УДК 004

Создание блокчейн с использованием Java

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Глаголев Владимир Александрович
Приамурский государственный университет имени Шолом-Алейхема
К.г.н., доцент, доцент кафедры информационных систем, математики и правовой информатики

Аннотация
В данной статье будут рассмотрены возможности создания собственного базового блокчейн. Так же будет внедрена система проверки доказательства работы. Данная работа будет выполнена в среде разработки IntelijIdea с использованием языка программирования Java.
Ключевые слова: Java, Blockchein, Bitcoin

Building a blockchain using Java

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University Student

Glagolev Vladimir Aleksandrovich
Sholom-Aleichem Priamursky State University
Ph.D, Associate Professor, Associate Professor of the Department of Information Systems, Mathematics and Legal Informatics

Abstract
This article will consider the possibilities of creating your own basic blockchain. A proof-of-work verification system will also be introduced. This work will be done in the IntelijIdea development environment using the Java programming language.
Keywords: Java, Blockchein, Bitcoin

В настоящее время технологии развиваются очень быстро и потребность в информационной безопасности постоянно возрастает. В связи

с этим становится востребованной технология Блокчейн, позволяющая сохранять целостность информации. Кроме того, технология позволяет создать децентрализованную среду, в которой транзакции и данные происходят без какой-либо сторонней организации.

Цель работы – создать базовую модель технологии блокчейн, а также внедрить систему доказательства работы.

Исследованиями в данной теме занимались следующие авторы.

Б. Кумалаков и Я. Шакан предложили разработку децентрализованного студенческого приложения, который будет обрабатывать и хранить токены, которые будут представлять собой кредиты, которые студенты будут получать за прохождение определенных курсов [1]. В.П.Часовских, В.Г. Лабунец, М.П. Воронов описали технологию блокчейн как основу развития и повышения эффективности электронной системы ВУЗа [2]. С.Г. Наумов в своей работе представил технологию блокчейн, способы добытия криптовалюты [3]. П.А. Андреева, Р.А. Андреев провели обзор на разновидности технологий блокчейн и видах их применения [4]. Л.А. Симанович, расписала возможности использования блокчейн для электронных торговых площадок [5].

Блокчейн — это просто цепочка блоков. Каждый блок в цепочке будет иметь свой собственный цифровой отпечаток, содержать цифровой отпечаток предыдущего блока и иметь некоторые данные.

Хэш – это цифровой отпечаток.

Каждый блок не просто содержит хэш предшествующего ему блока, но его собственный хэш частично вычисляется из предыдущего хэша. Если данные предыдущего блока изменены, хэш предыдущего блока изменится, что, в свою очередь, повлияет на все хэши последующих блоков. Вычисление и сравнение хэшей позволяет увидеть, является ли блокчейн недействительным. Следовательно, изменение любых данных в списке блоков изменит подпись и разорвет цепочку.

Для начала создадим конструктор, который будет создавать блоки (рис.1).

```java
import java.util.Date;

public class Block {

        public String hash;
        public String previousHash;
        private String data;
        private long timeStamp;



        public Block(String data,String previousHash ) {
                this.data = data;
                this.previousHash = previousHash;
                this.timeStamp = new Date().getTime();

        }
```

Рисунок 1 – Класс для создания блоков

Базовый блок содержит «hash» – цифровую подпись. Переменная «previousHash» необходима для хранения хэша предыдущего блока и «data» – это данных блока.

Далее необходимо реализовать способ для генерирования цифрового отпечатка.

Существует много криптографических алгоритмов, которые можно выбрать, однако SHA256 отлично подходит для этого примера.

Импортируем библиотеку «java.security.MessageDigest» для получения доступа к алгоритму SHA256.

Далее создадим удобный вспомогательный метод (рис. 2).

```
import java.security.MessageDigest;

import com.google.gson.GsonBuilder;

public class StringUtil {



        public static String applySha256(String input){

                try {
                        MessageDigest digest = MessageDigest.getInstance("SHA-256");



                        byte[] hash = digest.digest(input.getBytes("UTF-8"));

                        StringBuffer hexString = new StringBuffer();
                        for (int i = 0; i < hash.length; i++) {
                                String hex = Integer.toHexString(0xff & hash[i]);
                                if(hex.length() == 1) hexString.append('0');
                                hexString.append(hex);
                        }
                        return hexString.toString();
                }
                catch(Exception e) {
                        throw new RuntimeException(e);
                }
        }
```

Рисунок 2 – Создание вспомогательного класса

Данный метод принимает строку и применяет к ней алгоритм SHA256, а также возвращает сгенерированную подпись в виде строки.

