Desenvolvimento de uma *Blockchain* em *Rust* para Livro de Registro Acadêmico

Eduardo Augusto Silva Zborowski, Ana Paula Canal
Curso de Ciência da Computação
UFN - Universidade Franciscana
Santa Maria - RS
eduardo.zborowski@ufn.edu.br, apc@ufn.edu.br

***Resumo*—Este trabalho apresenta o desenvolvimento de uma *blockchain* em *Rust* voltada para a criação de um livro de registro acadêmico. A *blockchain* é uma tecnologia distribuída que oferece imutabilidade, transparência e segurança, tornando-a ideal para garantir a integridade e a disponibilidade de registros ao longo do tempo. O objetivo é desenvolver uma solução que preserve registros de forma distribuída e segura, seguindo os princípios da metodologia *FDD* (*Feature Driven Development*). Os principais resultados incluem a implementação de uma *blockchain* funcional e a documentação do processo de desenvolvimento.**

***Palavras-chave*: Segurança; Redes P2P; Sistema Distribuído**

I. Introdução

A preservação de registros é crucial para garantir a integridade e a disponibilidade de informações importantes ao longo do tempo. No entanto, os métodos tradicionais de armazenamento de dados, como o uso de bases de dados centralizadas, *backups* físicos e servidores de arquivos, podem ser suscetíveis a falhas e manipulações, o que levanta preocupações sobre a autenticidade e a confiabilidade dos registros [1].

A tecnologia *blockchain* surgiu em 2008 como o fundamento do Bitcoin, criado por Satoshi Nakamoto, e desde então tem evoluído para uma ampla gama de aplicações além das criptomoedas [2]. Inicialmente voltada para transações financeiras, a *blockchain* demonstrou seu potencial para revolucionar diversos setores, oferecendo um método seguro e transparente de registro de dados [3]. Hoje, a *blockchain* é reconhecida por sua capacidade de manter registros imutáveis, garantindo que as informações não possam ser alteradas ou removidas uma vez registradas [4, 5].

Neste contexto, a tecnologia *blockchain* surge como uma solução promissora. Uma *blockchain* é um registro digital distribuído que mantém uma lista crescente de registros (ou blocos) interligados de forma segura e transparente. Cada bloco na *blockchain* contém um registro criptograficamente seguro de transações, tornando extremamente difícil alterar os dados armazenados sem a concordância da maioria dos participantes da rede [2, 6].

Este trabalho tem como objetivo desenvolver uma *blockchain* em *Rust* especificamente projetada para servir como livro de registro acadêmico. A escolha da linguagem *Rust* se deve à sua segurança e desempenho, tornando-a uma excelente opção para o desenvolvimento de sistemas críticos e confiáveis [6, 7]. A *blockchain* proposta foi implementada como uma aplicação de código aberto, permitindo sua auditoria e contribuições da comunidade.

A. Justificativa

A implementação de uma *blockchain* em *Rust* para criação de um livro de registro oferece várias vantagens. Em primeiro lugar, a *blockchain* proporciona imutabilidade aos registros, garantindo que as informações armazenadas não possam ser alteradas ou apagadas retroativamente [1, 8]. Isso garante a integridade dos dados ao longo do tempo, fornecendo uma trilha de auditoria transparente para todas as transações [5].

Dentro de uma *blockchain*, não é possível alterar ou remover registros, pois cada bloco é criptograficamente ligado ao bloco anterior, formando uma cadeia inquebrável [2]. Qualquer tentativa de modificar um bloco invalidaria todos os blocos subsequentes, tornando a *blockchain* resistente a adulterações [1, 8].

Além disso, a natureza distribuída da *blockchain* elimina a necessidade de um ponto centralizado de controle, reduzindo assim os riscos de falhas de segurança e manipulação de dados. Cada nó na rede *blockchain* mantém uma cópia idêntica do registro como mostrado no apêndice, garantindo redundância e resiliência contra ataques maliciosos ou falhas de hardware [5].

A escolha da linguagem *Rust* é devido à sua ênfase em segurança de memória e desempenho, características essenciais para sistemas distribuídos como uma *blockchain*. O gerenciamento seguro de memória e a prevenção de falhas comuns, como erros de segmentação e condições de corrida, tornam *Rust* ideal para o desenvolvimento de uma *blockchain* robusta [7].

Além das vantagens técnicas, a implementação de uma *blockchain* em *Rust* também pode trazer benefícios significativos para a preservação de registros em situações de desastre, como as inundações que ocorrem no Rio Grande do Sul. Esses eventos climáticos têm causado prejuízos consideráveis, incluindo a perda de documentos e registros importantes devido a danos físicos [9]. Com uma *blockchain* distribuída, os registros podem ser mantidos de forma segura

e acessível, independentemente de danos físicos aos locais de armazenamento tradicionais. Assim, mesmo em situações de desastre, as informações essenciais permanecem protegidas e disponíveis, garantindo a continuidade das operações e a preservação da história e dados críticos [10].

Portanto, a combinação da tecnologia *blockchain* com a linguagem *Rust* oferece uma solução robusta e segura para a preservação de registros, mitigando riscos associados a falhas de segurança, desastres naturais e tentativas de adulteração.

B. Objetivos

O principal objetivo deste trabalho é desenvolver uma *blockchain* em *Rust* que seja capaz de servir como um livro de registro acadêmico. Para alcançar esse objetivo, são definidos os seguintes objetivos específicos:

- Pesquisar e compreender os princípios fundamentais da tecnologia *blockchain*.
- Projetar a arquitetura da *blockchain* em *Rust*, incluindo a definição de estruturas de dados, algoritmos de consenso e criptografia.
- Implementar a *blockchain* em *Rust*, seguindo as melhores práticas de desenvolvimento de software.
- Validar a *blockchain* e suas funcionalidades através de simulações de transações em um ambiente controlado.

II. Referencial Teórico

Serão discutidos os conceitos relacionados à tecnologia *blockchain*, definição, características principais, algoritmos de consenso e aplicações em diferentes domínios, bem como Livro de Registro Acadêmico, a linguagem Rust e os trabalhos correlatos.

A. Definição e Características da Blockchain

Uma *blockchain* é um registro digital distribuído que mantém uma lista crescente de registros (ou blocos) interligados de forma segura e transparente. Cada bloco na *blockchain* contém um registro criptograficamente seguro de transações, tornando extremamente difícil alterar os dados armazenados sem a concordância da maioria dos participantes da rede [2].

