

UNIVERSIDADE DO ESTADO DO AMAZONAS
ESCOLA SUPERIOR DE TECNOLOGIA
CURSO DE ENGENHARIA DA COMPUTAÇÃO

APLICAÇÃO BLOCKCHAIN EDUCACIONAL

Relatório Técnico

Tópicos Especiais em Computação IV

Manaus
2025

Sumário

1	Introdução	2
1.1	O que é o Projeto	2
1.2	Objetivos do Relatório	2
2	Explicação dos Conceitos de Bloco, Hash e Encadeamento	2
2.1	O que é um Bloco	2
2.2	O que é Hash	3
2.2.1	Como Funciona	3
2.3	O que é Encadeamento	4
2.3.1	Como Funciona o Encadeamento	4
3	Como Novos Blocos São Adicionados à Cadeia	4
3.1	Processo Passo a Passo	4
3.2	Proof of Work - A Mineração	5
3.2.1	Como Funciona	5
3.3	Criação e Adição do Bloco	6
3.4	Resumo do Processo Completo	6
4	Tecnologias Utilizadas	7
4.1	Backend (Python)	7
4.2	Frontend (React)	7
5	Vantagens do Modelo Desenvolvido	7
5.1	Transparência	7
5.2	Imutabilidade (Não Pode Ser Alterado)	7
5.3	Segurança	7
5.4	Sem Autoridade Central	7
5.5	Auditável	8
6	Limitações do Modelo Desenvolvido	8
6.1	Velocidade	8
6.2	Escala	8
6.3	Armazenamento	8
6.4	Rede	8
6.5	Segurança Avançada	8
6.6	Funcionalidades Limitadas	8
7	Conclusão	9
7.1	O que Aprendemos	9

7.2	Considerações Finais	9
-----	--------------------------------	---

1 Introdução

Este relatório apresenta uma aplicação blockchain educacional desenvolvida para demonstrar os conceitos fundamentais de tecnologia blockchain de forma prática e didática. O projeto foi construído utilizando Python no backend e React com TypeScript no frontend.

1.1 O que é o Projeto

É um sistema de blockchain local funcional que possui:

- 1.000.000 de tokens no total
- Sistema de cadastro de usuários
- Cada novo usuário recebe 10 tokens iniciais
- Permite transferências entre usuários
- Interface web para interação

1.2 Objetivos do Relatório

Este relatório busca explicar três conceitos fundamentais:

O que são blocos, hash e encadeamento criptográfico

Como novos blocos são adicionados à blockchain

Vantagens e limitações deste modelo

2 Explicação dos Conceitos de Bloco, Hash e Encadeamento

2.1 O que é um Bloco

Um bloco é como uma "caixa" que armazena informações na blockchain. Cada bloco contém:

- **Índice:** Número do bloco (1, 2, 3...)
- **Timestamp:** Data e hora de criação
- **Transações:** Lista de transferências de tokens
- **Hash anterior:** Referência ao bloco anterior

- **Proof:** Número que valida o bloco

Exemplo simplificado do código:

```

1 class Block:
2     def __init__(self, index, timestamp, transactions,
3                 previous_hash):
4         self.index = index
5         self.timestamp = timestamp
6         self.transactions = transactions
7         self.previous_hash = previous_hash
8         self.hash = self.calculat_hash()
9
10    def calculat_hash(self):
11        block_string = f"{self.index}{self.timestamp}" \
12                       f"{self.transactions}{self.previous_hash}"
13        return hashlib.sha256(block_string.encode()).hexdigest()

```

Listing 1: Estrutura básica de um bloco

2.2 O que é Hash

Hash é como uma "impressão digital" única de um bloco. É uma sequência de 64 caracteres gerada através de um algoritmo matemático (SHA-256) que transforma todas as informações do bloco em um código único.

2.2.1 Como Funciona

Pegamos todas as informações do bloco (índice, timestamp, transações, etc.)

Aplicamos o algoritmo SHA-256

Resultado: uma sequência única como "a3f5b8c2d9..."

