UNIVERSIDADE ESTADUAL DA PARAÍBA
CAMPUS VII - PATOS
CENTRO DE CIÊNCIAS EXATAS E SOCIAIS APLICADAS
CURSO DE GRADUAÇÃO EM BACHARELADO EM COMPUTAÇÃO

SAMUEL ALVES MEDEIROS

HYPERDRIVE: UMA API PARA O DARK

PATOS - PB
2024

SAMUEL ALVES MEDEIROS

HYPERDRIVE: UMA API PARA O DARK

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Computação do Centro de Ciências Exatas e Sociais Aplicadas da Universidade Estadual da Paraíba, como requisito parcial à obtenção do título de bacharel em Computação.

Orientador: Dr. Demetrio Gomes Mestre
Coorientador: Dr. Thiago Nóbrega Pereira

PATOS - PB
2024

M488h Medeiros, Samuel Alves.
Hyperdrive [manuscrito] : uma API para o dARK / Samuel Alves Medeiros. - 2024.
49 p. : il. colorido.

Digitado.
Trabalho de Conclusão de Curso (Graduação em Computação) - Universidade Estadual da Paraíba, Centro de Ciências Exatas e Sociais Aplicadas, 2024.
"Orientação : Prof. Dr. Demetrio Gomes Mestre, Coordenação do Curso de Computação - CCEA. "
"Coorientação: Prof. Dr. Thiago Nóbrega Pereira , UEPB - Universidade Estadual da Paraíba "

1. Identificadores Persistentes. 2. Blockchain. 3. Hyperdrive. 4. Padrão REST. 5. Serviço WEB. 6. Imultabilidade. I. Título

21. ed. CDD 005.8

Elaborada por Camile de A. Gomes - CRB - 15/559 COBIB/UEPB

SAMUEL ALVES MEDEIROS

HYPERDRIVE: UMA API PARA O DARK

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Estadual da Paraíba — Campus VII, em cumprimento à exigência para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 05/06/2024

BANCA EXAMINADORA

Demetrio Gomes Mestre
Prof. Dr. Demetrio Gomes Mestre
(Orientador)

Jucelio Soares dos Santos
Prof. Dr. Jucelio Soares dos Santos
(Examinador)

Harllem Alves do Nascimento
Prof. Bel. Harllem Alves do Nascimento
(Examinador)

Dedico este trabalho a todos aqueles que de alguma forma estiveram e estão próximos de mim, fazendo esta vida valer cada vez mais a pena.

AGRADECIMENTOS

Aos meus pais, pela dedicação e amor. Vocês são minha base e minha força para seguir em frente.

Aos meus professores e orientadores, pelo conhecimento transmitido durante toda a graduação e pela orientação ao longo deste trabalho. Os ensinamentos passados foram fundamentais para o meu crescimento acadêmico e profissional.

A todos que de alguma forma estiveram próximos de mim, compartilhando momentos e conhecimento, fazendo com que esta experiência seja gratificante.

“Any fool can write code that a computer can understand. Good programmers write code that humans can understand."

Martin Fowler

RESUMO

O meio acadêmico necessita garantir de maneira eficaz a autoria e a identificação das obras, recorrendo aos Identificadores Persistentes (PIDs) para assegurar a integridade dos recursos, independentemente de sua localização. Tradicionalmente, os sistemas de identificação dependem de modelos centralizados suportados por agências específicas. Este trabalho tem como objetivo desenvolver o Hyperdrive, um serviço web que integra um sistema de identificação descentralizado utilizando o padrão REST. A solução proposta visa atribuir, atualizar e recuperar PIDs para a gestão de publicações científicas, melhorando a integridade dos dados e a imutabilidade dos recursos. A metodologia inclui uma revisão bibliográfica sobre blockchain e PIDs, seguida pelo desenvolvimento e validação do Hyperdrive através de testes de funcionalidade, segurança e desempenho. Os resultados demonstram uma redução significativa no tempo de resposta devido à integração e implementação do REST com o sistema dARK descentralizado, superando as limitações das soluções centralizadas. Conclui-se que o Hyperdrive proporciona uma gestão eficiente de PIDs, assegurando autenticidade e integridade dos dados, além de oferecer uma base sólida para futuras expansões e aprimoramentos.

Palavras-chave: Identificadores Persistentes; Blockchain; Hyperdrive; REST; Serviço WEB; Imultabilidade.