A *blockchain* opera em uma rede de computadores (nós), onde cada nó mantém uma cópia completa da *blockchain*. Quando uma nova transação é realizada, ela é transmitida para a rede e validada por todos os nós. Uma vez validada, a transação é adicionada a um novo bloco, que é então anexado à *blockchain* existente. Este processo garante que todas as transações sejam verificadas e registradas de forma consistente em toda a rede, proporcionando uma base de dados imutável e auditável [11].

Outra definição de *blockchain* é fornecida por Narayanan et al. (2016), que descrevem a *blockchain* como um mecanismo de consenso distribuído que permite a transferência de valor ou informação entre partes sem a necessidade de um intermediário de confiança. Segundo os autores, a *blockchain* se destaca por sua capacidade de resistir à censura e por proporcionar transparência e segurança através de algoritmos criptográficos e protocolos de consenso descentralizados [11].

B. Blocos

Os blocos são os componentes fundamentais da *blockchain*. Cada bloco contém um conjunto de transações, um *hash* do bloco anterior, um *hash* do bloco atual, um *timestamp* e outros metadados relevantes. Os blocos são interligados de forma que qualquer alteração em um bloco invalida todos os blocos subsequentes [8]. Esta estrutura encadeada assegura a integridade dos dados, pois qualquer tentativa de alteração em um bloco seria detectada imediatamente. Dentro de um bloco, as transações são agrupadas e cada uma possui um identificador único. A estrutura do bloco também inclui a função *hash*, que é calculada a partir dos dados do bloco. Esta função *hash* é um valor criptográfico que serve como uma impressão digital do bloco, garantindo que qualquer alteração nos dados resultará em um *hash* completamente diferente [2].

C. Função Hash

A função *hash* é uma função criptográfica que gera um valor fixo a partir de uma entrada de dados. Na *blockchain*, a função *hash* é utilizada para garantir a integridade dos dados armazenados, pois qualquer alteração nos dados resulta em um *hash* completamente diferente [12]. No contexto deste projeto, a função *hash* foi implementada utilizando o algoritmo SHA-256. O SHA-256 é uma função de *hash* de criptografia que gera um *hash* de 256 bits (32 bytes) para qualquer entrada, proporcionando uma segurança robusta. [13].

Na Figura 1, há um exemplo de cadeia de blocos em uma *blockchain*, onde cada bloco contém um *hash* que depende do bloco anterior.

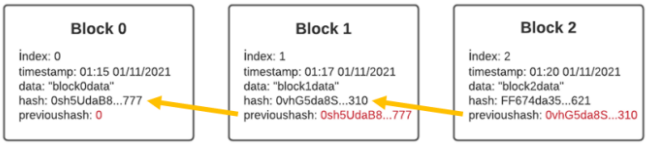

Figura 1. Exemplo de cadeia de blocos em uma *blockchain*. Adaptado de [14].

D. Transação

Uma transação na *blockchain* representa uma operação ou evento registrado. As transações são agrupadas em blocos e validadas pelos participantes da rede antes de serem adicionadas à *blockchain* [5]. Cada transação é assinada digitalmente pelo remetente, garantindo a autenticidade e a integridade da informação transmitida. A assinatura digital utiliza criptografia assimétrica, onde uma chave privada é usada para assinar a transação e uma chave pública

correspondente é usada para verificar a assinatura. Este processo assegura que apenas o proprietário da chave privada pode autorizar transações, proporcionando um alto nível de segurança [2].

Cada transação inclui a chave pública do novo proprietário e o *hash* da transação anterior. O proprietário atual usa sua chave privada para assinar a transação, e o novo proprietário verifica essa assinatura com a chave pública do anterior. Esse processo garante a integridade e autenticidade das transações ao longo da cadeia.

E. Algoritmos de Consenso

O consenso é o mecanismo pelo qual os participantes da rede *blockchain* concordam sobre o estado atual da *blockchain*. Para este projeto, considerando que não há criptomoeda envolvida, foi utilizado o algoritmo de consenso *Proof of Authority* (PoA) [15].

O PoA é um método simples e eficiente, onde um conjunto de autoridades pré-definidas valida os blocos. Este algoritmo é adequado para sistemas permissionados, onde os validadores são entidades conhecidas e confiáveis. No PoA, a identidade dos validadores é a principal forma de garantir a segurança e a integridade da *blockchain*, reduzindo o risco de ataques maliciosos [16].

A escolha do *Proof of Authority* (PoA) para este projeto deve-se a vários fatores, como em que o PoA é mais eficiente em termos de consumo de energia e velocidade de validação de transações em comparação com outros algoritmos como PoW [15]. Em um sistema permissionado, onde os validadores são entidades conhecidas, a identidade das autoridades proporciona uma camada adicional de segurança [17]. O mecanismo do PoA é mais simples de implementar e administrar, uma vez que não exige o mesmo nível de recursos computacionais que outros algoritmos [18].

As autoridades no PoA são entidades pré-definidas e confiáveis que têm a responsabilidade de validar novos blocos. Essas autoridades podem incluir tanto agências reguladoras que desejam manter a integridade dos registros [19]. Bancos e outras empresas financeiras que precisam garantir a segurança das transações [20].

A presença de autoridades confiáveis reduz significativamente o risco de atividades maliciosas na rede, garantindo que apenas entidades verificadas possam validar transações e criar novos blocos.

F. Segurança e Desempenho

A *blockchain* utiliza criptografia e algoritmos de consenso para garantir a segurança dos dados armazenados. A segurança é reforçada por técnicas criptográficas que garantem a imutabilidade e a integridade dos dados [5].

A segurança foi garantida pela implementação de algoritmos criptográficos, como o SHA-256 para as funções *hash* [21] e a criptografia RSA para a assinatura digital das transações [22]. Além disso, a rede utilizará certificados digitais para autenticar os nós participantes, garantindo que apenas entidades autorizadas possam validar e registrar transações. No contexto deste trabalho, é crucial seguir as melhores práticas de segurança cibernética para proteger os dados e os participantes da rede. Isso inclui a adoção de mecanismos de autenticação fortes, a utilização de canais de comunicação seguros [23].

TLS (*Transport Layer Security*) e SSL (*Secure Sockets Layer*) são protocolos criptográficos que fornecem comunicação segura sobre uma rede de computadores [23]. Eles garantem que os dados transmitidos entre os nós da rede sejam criptografados, impedindo que terceiros interceptem ou alterem a informação [24].

Um certificado digital é um documento eletrônico que utiliza uma assinatura digital para vincular uma chave pública a uma identidade [25]. Ele autentica a identidade dos nós participantes na rede blockchain, assegurando que apenas entidades autorizadas possam realizar transações e validar blocos [26].