Propriedades importantes:

- Mesma informação = mesmo hash (sempre)
- Qualquer mudança mínima = hash completamente diferente
- Impossível descobrir a informação original apenas pelo hash

Exemplo do código que calcula o hash:

```

1 def hash(self, block):
2     block_string = str(block).encode()
3     return hashlib.sha256(block_string).hexdigest()

```

Listing 2: Cálculo do hash de um bloco

2.3 O que é Encadeamento

Encadeamento é a forma como os blocos estão conectados uns aos outros, formando uma "corrente" que não pode ser quebrada.

2.3.1 Como Funciona o Encadeamento

Bloco 1 (Gênesis): Primeiro bloco, criado sem referência anterior

Bloco 2: Contém o hash do Bloco 1

Bloco 3: Contém o hash do Bloco 2

E assim por diante...

Por que isso é seguro?

Se alguém tentar alterar uma transação no Bloco 2:

- O hash do Bloco 2 muda completamente
- O Bloco 3 tem o hash antigo do Bloco 2 armazenado
- O sistema detecta que não batem e invalida a alteração
- Seria necessário alterar TODOS os blocos seguintes (praticamente impossível)

Código que valida o encadeamento:

```
1 def validat_chain(self):
2     for i in range(1, len(self.chain)):
3         block = self.chain[i]
4         previous_block = self.chain[i - 1]
5
6         if block['previous_hash'] != self.hash(previous_block):
7             return False
8
9         if not self.is_proof_valid(previous_block['proof'],
10                                     block['proof']):
11             return False
12     return True
```

Listing 3: Validação da integridade da cadeia

3 Como Novos Blocos São Adicionados à Cadeia

3.1 Processo Passo a Passo

Quando um usuário faz uma transferência de tokens, acontece o seguinte:

```

1 def add_transaction(self, sender, recipient, amount):
2     # Valida transa o atraves da economia de tokens
3     success, message = self.token_economy.transfer_tokens(
4         sender, recipient, amount)
5
6     if not success:
7         return False, message
8
9     transaction = {
10         'sender': sender,
11         'recipient': recipient,
12         'amount': amount,
13         'timestamp': self.get_current_timestamp(),
14         'hash': self.generat _transaction_hash(sender,
15                                             recipient, amount)
16     }
17
18     self.current_transactions.append(transaction)

```

Listing 4: Adição de transação

3.2 Proof of Work - A Mineração

Antes de adicionar um bloco, o sistema precisa resolver um quebra-cabeça matemático. Isso torna difícil adicionar blocos falsos.

3.2.1 Como Funciona

O sistema precisa encontrar um número que, quando combinado com informações do bloco anterior, gere um hash começando com quatro zeros.

```

1 def proof_of_work(self, last_proof):
2     proof = 0
3     while not self.is_proof_valid(last_proof, proof):
4         proof += 1
5     return proof
6
7 def is_proof_valid(self, last_proof, proof):
8     guess = f'{last_proof}{proof}'.encode()
9     guess_hash = hashlib.sha256(guess).hexdigest()
10    return guess_hash[:4] == "0000"

```

Listing 5: Algoritmo de Proof of Work

Por que isso é importante?

- Demora um tempo para encontrar esse número

- Dificulta fraudes (seria necessário refazer todo o trabalho)
- Todos podem verificar se o número está correto (é fácil conferir)

3.3 Criação e Adição do Bloco

Depois de resolver o quebra-cabeça, o bloco é criado:

```

1 def creat_block(self, proof, previous_hash):
2     block = {
3         'index': len(self.chain) + 1,
4         'timestamp': self.get_current_timestamp(),
5         'transactions': self.current_transactions,
6         'proof': proof,
7         'previous_hash': previous_hash,
8     }
9     self.current_transactions = []
10    self.chain.append(block)
11
12    # Persiste blockchain
13    self.save_blockchain()
14
15    return block

```

Listing 6: Criação e adição de bloco

3.4 Resumo do Processo Completo

Passo a passo de quando um usuário faz uma transferência:

Usuario solicita transferencia na interface web

Sistema valida: O usuário tem saldo suficiente?

