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

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:

1. O que são blocos, hash e encadeamento criptográfico
2. Como novos blocos são adicionados à blockchain
3. 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:

```
class Block:
    def __init__(self, index, timestamp, transactions,
                 previous_hash):
        self.index = index
        self.timestamp = timestamp
        self.transactions = transactions
        self.previous_hash = previous_hash
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        block_string = f"{self.index}{self.timestamp}" \
                       f"{self.transactions}{self.previous_hash}"
        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

1. Pegamos todas as informações do bloco (índice, timestamp, transações, etc.)
2. Aplicamos o algoritmo SHA-256
3. 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:

```
def hash(self, block):
    block_string = str(block).encode()
    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

1. **Bloco 1 (Gênesis)**: Primeiro bloco, criado sem referência anterior
2. **Bloco 2**: Contém o hash do Bloco 1
3. **Bloco 3**: Contém o hash do Bloco 2
4. 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:

```
def validate_chain(self):
    for i in range(1, len(self.chain)):
        block = self.chain[i]
        previous_block = self.chain[i - 1]

        if block['previous_hash'] != self.hash(previous_block):
            return False

        if not self.is_proof_valid(previous_block['proof'],
                                   block['proof']):
            return False
    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:

```python
def add_transaction(self, sender, recipient, amount):
    # Valida transacao atraves da economia de tokens
    success, message = self.token_economy.transfer_tokens(
        sender, recipient, amount)

    if not success:
        return False, message

    transaction = {
        'sender': sender,
        'recipient': recipient,
        'amount': amount,
        'timestamp': self.get_current_timestamp(),
        'hash': self.generate_transaction_hash(sender,
                                               recipient, amount)
    }

    self.current_transactions.append(transaction)
```

Listing 4: Adição de transação

3.2 Proof of Work - A Mineracao

Antes de adicionar um bloco, o sistema precisa resolver um quebra-cabeca matematico. Isso torna dificil adicionar blocos falsos.

3.2.1 Como Funciona

O sistema precisa encontrar um numero que, quando combinado com informacoes do bloco anterior, gere um hash comecando com quatro zeros.

```python
def proof_of_work(self, last_proof):
    proof = 0
    while not self.is_proof_valid(last_proof, proof):
        proof += 1
    return proof

def is_proof_valid(self, last_proof, proof):
    guess = f'{last_proof}{proof}'.encode()
    guess_hash = hashlib.sha256(guess).hexdigest()
    return guess_hash[:4] == "0000"
```

Listing 5: Algoritmo de Proof of Work

Por que isso eh importante?

- Demora um tempo para encontrar esse numero

- Dificulta fraudes (seria necessario refazer todo o trabalho)
- Todos podem verificar se o numero esta correto (eh facil conferir)

3.3 Criacao e Adicao do Bloco

Depois de resolver o quebra-cabeca, o bloco eh criado:

```
def create_block(self, proof, previous_hash):
    block = {
        'index': len(self.chain) + 1,
        'timestamp': self.get_current_timestamp(),
        'transactions': self.current_transactions,
        'proof': proof,
        'previous_hash': previous_hash,
    }
    self.current_transactions = []
    self.chain.append(block)

    # Persiste blockchain
    self.save_blockchain()

    return block
```

Listing 6: Criação e adição de bloco

3.4 Resumo do Processo Completo

Passo a passo de quando um usuario faz uma transferencia:

1. **Usuario solicita transferencia** na interface web
2. **Sistema valida**: O usuario tem saldo suficiente?
3. **Transacao eh criada** com remetente, destinatario e valor
4. **Proof of Work**: Sistema resolve o quebra-cabeca matematico
5. **Novo bloco eh criado** contendo a transacao
6. **Hash anterior**: Sistema pega o hash do ultimo bloco
7. **Bloco eh adicionado** ao final da cadeia
8. **Dados sao salvos** em arquivo
9. **Confirmacao** eh enviada ao usuario

Analogia simples: Eh como escrever em um caderno onde cada pagina (bloco) tem um resumo (hash) da pagina anterior. Se alguem tentar apagar algo de uma pagina antiga, os resumos das paginas seguintes nao vao bater, revelando a alteracao.

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 (Nao Pode Ser Alterado)

Uma vez que uma transacao esta em um bloco, eh praticamente impossivel altera-la ou apaga-la. O encadeamento protege contra modificacoes.

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

Nao precisa de um banco ou governo para validar as transacoes. A matematica e o codigo fazem isso automaticamente.

5.5 Auditavel

Eh possivel rastrear qualquer transacao desde o inicio da blockchain ate 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 Seguranca Avancada

- Nao tem criptografia de chave publica/privada (como carteiras reais)
- Sistema de login eh basico
- Nao tem protecao contra varios tipos de ataques

6.6 Funcionalidades Limitadas

Nao implementa recursos avancados como:

- Smart contracts (contratos inteligentes)

- Taxas de transacao
- Multiplos tipos de moedas

7 Conclusão

7.1 O que Aprendemos

Este projeto demonstrou de forma prática os três conceitos principais:

1. **Bloco, Hash e Encadeamento**:
   - Bloco eh a caixa que guarda as transacoes
   - Hash eh a impressao digital unica de cada bloco
   - Encadeamento conecta os blocos de forma segura
2. **Como Blocos Sao Adicionados**:
   - Transacao eh criada
   - Sistema resolve quebra-cabeca matematico (Proof of Work)
   - Novo bloco eh criado e encadeado ao anterior
   - Dados sao salvos e confirmados
3. **Vantagens e Limitacoes**:
   - **Vantagens**: Transparente, seguro, imutavel, sem autoridade central
   - **Limitacoes**: Lento, nao escala bem, execucao local, funcionalidades basicas

7.2 Consideracoes Finais

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

A experiencia de ver o codigo funcionando ajuda a entender por que blockchains sao consideradas revolucionarias: elas permitem transacoes seguras e verificaveis sem precisar confiar em uma autoridade central.

Resumo em uma frase: Blockchain eh uma cadeia de blocos conectados por hashes, onde cada bloco guarda transacoes de forma segura e transparente, e alterar o passado eh praticamente impossivel devido ao encadeamento criptografico.