G. Plataformas Blockchain

Existem várias plataformas *blockchain*, cada uma com suas próprias características e funcionalidades. A primeira e mais conhecida plataforma de *blockchain*, utilizada principalmente para transações financeiras [2]. O *Bitcoin* utiliza o algoritmo de consenso *Proof of Work* (PoW). Uma plataforma que suporta contratos inteligentes e aplicações descentralizadas (*dApps*) [18]. O *Ethereum* inicialmente utilizava PoW, mas está em processo de transição para *Proof of Stake* (PoS), conhecido como *Ethereum 2.0*, para melhorar a eficiência energética e a escalabilidade.

H. Aplicações da Blockchain

A tecnologia *blockchain* tem sido amplamente adotada em uma variedade de domínios, nas finanças a *blockchain* é utilizada em sistemas de pagamento e transferência de dinheiro, oferecendo transações mais rápidas, seguras e econômicas [3]. Instituições financeiras utilizam *blockchain* para reduzir custos operacionais e aumentar a transparência nas transações. As criptomoedas são uma das aplicações mais conhecidas da *blockchain*, oferecendo uma forma descentralizada de realizar transações financeiras [2]. Na área da saúde, a *blockchain* é utilizada para armazenar registros médicos de forma segura e transparente, garantindo a privacidade e a integridade dos dados dos pacientes [27]. Na educação, a *blockchain* pode ser usada para preservar registros acadêmicos, certificados e diplomas de forma imutável e transparente [4].Instituições de ensino utilizam *blockchain* para emitir e verificar credenciais acadêmicas. Contratos inteligentes são programas que executam automaticamente as condições e termos de um contrato sem a necessidade de intermediários [28]. São amplamente utilizados para automatizar processos comerciais e legais. A integração da *blockchain*

com dispositivos *IoT* pode aumentar a segurança e a transparência das transações e dados coletados[29]. Empresas utilizam *blockchain* para gerenciar e monitorar dispositivos *IoT* de forma segura. A *blockchain* é utilizada para rastrear o movimento de produtos e garantir a autenticidade e a integridade dos registros em toda a cadeia de suprimentos [30]. Ela ajuda a prevenir fraudes e aumenta a transparência nas operações logísticas. Governos e organizações usam *blockchain* para aumentar a transparência e a eficiência dos processos administrativos, como votação eletrônica e registro de propriedade [31].

I. Livro de Registro Acadêmico

Um livro de registro é um documento ou sistema utilizado para registrar e armazenar informações de forma organizada e acessível. Na *blockchain*, o livro de registro é mantido de forma distribuída, garantindo a integridade e a disponibilidade dos registros ao longo do tempo. Este registro pode ser utilizado em diversos contextos, como registros financeiros, médicos e acadêmicos, proporcionando uma trilha auditável de todas as transações. A distribuição dos registros entre múltiplos nós na rede *blockchain* assegura que os dados permaneçam acessíveis mesmo em caso de falhas de hardware ou ataques cibernéticos.

Uma *blockchain* como livro de registro acadêmico opera de maneira descentralizada, onde cada nó da rede possui uma cópia completa do registro(ver Apêndice 6). Cada entrada de dado (por exemplo, uma nota ou certificado) é transformada em um bloco que, após verificação e validação por consenso entre os nós, é adicionado à cadeia de blocos de forma imutável. Este sistema oferece várias vantagens: registro descentralizado, imutabilidade, verificação e validação, transparência, acessibilidade e segurança.

Suponha uma universidade que deseja implementar um sistema de *blockchain* para gerenciar seus registros acadêmicos. Ao se matricular, os dados do estudante são registrados na *blockchain*. Cada nota obtida pelo estudante é adicionada como um bloco na *blockchain*, após validação pelo consenso da rede. Exemplos práticos incluem o MIT Media Lab, que implementou um sistema de diplomas digitais baseado em *blockchain*, permitindo a verificação de credenciais acadêmicas de forma segura e eficiente [32], e a Sony Global Education, que desenvolveu uma plataforma baseada em *blockchain* para gerenciar registros acadêmicos e certificados, proporcionando maior transparência e confiança no sistema educacional [33].

J. Metodologia FDD

A metodologia *FDD* (*Feature Driven Development*) segue um processo estruturado composto por cinco atividades principais.[34].

- Desenvolver um Modelo Geral: Esta etapa envolve a criação de um modelo abrangente do domínio do problema, ajudando a estabelecer uma visão compartilhada e compreensiva do projeto.
- Construir uma Lista de Funcionalidades: Baseando-se no modelo geral, uma lista detalhada de funcionalidades é criada, priorizando o que é mais importante para o negócio e para os usuários.
- Planejar por Funcionalidade: Desenvolve-se um plano detalhado para a implementação de cada funcionalidade, definindo as prioridades e alocando recursos adequadamente.
- Projetar por Funcionalidade: Nesta fase, um projeto detalhado é desenvolvido para cada funcionalidade, garantindo que todos os aspectos técnicos e de design sejam considerados antes do desenvolvimento.
- Construir por Funcionalidade: A etapa final envolve a construção real da funcionalidade conforme o projeto detalhado, seguida de testes rigorosos para assegurar que a funcionalidade atenda aos requisitos especificados.

Essas duas últimas etapas são iterativas e incrementais, permitindo que o desenvolvimento seja flexível e adaptável às mudanças e necessidades do projeto [34].

K. Linguagem Rust

A linguagem *Rust* foi desenvolvida em 2010 pela Mozilla Research. *Rust* é uma linguagem de programação de sistemas que se destaca por sua segurança e desempenho. *Rust* previne problemas comuns como erros de memória e condições de corrida. Sua eficiência a torna ideal para aplicações de alto desempenho, como a *blockchain* [6].

Rust segue paradigmas de programação orientada a objetos e funcional, permitindo a criação de código seguro e eficiente. A ausência de coleta de lixo (*garbage collection*) e o gerenciamento de memória baseado em regras de posse (*ownership*) garantem que os programas em *Rust* sejam rápidos e confiáveis [7].

A biblioteca RustCrypto foi utilizada para implementar as funções de *hash* e algoritmos de criptografia necessários para a *blockchain*. RustCrypto é uma coleção de bibliotecas de criptografia escrita em *Rust*, conhecida por sua eficiência e segurança [35].

L. Trabalhos Correlatos

Diversos trabalhos anteriores têm explorado o uso de *blockchain* para preservação de registros e outras aplicações.

