Universidade de Fortaleza

PROGRAMAÇÃO FUNCIONAL (T300)	
Nome do Projeto: Blockchain	
TERMO DE ABERTURA DO PROJETO	
Professor: **Gilson Pereira do Carmo Filho**	**Semestre: 2024.1**
Avaliação: AV3	**Pontos: 10**

1. Justificativa

A tecnologia *blockchain* é um mecanismo de banco de dados avançado que permite o compartilhamento transparente de informações em uma rede de computadores, armazenando os dados em blocos interligados em uma cadeia. Enquanto as tecnologias de bancos de dados tradicionais possuem vários desafios para o registro de transações, a *blockchain* reduz esses problemas criando um sistema descentralizado, à prova de violações.

Os dados na *blockchain* são cronologicamente consistentes porque não é possível excluir nem modificar a cadeia sem o consenso da rede. Como resultado, a *blockchain* pode ser usada para criar um livro-razão (*ledger*) inalterável ou imutável para monitorar pedidos, pagamentos, contas e outras transações. O sistema tem mecanismos integrados que impedem entradas de transações não autorizadas e criam consistência na visualização compartilhada dessas transações.

Apesar de ser uma tecnologia relativamente recente, a *blockchain* está sendo adotada em vários setores além da criação de moedas digitais: energia, financeiro, mídia e entretenimento e varejo.

2. Descrição do Produto

O produto a ser desenvolvido neste projeto deverá ser uma API (*Application Programming Interface*) para registrar, em uma *blockchain*, as transações do gerenciador de finanças pessoais construído no livro **Programação Funcional: Uma Introdução em Clojure**.

Blockchain é um tipo de base de dados distribuída que guarda um registro de transações permanente e inviolável. A base de dados *blockchain* consiste em dois tipos de registros: transações individuais e blocos.

Um bloco é a parte concreta da *blockchain* onde são registradas algumas ou todas as transações mais recentes. Toda vez que um bloco é concluído, este é guardado

permanentemente na *blockchain* e um novo bloco é gerado. Existe um número incontável de blocos que são ligados uns aos outros, como uma cadeia, onde cada bloco contém uma referência para o bloco anterior.

O bloco inicial (gênese) da *blockchain* deverá ser codificado na aplicação e serve como o estado inicial do sistema. Ele pode conter informações sobre as regras ou instruções sobre o banco de dados restante. Feito isto, o banco de dados será formado a partir de uma série de blocos que, juntos, formam uma cadeia.

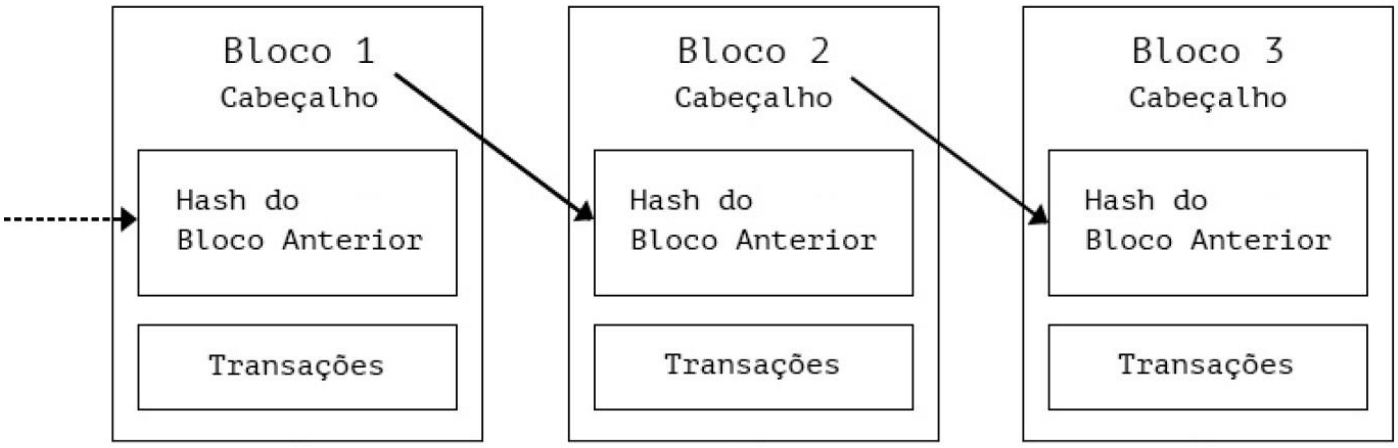
Cada bloco na cadeia deverá conter uma lista de transações. À medida em que forem adicionadas transações no gerenciador de finanças, elas devem ser armazenadas no bloco e uma assinatura (*hash*) deve então ser adicionada no mesmo. O *hash* apresenta-se como uma chave criptografada e deverá ser gerado pela função de dispersão criptográfica SHA-256. Como exemplo, um gerador de *hash* com essa função pode ser acessado clicando aqui.