Теперь воспользуемся методом «applySha256» в новом методе класса «Block» для вычисления хеша. Необходимо вычислить хэш из всех частей блока, которые не должны подделываться. В данном примере создаем подпись на основе трех вводных: data, timeStamp, previousHash (рис. 3).

```
public String calculateHash() {
        String calculatedhash = StringUtil.applySha256(
                        previousHash +
                        Long.toString(timeStamp) +
                        Integer.toString(nonce) +
                        data
                        );
        return calculatedhash;
}
```

Рисунок 3 – Вычисление Хэша

Так же добавим данный метод в конструктор Block (рис. 4).

```
import java.util.Date;

public class Block {

        public String hash;
        public String previousHash;
        private String data;
        private long timeStamp;
        private int nonce;


        public Block(String data,String previousHash ) {
                this.data = data;
                this.previousHash = previousHash;
                this.timeStamp = new Date().getTime();

                this.hash = calculateHash();
        }
```

Рисунок 4 – Конструктор Block

Теперь попробуем данную генерацию блоков в действительности, создадим первые три блока (рис.5).

```
public static void main(String[] args) {

        Block genesisBlock = new Block("Hi im the first block", "0");
        System.out.println("Hash for block 1 : " + genesisBlock.hash);

        Block secondBlock = new Block("Yo im the second block",genesisBlock.hash);
        System.out.println("Hash for block 2 : " + secondBlock.hash);

        Block thirdBlock = new Block("Hey im the third block",secondBlock.hash);
        System.out.println("Hash for block 3 : " + thirdBlock.hash);

}
```

Рисунок 5 – Создание блоков

После запуска программы, вывод будет следующим (рис. 7).

```
Hash for block 1 : 040c503d5077ee753e8c17ce18b627ef1286c806507fd52baec4610924ed086b
Hash for block 2 : d5c73e535cbcfa6c10df7fba28541cd61109a42403fcd7df5134e1db7c325fd0
Hash for block 3 : f7492355aaf5b0516aaab5307537cc93dc23b3a46d57a37af52baa4c09b062d7
```

Рисунок 7 – Вывод блоков

Каждый блок теперь имеет собственную цифровую подпись, основанную на его информации и подписи предыдущего блока.

В настоящее время это не очень похоже на цепочку блоков, поэтому сохраним блоки в «ArrayList», а также импортируем «gson», чтобы просмотреть его как «Json» (рис.8).

```
public static ArrayList<Block> blockchain = new ArrayList<Block>();

public static void main(String[] args) {

        blockchain.add(new Block("Hi im the first block", "0"));
        blockchain.add(new Block("Yo im the second block",blockchain.get(blockchain.size()-1).hash));
        blockchain.add(new Block("Hey im the third block",blockchain.get(blockchain.size()-1).hash));

        String blockchainJson = new GsonBuilder().setPrettyPrinting().create().toJson(blockchain);
        System.out.println(blockchainJson);
}
```

Рисунок 8 – Создание блоков в ArrayList

Теперь необходим способ проверить целостность блокчейна. Для этого создадим логический метод «isChainValid()», который будет перебирать все блоки в цепочке и сравнивать хэши. Этот метод должен будет проверять, действительно ли хеш-переменная равна вычисленной хэш-функции, а хэш предыдущего блока равен предыдущей переменной «Hash» (рис. 9).

```
public static Boolean isChainValid() {
        Block currentBlock;
        Block previousBlock;


        for(int i=1; i < blockchain.size(); i++) {
                currentBlock = blockchain.get(i);
                previousBlock = blockchain.get(i-1);

                if(!currentBlock.hash.equals(currentBlock.calculateHash()) ){
                        System.out.println("Current Hashes not equal");
                        return false;
                }

                if(!previousBlock.hash.equals(currentBlock.previousHash) ) {
                        System.out.println("Previous Hashes not equal");
                        return false;
                }
        }
        return true;
}
```

Рисунок 9 – Функция проверки

Любое изменение блоков блокчейна приведет к тому, что этот метод вернет «false».

В сети биткойн узлы делятся своими цепочками блоков, и сеть принимает самую длинную действующую цепочку. Следовательно, злоумышленнику потребуется больше вычислительной мощности, чем остальным партнерам вместе взятым, так как система «hashcash proof of work» проверяет данную цепочку на целостность, а для создания цепочки большего размера требуется значительное время и вычислительная мощность.

В данной статье были рассмотрены возможность создания технологии блокчейн, а также создан метод для проверки действительности.

Библиографический список

1. Кумалаков Б. и Шакан Я. Блокчейн в образовании: как управлять студенческим зачетом вуза через блокчейн? // Вестник алматинского университета энергетики и связи 2020. №2(49). С. 128-133.
2. Часовских В.П., Лабунец В.Г., Воронов М.П. Технология "блокчейн" (blockchain) в образовании вузов и цифровой экономике // Эко-потенциал 2018. №2(18). С. 99 - 105.
3. Наумов С.Г. Блокчейн, криптовалюта и майнинг // Евразийская адвокатура 2021. №2(51). С. 32.
4. Андреева П.А., Андреев Р.А. Обзор технологии блокчейн: виды блокчейна и их применение // Интеллектуальные системы в производстве 2019. №1. С. 11-14.
5. Симанович Л.А. Возможности использования технологии блокчейн (blockchain) для электронной торговой площадки // Национальная безопасность и стратегическое планирование 2019. №1(21). С. 134-140.