- Abouelmehdi et al. [27] analisa a viabilidade da *blockchain* no setor de saúde, destacando os benefícios em termos de segurança e privacidade dos dados. O estudo sugere que a *blockchain* pode melhorar a integridade dos registros médicos e facilitar o compartilhamento seguro de informações entre instituições. Este trabalho contribuiu para a pesquisa ao demonstrar a importância da segurança e privacidade em sistemas de *blockchain*,

aspectos fundamentais para a implementação de um livro de registros acadêmicos.

- Grech et al. [4] investiga a aplicação da *blockchain* na educação, enfatizando a imutabilidade e a transparência dos registros acadêmicos. A pesquisa destaca como a *blockchain* pode garantir a autenticidade dos certificados e diplomas emitidos por instituições de ensino. Este estudo foi essencial para entender a aplicação prática da *blockchain* no contexto educacional, fornecendo *insights* sobre a importância da imutabilidade e da autenticidade dos registros, elementos cruciais para meu projeto.
- Christidis et al. [29] explora a integração da *blockchain* com dispositivos *IoT*, demonstrando como a tecnologia pode aumentar a segurança e a transparência dos dados coletados por esses dispositivos. O estudo propõe soluções para os desafios de escalabilidade e segurança na implementação de *blockchain* em ambientes *IoT*. A análise deste trabalho ajudou a identificar os desafios de escalabilidade e segurança, que também são relevantes para a implementação de uma *blockchain* eficiente e confiável para registros acadêmicos.

Em comparação, o presente trabalho foca no desenvolvimento de uma *blockchain* em *Rust* para a criação de um livro de registro, diferenciando-se pelo uso de *Rust* e pela abordagem orientada a funcionalidades.

III. Desenvolvimento da Proposta

Para o desenvolvimento da *blockchain* em *Rust*, foi utilizada a metodologia *FDD* (*Feature-Driven Development*), que se concentra na entrega contínua de funcionalidades ao longo do ciclo de desenvolvimento. As principais tecnologias utilizadas no desenvolvimento da *blockchain* incluem a linguagem de programação *Rust*, escolhida por sua segurança e desempenho, e a biblioteca de criptografia *RustCrypto*. Além disso, o algoritmo de consenso *Proof of Authority* (PoA) foi implementado para garantir que todos os nós da rede concordem com o estado atual da *blockchain*.

A. Arquitetura da Blockchain

A arquitetura da *blockchain* proposta é composta pelos seguintes componentes principais:

- Bloco: Cada bloco contém uma lista de transações, um *hash* do bloco anterior, um *hash* do bloco atual, um *timestamp* e outros metadados relevantes.
- Transação: Representa uma operação ou evento registrado na *blockchain*.
- Função *Hash*: Utilizada para gerar *hashes* criptograficamente seguros dos blocos.
- Algoritmo de Consenso: Utilizado pelos nós da rede para validar novos blocos.
- Nó da Rede: Cada nó mantém uma cópia da *blockchain* e participa do processo de validação de blocos.

A seguir, o desenvolvimento da proposta é apresentado conforme as etapas da metodologia FDD. Nesta etapa, é criado um modelo geral da *blockchain*, incluindo os principais componentes e suas interações. Diagramas *UML* são utilizados para representar a arquitetura geral do sistema, ajudando a visualizar a estrutura e o fluxo de dados.

A Figura 2 mostra a estrutura da *blockchain* onde nós da rede validam blocos, que contêm transações. Blocos são validados por um algoritmo de consenso e protegidos por funções *hash* que garantem a integridade dos dados.

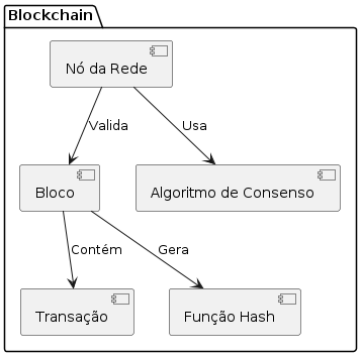

Figura 2. Diagrama de Componentes da *Blockchain*. Fonte: Elaborado pelo autor.

A Figura 3 detalha o processo de adição de um bloco na *blockchain*. Ele mostra a interação entre um cliente, nó da rede, algoritmo de consenso e a *blockchain*. O cliente solicita uma transação, que é então validada pelo nó da rede. O algoritmo de consenso é iniciado para validar o bloco, e uma vez validado, o bloco é adicionado à blockchain, e a transação é confirmada ao cliente.

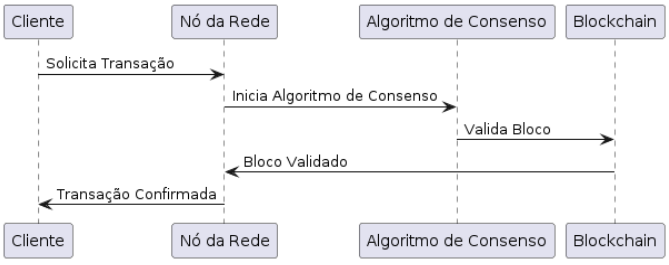

Figura 3. Diagrama de Sequência para Adição de Bloco. Fonte: Elaborado pelo autor.

A Figura 4 contém o fluxo de trabalho para a validação de transações na *blockchain*. O processo começa com o recebimento de uma transação, seguida pela sua validação. Se a transação for válida, ela é adicionada a um bloco; caso contrário, é rejeitada. O próximo passo é calcular o *hash* do bloco e, finalmente, distribuir o bloco validado para os nós da rede.

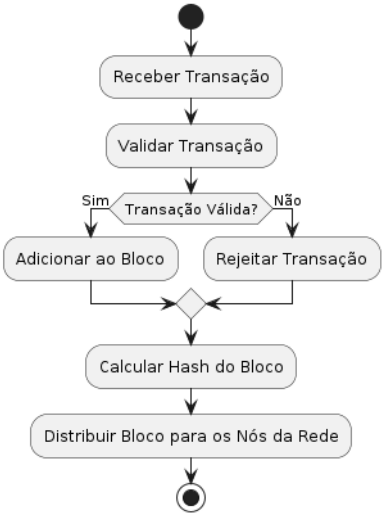

Figura 4. Diagrama de Atividades para Validação de Transações. Fonte: Elaborado pelo autor.

B. Construir uma Lista de Funcionalidades

A partir do modelo geral, é criada uma lista detalhada de funcionalidades que a *blockchain* deve oferecer. Cada funcionalidade é descrita de maneira clara e objetiva, permitindo a priorização e o planejamento das atividades de desenvolvimento.

