change a block in the middle of the blockchain:

console.log("\nChange a block under index 5: ");

myBlockchain.chain[4].data = { amount: 1000 };

console.log("Blockchain validity: ", myBlockchain.isChainValid());

The output:

A black background with white text

Description automatically generated

The validity of myBlockchain failed.

Change the last block of the blockchain:

console.log("\nChange the last block: ");

myBlockchain.chain[10].data = { amount: 9999 };

console.log("Blockchain validity: ", myBlockchain.isChainValid());

The output:

A black background with white text

Description automatically generated

The validity of myBlockchain failed.

Add another block at the end of the blockchain:

console.log("\nAdding new block: ");

myBlockchain.addBlock(new Block(11, "09/03/2025", { amount: 111 }));

console.log("Blockchain validity: ", myBlockchain.isChainValid());

The output:



The validity of myBlockchain also failed.

Since I modified some blocks’ data, it failed validation. The blockchain detected the fake with the function isChainValid(). There is an option to try to recalculate hash after changing the block’s data:

myBlockchain.chain[4].data = { amount: 1000 };

myBlockchain.chain[4].hash = myBlockchain.chain[4].calculateHash();

However, the previousHash of the block with index six would be outdated. And the validation of the myBlockchain would fail again:

A black background with white text

Description automatically generated

Therefore, the hash of all blocks should be recalculated to cover the modification, but it is a time- and energy-consuming option, which makes blockchain a reliable and secure technology.

Unit tests:

const { Block, Blockchain } = require("./blockchain");

const SHA256 = require("crypto-js/sha256");

describe('Blockchain', () => {

let blockchain;

beforeEach(() => {

blockchain = new Blockchain();

});

test('should add new blocks correctly', () => {

blockchain.addBlock(new Block(1, "09/03/2025", { amount: 10 }));

blockchain.addBlock(new Block(2, "09/03/2025", { amount: 20 }));

blockchain.addBlock(new Block(3, "09/03/2025", { amount: 30 }));

expect(blockchain.chain.length).toBe(4);

expect(blockchain.chain[1].previousHash).toBe(blockchain.chain[0].hash);

expect(blockchain.chain[2].previousHash).toBe(blockchain.chain[1].hash);

expect(blockchain.chain[3].previousHash).toBe(blockchain.chain[2].hash);

});

test('should validate blockchain integrity correctly', () => {

blockchain.addBlock(new Block(1, "09/03/2025", { amount: 10 }));

expect(blockchain.isChainValid()).toBe(true);

blockchain.chain[1].data = { amount: 100 };

expect(blockchain.isChainValid()).toBe(false);

});

test('should calculate hash correctly', () => {

const block = new Block(1, "09/03/2025", { amount: 50 });

const calculatedHash = block.calculateHash();

expect(calculatedHash).toBe(SHA256(block.index + block.previousHash + block.timestamp + block.nonce + JSON.stringify(block.data)).toString());

});

test('should mine block correctly', () => {

const block = new Block(1, "09/03/2025", { amount: 50 });

block.mineBlock(blockchain.difficulty);

expect(block.hash.substring(0, blockchain.difficulty)).toBe(Array(blockchain.difficulty + 1).join("0"));

});

test('should add a new block and validate chain correctly', () => {

blockchain.addBlock(new Block(4, "09/03/2025", { amount: 40 }));

expect(blockchain.isChainValid()).toBe(true);

});

});

The output:

A screenshot of a computer

Description automatically generated

Link to the GitHub repository:

<https://github.com/AlinaVerkholomova/blockhain_dev_assignment>