ABSTRACT

The academic field needs to effectively ensure the authorship and identification of works, relying on Persistent Identifiers (PIDs) to ensure the integrity of resources regardless of their location. Traditionally, identification systems depend on centralized models supported by specific agencies. This work aims to develop Hyperdrive, a web service that integrates a decentralized identification system using the REST standard. The proposed solution aims to assign, update, and retrieve PIDs for managing scientific publications, improving data integrity and resource immutability. The methodology includes a literature review on blockchain and PIDs, followed by the development and validation of Hyperdrive through functionality, security, and performance tests. The results demonstrate a significant reduction in response time due to the integration and implementation of REST with the decentralized dARK system, overcoming the limitations of centralized solutions. It is concluded that Hyperdrive provides efficient PID management, ensuring data authenticity and integrity, while also offering a solid foundation for future expansions and enhancements.

Keywords: Persistent Identifiers; Blockchain; Hyperdrive; REST; Web Service; Immutability.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
ARK	*Archival Resource Key*
DOI	*Digital Object Identifier*
dARK	*decentralized Archival Resource Key*
HTTP	*Hypertext Transfer Protocol*
ISSN	*International Standard Serial Number*
JSON	*JavaScript Object Notation*
JWT	*Json Web Token*
ORCID	*Open Researcher and Contributor ID*
P2P	*Peer-to-peer*
PID	*Persistent identifier*
PURL	*Persistent uniform resource locator*
REST	*Representational State Transfer*
URL	*Uniform Resource Locator*

SUMÁRIO

1 INTRODUÇÃO

O meio acadêmico, que constantemente produz e divulga conhecimento por meio de publicações, precisa garantir a autoria e a identificação das obras de maneira eficaz. Para isso, utiliza Identificadores Persistentes (PIDs), que mantêm a integridade dos recursos, independentemente de onde estejam localizados (Sayão, 2007).

O *Digital Object Identifier* (DOI) faz parte de um sistema que possibilita a identificação singular e permanente de uma ampla gama de entidades (sejam elas físicas, digitais ou abstratas) no âmbito da internet, por meio da disponibilização de identificadores digitais específicos para esses objetos (IBICT, 2016). Por outro lado, *Archival Resource Key* (ARK) é um sistema de identificação cujo propósito é prover uma identificação duradoura para os objetos de forma persistente (JUNIOR et al., 2020).

Nesse contexto, emerge o*decentralized Archival Resource Key* (dARK), uma versão descentralizada do ARK. Seu propósito é emancipar os dados de uma estrutura centralizada, permitindo que todos os participantes da rede possam gerenciá-los. A utilização da *blockchain* é fundamental para a descentralização dos dados e para garantir sua imutabilidade (Segundo et al., 2023). A *blockchain* é uma tecnologia que utiliza um registro distribuído e seguro, funciona como um protocolo confiável que possibilita o armazenamento e verificação de informações de maneira transparente e segura, contribuindo assim para a autonomia e integridade dos dados (Pierro, 2017).

Este estudo tem como objetivo desenvolver o Hyperdrive, uma *Application Programming Interface* (API) que estabelece comunicação com o sistema dARK, otimizando a obtenção de Identificadores Persistentes (PIDs) na *blockchain*. Essa otimização busca aumentar a velocidade na obtenção desses recursos, garantir a integridade dos dados acadêmicos e assegurar uma disseminação confiável do conhecimento.

A abordagem descentralizada do sistema dARK surge como uma solução promissora para fortalecer a segurança e a transparência no acesso às informações acadêmicas. A autenticidade das informações é de importância crucial no contexto acadêmico (Sayão, 2007). Esta pesquisa busca evidenciar os benefícios dessa integração, enfatizando seu impacto positivo no avanço da sociedade, ao proporcionar um acesso confiável ao conhecimento.

Este trabalho seguiu uma abordagem de pesquisa aplicada, começando com uma revisão bibliográfica sobre PIDs, sistemas descentralizados, como o dARK, a tecnologia *blockchain*. Posteriormente, foi desenvolvido o Hyperdrive, um serviço web validado por meio de testes de funcionalidade, segurança e desempenho, para avaliar sua eficiência na obtenção de PIDs no sistema dARK.

1.1 Objetivo Geral

O objetivo geral deste estudo é criar/analisar uma API robusta que interaja de maneira eficiente com o dARK, assegurando uma gestão fluida em diversos cenários de requisições e

respostas. Busca-se alcançar um bom tempo de resposta das requisições e aplicar os princípios *Representational State Transfer* (REST). Além disso, o projeto visa implementar métodos eficazes de integração para garantir a proteção de dados e a segurança das comunicações com os demais serviços.