- Registro de transações de forma imutável.
- Validação de transações utilizando o algoritmo de consenso PoA.
- Implementação de funções *hash* para garantir a integridade dos dados.
- Autenticação dos nós participantes utilizando certificados digitais.
- Criação de uma interface para consulta e auditoria dos registros.

C. Planejar por Funcionalidade

Nesta fase, cada funcionalidade é planejada individualmente. São definidos os requisitos, os recursos necessários, os prazos e as responsabilidades. O planejamento detalhado garante que todas as funcionalidades sejam desenvolvidas de acordo com as expectativas do desenvolvedor.

Tabela I
PLANEJAMENTO POR FUNCIONALIDADE

Funcionalidade	**Requisitos**	**Prazo**
Registro de transações	Segurança, Imutabilidade	2 semanas
Validação de transações	Algoritmo PoA	3 semanas
Implementação de funções *hash*	Algoritmo SHA-256	2 semanas
Autenticação de nós	Certificados digitais	2 semanas
Interface de consulta	Auditoria, Transparência	4 semanas

D. Projeto das Funcionalidades

Neste estágio, detalhamos as funcionalidades essenciais implementadas na blockchain acadêmica, fundamentando-se no código desenvolvido em Rust e explicando de forma técnica e concisa.

1) Registro de Transações: Cada transação é representada pela estrutura `Transacao`, que encapsula dados relevantes como ID da transação, informações do estudante, período letivo, disciplinas e notas. Ao criar uma transação, ela é adicionada a uma fila de transações pendentes na instância da blockchain. Essa fila é implementada usando uma estrutura de fila dupla, permitindo inserções e remoções eficientes.

No módulo principal, as transações são coletadas a partir da entrada do usuário. Após a criação, a transação é adicionada à blockchain local e difundida para a rede P2P utilizando o protocolo *Gossipsub* da biblioteca `libp2p`. Isso assegura que todos os nós da rede recebam e armazenem as transações pendentes, mantendo a consistência dos dados.

```rust
use serde::{Serialize, Deserialize};
use chrono::{DateTime, Utc};
use super::{Estudante, PeriodoLetivo};

#[derive(Serialize, Deserialize, Debug, Clone)]
pub struct Transacao {
    pub id_transacao: u32,
    pub estudante: Estudante,
    pub periodo_letivo: PeriodoLetivo,
    pub timestamp: DateTime<Utc>,
}

impl Transacao {
    pub fn nova_transacao(
        id_transacao: u32,
        estudante: Estudante,
        periodo_letivo: PeriodoLetivo,
    ) -> Self {
        let timestamp = Utc::now();
        Transacao {
            id_transacao,
            estudante,
            periodo_letivo,
            timestamp,
        }
    }
}
```

Listing 1. Estrutura `Transacao` representando uma transação na blockchain.

A estrutura `Transacao` encapsula informações acadêmicas, como ID da transação, estudante, período letivo e o `timestamp` de criação. A utilização de `Serialize` e `Deserialize` facilita a conversão entre a estrutura e formatos como JSON, essencial para persistência e comunicação em rede. O método `nova_transacao` cria instâncias dessa estrutura, incluindo automaticamente o `timestamp` atual.

2) Validação de Transações com Proof of Authority (PoA): Para garantir que apenas transações válidas sejam registradas na blockchain, foi implementado o algoritmo de consenso *Proof of Authority* (PoA), onde nós autorizados, identificados como autoridades, são responsáveis pela validação e criação de blocos. Cada autoridade possui uma chave privada RSA para assinar digitalmente os blocos, enquanto

as chaves públicas correspondentes são conhecidas pelos demais nós e armazenadas em um arquivo de configuração.

Quando um nó autoridade decide criar um novo bloco, ele reúne as transações pendentes e utiliza sua chave privada para assinar o bloco. Essa assinatura é anexada ao bloco junto com o ID da autoridade. Ao receber um novo bloco, os nós validam a assinatura utilizando a chave pública da autoridade correspondente. Se a assinatura for válida e o bloco estiver corretamente encadeado na *blockchain* — ou seja, o *hash* do bloco anterior coincide com o *hash* atual do último bloco local — o bloco é aceito e adicionado à cadeia.

Esse mecanismo garante que apenas blocos provenientes de autoridades reconhecidas sejam incorporados, reforçando a segurança e a integridade do sistema.

```rust
use rsa::{RsaPrivateKey, pkcs1v15::SigningKey};
use base64::encode;
use serde::{Serialize, Deserialize};
use sha2::{Digest, Sha256};
use chrono::Utc;

#[derive(Serialize, Deserialize, Debug, Clone)]
pub struct Bloco {
    pub indice: u32,
    pub hash_anterior: String,
    pub hash_atual: String,
    pub timestamp: chrono::DateTime<Utc>,
    pub transacoes: Vec<Transacao>,
    pub id_autoridade: u32,
    pub assinatura_autoridade: String,
}

impl Bloco {
    pub fn assinar_bloco(&mut self, chave_privada: &
         RsaPrivateKey, id_autoridade: u32) {
        self.id_autoridade = id_autoridade;
        self.hash_atual = self.calcular_hash();

        let dados_para_assinar = format!(
            "{}{}{}{}{:?}",
            self.indice,
            self.hash_anterior,
            self.hash_atual,
            self.timestamp,
            self.transacoes
        );

        let signing_key = SigningKey::<Sha256>::new(
             chave_privada.clone());
        let assinatura = signing_key.sign(
             dados_para_assinar.as_bytes());
        self.assinatura_autoridade = encode(assinatura.
             as_ref());
    }
}
```

Listing 2. Estrutura `Bloco` e assinatura digital com Proof of Authority (PoA).

A estrutura `Bloco` representa os dados básicos de um bloco na blockchain, incluindo o índice, `hash_anterior`, `hash_atual`, timestamp, transações e assinatura da autoridade. A função `assinar_bloco` utiliza uma chave privada RSA para assinar digitalmente os dados do bloco, garantindo a autenticidade e integridade da informação.

3) Funções de Hashing com SHA-256: Para garantir a integridade dos blocos e da cadeia como um todo, utilizando funções de *hash* criptográficas. Cada bloco na blockchain possui um *hash* calculado utilizando o algoritmo SHA-256. O *hash* é gerado a partir de uma representação serializada do bloco, excluindo o campo do *hash* atual para evitar autoreferência. Esse *hash* serve como uma impressão digital única do bloco. Além disso, cada bloco armazena o *hash* do bloco anterior, criando uma ligação entre eles.