Transacao é criada com remetente, destinatário e valor

Proof of Work: Sistema resolve o quebra-cabeça matemático

Novo bloco é criado contendo a transação

Hash anterior: Sistema pega o hash do último bloco

Bloco é adicionado ao final da cadeia

Dados são salvos em arquivo

Confirmação é enviada ao usuário

Analogia simples: É como escrever em um caderno onde cada página (bloco) tem um resumo (hash) da página anterior. Se alguém tentar apagar algo de uma página antiga, os resumos das páginas seguintes não vão bater, revelando a alteração.

4 Tecnologias Utilizadas

A aplicação foi desenvolvida em duas partes:

4.1 Backend (Python)

- **Python + Flask:** Servidor que processa as transações
- **SHA-256:** Algoritmo para gerar os hashes
- **JSON:** Formato para salvar os dados (blocos, usuários, saldos)

4.2 Frontend (React)

- **React + TypeScript:** Interface visual para os usuários
- **Telas:** Login, cadastro, dashboard com saldo e transferências

5 Vantagens do Modelo Desenvolvido

5.1 Transparência

Todos podem ver todas as transações e blocos da rede. Não há segredos sobre o histórico de operações.

5.2 Imutabilidade (Não Pode Ser Alterado)

Uma vez que uma transação está em um bloco, é praticamente impossível alterá-la ou apagá-la. O encadeamento protege contra modificações.

5.3 Segurança

- Senhas protegidas com hash
- Sistema verifica se você tem saldo antes de transferir
- Proof of Work dificulta fraudes

5.4 Sem Autoridade Central

Não precisa de um banco ou governo para validar as transações. A matemática e o código fazem isso automaticamente.

5.5 Auditável

É possível rastrear qualquer transação desde o início da blockchain até hoje.

6 Limitações do Modelo Desenvolvido

6.1 Velocidade

O Proof of Work deixa as transações mais lentas. Cada bloco demora um tempo para ser criado.

6.2 Escala

Este modelo funciona bem para aprendizado, mas não suportaria milhões de transações como Bitcoin ou Ethereum.

6.3 Armazenamento

Usa arquivos JSON simples. Para uma aplicação real, seria necessário um banco de dados mais robusto.

6.4 Rede

Roda apenas em um computador (local). Uma blockchain real precisa estar distribuída em vários computadores ao mesmo tempo.

6.5 Segurança Avançada

- Não tem criptografia de chave pública/privada (como carteiras reais)
- Sistema de login é básico
- Não tem proteção contra vários tipos de ataques

6.6 Funcionalidades Limitadas

Não implementa recursos avançados como:

- Smart contracts (contratos inteligentes)
- Taxas de transação
- Múltiplos tipos de moedas

7 Conclusão

7.1 O que Aprendemos

Este projeto demonstrou de forma prática os três conceitos principais:

Bloco, Hash e Encadeamento:

- Bloco é a caixa que guarda as transações
- Hash é a impressão digital única de cada bloco
- Encadeamento conecta os blocos de forma segura

Como Blocos São Adicionados:

- Transacao é criada
- Sistema resolve quebra-cabeça matemático (Proof of Work)
- Novo bloco é criado e encadeado ao anterior
- Dados são salvos e confirmados

Vantagens e Limitações:

- **Vantagens:** Transparente, seguro, imutavel, sem autoridade central
- **Limitações:** Lento, não escala bem, execução local, funcionalidades basicas

7.2 Considerações Finais

Este projeto é uma ferramenta educacional que simplifica conceitos complexos de blockchain. Embora tenha limitações em comparação com blockchains reais (Bitcoin, Ethereum), ele cumpre seu objetivo de ensinar os fundamentos de forma clara e prática.

A experiência de ver o código funcionando ajuda a entender por que blockchains são consideradas revolucionárias: elas permitem transações seguras e verificáveis sem precisar confiar em uma autoridade central.

Resumo em uma frase: Blockchain é uma cadeia de blocos conectados por hashes, onde cada bloco guarda transações de forma segura e transparente, e alterar o passado é praticamente impossível devido ao encadeamento criptográfico.