Portanto, cada bloco na cadeia deverá ter as seguintes informações armazenadas:

- **Número do bloco:** número sequencial que identifica unicamente o bloco;
- **Nonce (*Number used only once*):** número necessário para gerar um *hash* com 4 zeros à esquerda. Esse número deve ser gerado pelo algoritmo de consenso Prova de Trabalho (*Proof-of-Work* - PoW) durante o processo de criação (mineração) do bloco;
- **Dados:** lista de transações a ser armazenada;
- ***Hash* prévio:** *hash* do bloco anterior;
- ***Hash*:** *hash* do bloco, gerado a partir dos quatro itens acima.

Neste ponto, a ideia de uma cadeia deve estar clara: cada novo bloco contém dentro de si o *hash* do bloco anterior. Assim, o *hash* de cada bloco está ligado ao bloco anterior da cadeia. Esses *hashes* formam ligações voltando entre as cadeias até chegar ao bloco gênese. Isso é crucial porque é o que dá imutabilidade às *blockchains*: se um invasor corromper um bloco anterior na cadeia, todos os blocos subsequentes conterão *hashes* incorretos.

Como exemplo, um simulador de *blockchain* pode ser acessado clicando aqui. A utilização deste simulador, juntamente com gerador de *hash*, é demonstrada no vídeo Blockchain na prática.



Blockchain simplificada

3. Principais Entregas

As principais entregas do projeto são:

ENTREGAS	REQUISITOS
Gerenciador Financeiro	• O gerenciador de finanças pessoais deverá ser implementado conforme apresentado no livro **Programação Funcional: Uma Introdução em Clojure**; • Uma interface de usuário deverá ser disponibilizada com opções para cadastrar e exibir as transações.

Blockchain	• A cadeia de blocos deverá ser implementada por meio de **mapas** (*hash-maps*) armazenados em um **átomo** (*atom*);
	• A API da *blockchain* deverá contar com métodos para:
	  ○ Criar uma transação em um bloco;
	  ○ Minerar um novo bloco;
	  ○ Devolver a cadeia completa.
	• Uma interface de usuário deverá ser disponibilizada com opções para registrar as transações na *blockchain* e exibir os seus blocos.

4. Estratégia de Condução

- Dúvidas acerca dos requisitos do projeto deverão ser esclarecidas com o professor;
- O código fonte do projeto deverá ser enviado para o AVA até o prazo estipulado para a entrega. Não serão aceitas entregas após o prazo;
- Os projetos deverão ser apresentados pessoalmente ao professor na data definida. Trabalhos feitos em dupla deverão ser apresentados por ambos os alunos;
- A apresentação consistirá em **execução e teste** do aplicativo, seguido de **arguição do professor** sobre o código fonte. **Trabalhos não apresentados receberão a nota ZERO**;
- O código fonte do projeto será submetido a uma ferramenta de verificação de plágio. Qualquer tentativa de cópia do projeto de outro aluno ou da Internet, ou qualquer outra tentativa de fraudar o projeto, incluindo cópia de trechos do código fonte, resultará em aplicação de nota **ZERO**.

5. Critério de Avaliação

Na avaliação do projeto serão consideradas a execução correta das funcionalidades do programa e a conformidade do código fonte ao conteúdo abordado na disciplina.

A tabela a seguir mostra a pontuação das entregas do projeto.

ENTREGA	PONTOS
Gerenciador Financeiro	**3,0**
API (livro)	1,5
Interface de usuário	1,5
Blockchain	**7,0**
API	5,0
Interface de usuário	2,0
TOTAL	**10,0**

6. Premissas e Restrições

PREMISSAS	RESTRIÇÕES
• Os alunos deverão ter estudado os **capítulos 9 a 13** do livro **Programação Funcional: Uma Introdução em Clojure**.	• O projeto tem um prazo total de 4 (quatro) semanas; • O *software* deverá ser desenvolvido na linguagem **Clojure**; • O projeto deverá ser realizado em dupla ou individualmente.