Ao recalcular o *hash* de um bloco recebido e compará-lo com o *hash* fornecido, é possível verificar se o bloco não foi alterado. Qualquer modificação nos dados do bloco resultaria em um *hash* diferente, invalidando a cadeia subsequente.

```rust
use sha2::{Digest, Sha256};

impl Bloco {
    pub fn calcular_hash(&self) -> String {
        let mut bloco_clone = self.clone();
        bloco_clone.hash_atual = String::new();
        let bloco_serializado = serde_json::to_string(&
             bloco_clone).unwrap();
        let mut hasher = Sha256::new();
        hasher.update(bloco_serializado.as_bytes());
        let resultado = hasher.finalize();
        format!("{:x}", resultado)
    }
}
```

Listing 3. Função `calcular_hash` para garantir integridade dos blocos.

A função `calcular_hash` calcula o *hash* exclusivo de um bloco, serializando seus dados com a biblioteca `serde_json` e processando-os com o algoritmo `SHA-256`. O resultado é um identificador único que assegura a integridade do bloco.

4) Rede P2P e Comunicação entre Nós: Implementamos uma rede P2P para difusão de transações e blocos, permitindo a sincronização entre os nós participantes. Foi utilizada a biblioteca `libp2p` para estabelecer comunicação entre os nós. Dois protocolos principais são empregados:

- **Gossipsub**: Um protocolo de publicação/assinatura (*pub-sub*) que permite a difusão eficiente de mensagens na rede. É utilizado para transmitir novas transações e blocos entre os nós.
- **Request-Response**: Um protocolo de requisição e resposta que permite a solicitação de dados específicos a um nó. É empregado para solicitar a blockchain completa quando um nó recém-conectado deseja se atualizar.

Ao iniciar, cada nó configura seu identificador e se conecta a peers conhecidos. O nó fica em um loop assíncrono, aguardando eventos da rede, como o recebimento de novas transações ou blocos, e também processando entradas do usuário. Esse design permite que o nó responda rapidamente a eventos da rede enquanto realiza outras tarefas.

5) Persistência e Carregamento da Blockchain: Garantimos a persistência dos dados da *blockchain* localmente, permitindo que os nós retomem suas operações após interrupções. A *blockchain* é serializada em formato JSON e salva em disco. Ao iniciar, o nó tenta carregar a *blockchain* existente a partir do arquivo. Se não for possível (por exemplo, se o arquivo não existir ou estiver corrompido), o nó inicia uma *blockchain* vazia e solicita a versão atualizada da rede.

Essa persistência local permite que os nós mantenham um histórico das transações e blocos sem depender exclusivamente da rede, aumentando a resiliência do sistema.

E. Construção da Blockchain

Com base nas funcionalidades projetadas, realizamos a implementação prática seguindo os requisitos definidos. Para a implementação do registro de Registro de Transações, as transações são coletadas através de um menu interativo no terminal, onde o usuário insere os dados necessários. Após a criação, a transação é adicionada à fila de transações pendentes da *blockchain* local. Em seguida, é difundida para a rede *P2P*, garantindo que todos os nós recebam e armazenem a transação para inclusão em futuros blocos.

Na implementação do algoritmo PoA, os nós autoridades são identificados por um argumento de linha de comando ao iniciar o programa. Esses nós carregam suas chaves privadas *RSA* e possuem a capacidade de criar e assinar novos blocos. Ao criar um bloco, o nó autoridade reúne as transações pendentes, cria o bloco com o *hash* do bloco anterior correto, assina o bloco e o adiciona à sua cadeia local. Em seguida, o bloco é difundido para a rede, onde outros nós validam a assinatura e, se válida, adicionam o bloco à sua cadeia.

Nas funções *hash*, o cálculo do *hash* de cada bloco é essencial para manter a integridade da blockchain. Implementamos uma função que serializa o bloco (excluindo o campo do *hash* atual) e calcula o *hash* SHA-256 desse conteúdo. Ao adicionar um novo bloco, o *hash* é recalculado e armazenado no campo do *hash* atual. Essa prática garante que qualquer alteração nos dados do bloco resulte em um *hash* diferente, permitindo a detecção de manipulações.

Na rede *P2P* e sincronização de *blockchain*, a comunicação entre os nós é gerenciada pelo módulo de rede, que utiliza a biblioteca `libp2p`. Ao iniciar, cada nó tenta se conectar a *peers* conhecidos, estabelecendo uma rede descentralizada de comunicação. Para a difusão de transações e blocos, utiliza-se o protocolo *Gossipsub*, que permite uma propagação eficiente das informações para todos os nós. Quando um nó detecta que sua blockchain está incompleta (por exemplo, ao ser inicializado com uma cadeia vazia), ele utiliza o protocolo *Request-Response* para solicitar a cadeia completa de um peer, garantindo assim que todos os nós mantenham a consistência dos dados.

O trecho de código a seguir ilustra como a *blockchain* é carregada do disco ou, caso ocorra um erro (como a ausência do arquivo), é inicializada vazia e configurada para solicitar a atualização da rede. Essa lógica permite que o sistema se recupere automaticamente e se sincronize com os outros nós.

```
use std::sync::{Arc, Mutex};

let blockchain = match Blockchain::carregar_do_disco("
    blockchain.json") {
    Ok(bc) => Arc::new(Mutex::new(bc)),
    Err(e) => {
        println!("Erro ao carregar a blockchain local:
            {:?}.", e);
        println!("Inicializando uma blockchain vazia e
            solicitando atualiza o da rede...");
        Arc::new(Mutex::new(Blockchain::nova_blockchain())
            )
    }
};
let mut p2p_swarm = iniciar_rede(blockchain.clone()).await
    ;

{
    let bc = blockchain.lock().await;
    if bc.cadeia.len() <= 1 {
        p2p_swarm.solicitar_blockchain();
    }
}
```

Listing 4. Inicialização e sincronização da blockchain com a rede.

Neste código, utiliza-se `Arc<Mutex<...>>` para garantir que a *blockchain* possa ser compartilhada e modificada de forma segura entre múltiplas *threads*. A escolha de `Arc` (contagem de referência atômica) permite que várias *threads* acessem uma única instância da *blockchain*, enquanto `Mutex` protege o acesso a esses dados compartilhados, evitando condições de corrida e consequentemente protegendo a seção crítica. Esse *design* é essencial para um sistema distribuído como uma *blockchain*, onde múltiplas operações de leitura e escrita ocorrem simultaneamente.

F. Execução e Interação com a Blockchain

A Figura 5 apresenta um exemplo de execução da aplicação de *blockchain* acadêmica, onde o usuário interage com o sistema para criar uma transação e difundi-la na rede.

