Blockchain implementation report

Mieszko Gąska 144432
Adrian Głogowski 144232

June 14, 2023

Contents

1 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.

2 Used technologies

- PostgreSQL - database
- Java (Spring) - server
- React - user interface

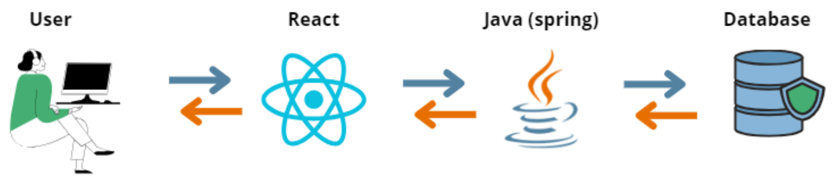


Figure 1: Project communication workflow

3 Database

In this section, we present key actors/objects in our project and their properties, whose purpose will be described in more detail later in the report.

3.1 Users

- **ID**
- **password**
- **username**

ID of user distinguishes him from others, we store this information in UUID type which ensure value to be unique and written as a sequence of lower-case hexadecimal digits. Password is secure stored as hash value, for what we used **PasswordEncoder** method from Spring framework.

3.2 Block

- **index**
- **hash**
- **nonce**
- **previous hash**
- **timestamp**

Index and hash are like ID, where index is used to enumerate each block, which it is used later to distinguish block in transaction, hash is like a true pointer to this block in blockchain. Nonce property is a number which counts how many attempts block needed to find the correct value hash value, in our project we take that hash value have to begin with 16 zeroes, while it is calculating by method:

$$index + timestamp + data + previousHash + nonce \tag{1}$$

Where data is a list of transactions.
Previous hash is a hash value of the previous block, if there is no previous block in the transaction, then we set this value to '0'.
And the last - timestamp is a time when the block starts to mining and transaction is already added to it.

3.3 Transaction

- **ID**
- **recipient**
- **sender**
- **value**
- **block index**

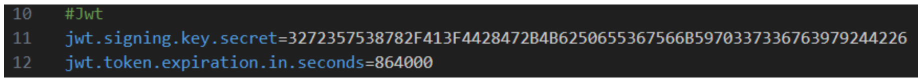
ID distinguishes transaction. Recipient is UUID of the person who was commissioned to execute this transaction and who will get rewarded by the amount of value property.
Sender is UUID of the person who submitted the transaction.
Block index is the identity of the block that executes the transaction.

4 User authorization

Our application have login/registration functionality, by that only authorized users can join to the blockchain. Process of authorization can be described in a several steps:

a. User creates account using registration panel

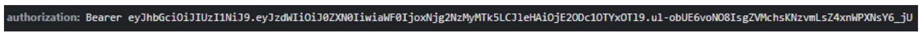
b. After successful registration user login to the application, during that server checks his login credentials and if everything is correct, returns **Json Web Token (JWT)**. JWT is signed by a secret value contained in the server configuration settings. For better security we also set the expiration time of the JWT token.



```
#Jwt
jwt.signing.key.secret=3272357538782F413F4428472B4B6250655367566B5970337336763979244226
jwt.token.expiration.in.seconds=864000
```

Figure 2: JWT configuration

Whenever the user wants to access a protected route or resource, the user agent should send the JWT, typically in the Authorization header using the Bearer schema, which should look like the following:



```
authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ0ZXN0IiwiaWF0IjoxNjg2NzMyMTk5LCJleHAiOjE2ODc1OTYxOT19.ul-obUE6voNO8IsgZVMchsKNzvmLsZ4xnWPXNsY6_jU
```

Figure 3: Token in header

5 Blockchain implementation logic

During block mining, at the beginning, we are searching for the transaction with the highest value. In case there is no transaction at all, Block is mining every 10 second, searching for a transaction that can be picked up. When we choose the transaction, we set up the timestamp, data and previous hash properties of block, which we already described more specific in subsection **Block** of Database section. Next we go to the consensus mechanism.

5.1 consensus mechanism

For our implementation, we chose **proof-of-work** mechanism, where the difficulty is to start hash value of each block with 16 zeroes. We also described it in the **Block** subsection. after obtaining the right value hash value, we save it and the number of iteration which was needed to calculate it (nonce property). And that prepared block is joined to the blockchain.
Of course, we also added validation methods for creatable object i.e. blocks and transactions which occurs before adding new block or transaction. For example, that blocks hashes for sure are unique and can't be found the same in the blockchain.