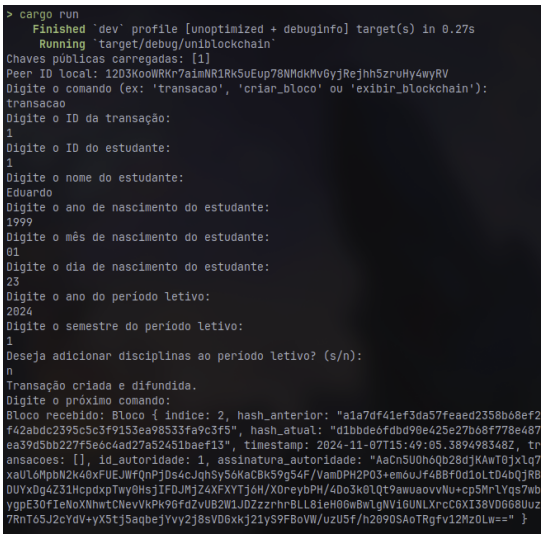

Figura 5. Execução da aplicação de *blockchain* com criação e difusão de uma transação. Terminal CLI, Arch Linux.

Para mais detalhes sobre a execução da blockchain, incluindo um vídeo demonstrativo, consulte o repositório no GitHub[1].

[1]Disponível em: https://github.com/eduardozborowski/uniblockchain

Após compilar o projeto com o comando `cargo run`, o sistema inicializa e carrega as chaves públicas dos nós, permitindo a interação com a rede. O usuário, então, escolhe entre os comandos disponíveis, como `transacao` para criar uma nova transação, `criar_bloco` para gerar um bloco e `exibir_blockchain` para visualizar a cadeia de blocos.

Na execução apresentada, o comando `transacao` foi utilizado. O sistema solicita ao usuário uma série de dados, incluindo o ID da transação, dados do estudante (nome, data de nascimento) e o período letivo. Após inserir essas informações, o sistema cria a transação e a difunde para os nós conectados na rede. Em seguida, o comando `criar_bloco` pode ser utilizado para minerar um bloco com as transações pendentes.

Quando o bloco é recebido pela rede, ele exibe informações detalhadas, como o índice do bloco, o *hash* do bloco anterior, o *hash* atual, o *timestamp*, e a assinatura da autoridade responsável pela validação, como mostra a Figura 5.

Sobre a persistência e carregamento da *blockchain* para assegurar que os dados da blockchain não sejam perdidos entre execuções, implementamos funções para salvar e carregar a cadeia de blocos em um arquivo local. Ao adicionar um novo bloco ou transação, o nó salva o estado atualizado em disco. Ao iniciar, tenta carregar a blockchain existente; se não for possível, inicia uma nova cadeia e solicita a versão atual da rede.

IV. Conclusão e Trabalhos Futuros

O desenvolvimento de uma *blockchain* acadêmica funcional em *Rust* representou não apenas um avanço técnico, mas também um marco significativo na minha trajetória acadêmica e profissional. A implementação de componentes essenciais como registro de transações, validação por Prova de Autoridade (PoA), *hashing* com SHA-256 e comunicação em rede P2P proporcionou um aprofundamento prático nos conceitos de segurança, desempenho e arquitetura de sistemas distribuídos.

O uso de *Rust* ampliou minhas competências em programação segura e eficiente, características fundamentais para projetos críticos como *blockchains*. Além disso, a abordagem prática do projeto, desde o desenvolvimento até a resolução de problemas em múltiplas plataformas, como Linux e Windows, contribuiu para minha capacidade de diagnosticar e solucionar problemas reais, como as limitações impostas por *firewalls* e portas no ambiente Windows. Futuramente, pretende-se expandir essa experiência implementando uma interface gráfica em BunJS e React, o que permitirá transformar o sistema em uma solução mais acessível e intuitiva. Adicionalmente, planeja-se medir o desempenho do sistema em termos de quantidade de transações processadas e realizar testes de segurança abrangentes. Essas iniciativas visam não apenas validar a eficiência e a robustez da *blockchain* desenvolvida, mas também identificar áreas de melhoria e potencial para escalabilidade.

Referências

[1] Dylan Yaga et al. *Blockchain Technology Overview*. 2019. URL: https://doi.org/10.6028/NIST.IR.8202.

[2] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[3] Marc Pilkington. "Blockchain Technology: Principles and Applications". Em: *Research Handbook on Digital Transformations*. Edward Elgar Publishing, 2016, pp. 225–253.

[4] Alexander Grech e Anthony F. Camilleri. *Blockchain in Education*. Publications Office of the European Union, 2017. URL: https://publications.jrc.ec.europa.eu/repository/handle/JRC108255.

[5] Fran Casino, Thomas K. Dasaklis e Constantinos Patsakis. "A systematic literature review of blockchain-based applications: current status, classification and open issues". Em: *Telematics and Informatics* 36 (2019), pp. 55–81.

[6] Nicholas D. Matsakis e Felix S. Klock. "The Rust Language". Em: *ACM SIGAda Ada Letters* 34.3 (2014), pp. 103–104.

[7] Steve Klabnik e Carol Nichols. *The Rust Programming Language*. No Starch Press, 2018.

[8] Christian Cachin. *Blockchain Technology: Concepts and Applications*. 2016. URL: https://www.zurich.ibm.com/dccl/papers/cachin_dccl.pdf.

[9] *Enchentes no Rio Grande do Sul causam prejuízos significativos*. 2023. URL: https://www.jornaldocomercio.com/_conteudo/2023/07/geral/972005-enchentes-no-rs-causam-prejuizos-significativos.html.

[10] *Chuvas no RS: Perdas e Desafios*. 2023. URL: https://g1.globo.com/rs/rio-grande-do-sul/noticia/2023/07/chuvas-no-rs-perdas-e-desafios.ghtml.

[11] Arvind Narayanan et al. *Bitcoin and Cryptocurrency Technologies*. Princeton University Press, 2016.

[12] Dylan Yaga et al. *Blockchain Technology Overview*. Rel. téc. NISTIR 8202. National Institute of Standards e Technology, 2018. URL: https://doi.org/10.6028/NIST.IR.8202.

[13] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. Rel. téc. Federal Information Processing Standards Publication 180-4, 2015. URL: https://doi.org/10.6028/NIST.FIPS.180-4.

[14] V. Zanetta e M. V. S. Oliveira. *Estudo de Aplicação de Blockchain em Processos Industriais*. 2021. URL: https://www.usf.edu.br/galeria/getImage/768/613972367975680.pdf.

[15] Marko Vukolić. “The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication”. Em: *Open Problems in Network Security*. Springer International Publishing, 2015, pp. 112–125.

[16] Stefano De Angelis et al. “Proof-of-Authority: Consensus Model with Identity at Stake”. Em: *Proceedings of the 2017 ACM Conference on Computer and Communications Security*. ACM, 2017, pp. 1–6.

[17] Xiaoqi Xu et al. “A taxonomy of blockchain-based systems for architecture design”. Em: *2017 IEEE International Conference on Software Architecture (ICSA)*. IEEE. 2017, pp. 243–252.

[18] Gavin Wood. *Ethereum: A Secure Decentralised Generalised Transaction Ledger*. Ethereum Project Yellow Paper. 2014. URL: https://ethereum.github.io/yellowpaper/paper.pdf.

[19] Elli Androulaki et al. “Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains”. Em: *Proceedings of the Thirteenth EuroSys Conference*. ACM, 2018, pp. 1–15.

[20] Miguel Castro e Barbara Liskov. “Practical Byzantine Fault Tolerance”. Em: *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation* (1999), pp. 173–186.

[21] Donald Eastlake e Paul Jones. *US Secure Hash Algorithm 1 (SHA1)*. RFC 3174. 2001. DOI: 10.17487/RFC3174. URL: https://rfc-editor.org/rfc/rfc3174.

[22] Ronald L. Rivest, Adi Shamir e Leonard Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. Em: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[23] Tim Dierks e Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2*. RFC 5246. 2008. DOI: 10.17487/RFC5246. URL: https://rfc-editor.org/rfc/rfc5246.

[24] Eric Rescorla. *SSL and TLS: Designing and Building Secure Systems*. Addison-Wesley Professional, 2001.

[25] Carlisle Adams e Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. Addison-Wesley Professional, 1999.

[26] Santosh Chokhani et al. *Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*. RFC 3647. 2003. DOI: 10.17487/RFC3647. URL: https://rfc-editor.org/rfc/rfc3647.

[27] Karim Abouelmehdi, Abdelmajid Beni-Hessane e Hayat Khaloufi. “Blockchain for healthcare data management: Opportunities, challenges, and future recommendations”. Em: *Saudi Journal of Biological Sciences* 25.5 (2018), pp. 1206–1211.

[28] Nick Szabo. “The Idea of Smart Contracts”. Em: *Nick Szabo’s Papers and Concise Tutorials* (1997). URL: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/idea.html.

[29] Konstantinos Christidis e Michael Devetsikiotis. “Blockchains and Smart Contracts for the Internet of Things”. Em: *IEEE Access* 4 (2016), pp. 2292–2303.

[30] Feng Tian. “An Agri-Food Supply Chain Traceability System for China Based on RFID & Blockchain Technology”. Em: *13th International Conference on Service Systems and Service Management (ICSSSM)*. IEEE, 2016, pp. 1–6.

[31] Simona Ibba et al. “CitySense: Blockchain-Oriented Smart Cities”. Em: *Computers* 6.4 (2017), p. 6.

[32] MIT Media Lab. *Digital Diplomas on the Blockchain*. https://www.media.mit.edu/projects/blockchain-diplomas/overview/. 2017.

[33] Sony Global Education. *Sony Develops Blockchain-based Common Database Platform for Education*. https://www.sonyged.com/news/2018/2/13/blockchain-based-educational-platform. 2018.

[34] Stephen R. Palmer e John M. Felsing. *A Practical Guide to Feature-Driven Development*. Prentice Hall Professional, 2002.

[35] RustCrypto. *RustCrypto: Collection of Cryptographic Libraries in Rust*. https://github.com/RustCrypto. 2021.

APÊNDICE: DIAGRAMA DE CLASSES DA BLOCKCHAIN

O diagrama de classes na Figura 6 ilustra a estrutura de dados utilizada na blockchain para o livro de registro acadêmico. Cada entidade no diagrama representa um componente essencial do sistema, e suas relações descrevem como os dados são organizados e conectados.

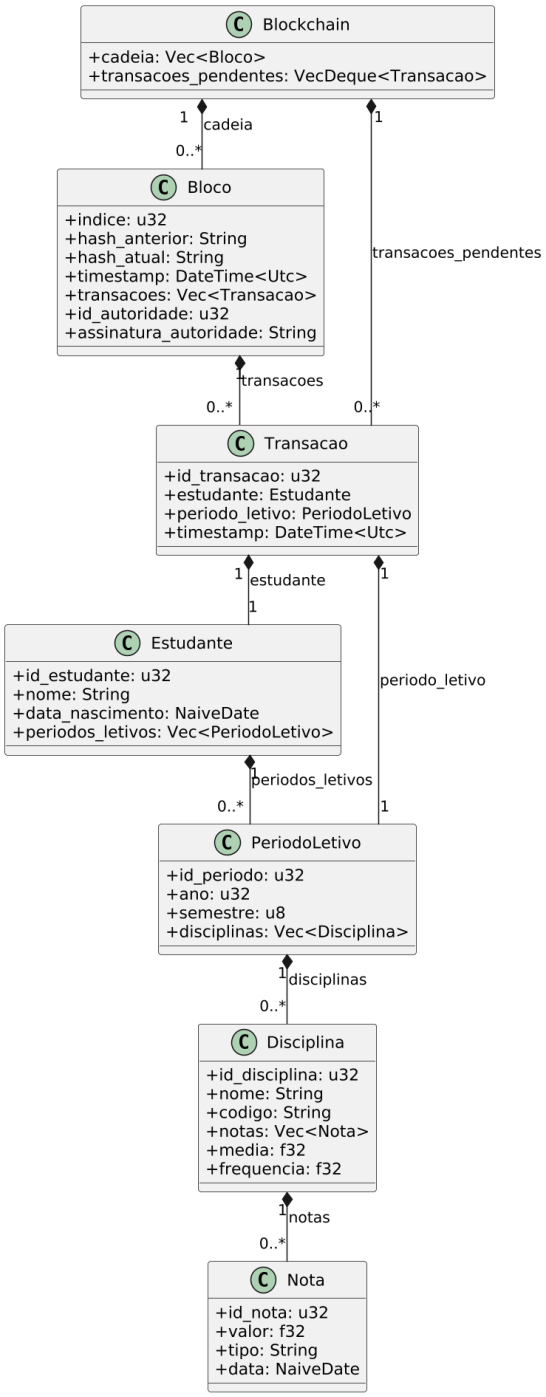

Figura 6. Diagrama de Classes do Livro de Registro Acadêmico na *Blockchain*. Fonte: Elaborado pelo Autor.