1.2 Objetivos Específicos

Para se alcançar o objetivo geral deste trabalho, foram necessários atingir os seguintes objetivos específicos:

- Realizar uma revisão abrangente sobre os conceitos fundamentais da tecnologia *blockchain*, compreendendo suas características, estrutura, e funcionamento.
- Desenvolver estratégias para aprimorar a integração eficaz com o sistema dARK, considerando tanto requisições síncronas quanto assíncronas, para garantir uma interação fluida e eficiente.
- Buscar uma forma de reduzir o tempo de latência das respostas quando implementado o padrão REST.
- Pesquisar formas de análises abrangentes para avaliar a eficiência e segurança da integração da API Hyperdrive com o sistema dARK, garantindo a eficácia na obtenção de PIDs e a proteção dos dados.

1.3 Justificativa

Segundo Johnson Anthony Watkinson (2018) mais de 3 milhões de artigos científicos são publicados anualmente. A adoção de PIDs é crucial para garantir a autenticidade e integridade dos dados acadêmicos, simplificando sua identificação independentemente da plataforma utilizada. A integração com um serviço REST, aliado aos dados descentralizados e imutáveis do dARK, torna o uso de PIDs altamente vantajoso, proporcionando confiabilidade e eficiência na transmissão desses recursos.

Por meio da integração efetiva com o sistema dARK, conquista-se a confiança de uma estrutura descentralizada, reforçando substancialmente a segurança dos dados. Essa integração almeja uma entrada controlada, por meio de um processo de autenticação e mapeamento detalhado dos dados, o que se configura como um mecanismo crucial para garantir a robustez e a segurança do sistema.

O Hyperdrive atua como um facilitador, tornando a integração com o dARK mais eficaz e segura. Ao agilizar a obtenção de PIDs, Digital Object Identifiers (DOIs) e outros, e (simultaneamente) garantir a segurança na transmissão de dados, ele contribui para uma experiência customizada e confiável no acesso e compartilhamento de recursos.

1.4 Metodologia

Este estudo adota uma abordagem de pesquisa aplicada e descritiva, proporcionando uma investigação aprofundada dos conceitos centrais relacionados à fusão de Identificadores Persistentes (PIDs) e à tecnologia *blockchain*. A revisão conceitual estabelece a base teórica essencial para se compreender os requisitos, além de orientar o desenvolvimento da API e fundamentar a validação da implementação, incluindo os testes.

As atividades realizadas incluem a realização de uma revisão bibliográfica sobre os conceitos de PIDs, DOIs, ARK, dARK, *blockchain* e APIs, além do levantamento dos diagramas de classes, estados e suas respectivas documentações. Essas etapas são fundamentais para garantir uma compreensão abrangente e detalhada dos elementos envolvidos e para assegurar a eficácia da implementação proposta.

1.5 Estrutura do trabalho

Este trabalho apresenta seis capítulos e está organizado da seguinte maneira: no Capítulo 1, é apresentada uma visão geral deste trabalho em relação à contextualização do problema, objetivos, justificava e estrutura do trabalho; no Capítulo 2, são apresentados os conceitos e trabalhos relacionados a esse trabalho de conclusão de curso; no Capítulo 3, é apresentada uma visão geral do sistema e alguns desafios para a realização do mesmo; no Capítulo 4, é apresentado a ferramenta desenvolvida; No Capítulo 5 é demonstrado os resultados obtidos; no último capítulo são apresentadas a conclusão e possibilidades de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Em adição, é vital destacar que este referencial teórico constrói o sólido alicerce que permite a plena compreensão dos requisitos, desafios e possibilidades inerentes à integração de Identificadores Persistentes e tecnologia *blockchain*. Ao explorar a sinergia entre esses conceitos, traçamos as diretrizes essenciais para a concepção e implementação da API Hyperdrive. Essa base teórica fornece a estrutura necessária para orientar o desenvolvimento da API, assegurando sua eficiência, segurança e pertinência no contexto da gestão de Identificadores Persistentes por intermédio do sistema dARK.

2.1 Identificadores Persistentes (PIDs)

Atualmente, a internet se vale de diversos meios para construir identificadores. Um dos mais amplamente adotados pela comunidade é o *Uniform Resource Locator* (URL), que, entretanto, é volátil. Com o tempo, sua utilização pode se tornar problemática devido a vários fatores, como alterações na infraestrutura, atualização das tecnologias ou desatualização do recurso (Sayão, 2007).

Um Identificador Persistente (PID) representa uma referência duradoura associada a um documento, arquivo, página *web* ou outro objeto (Meadows; Haak; Brown, 2019). Mesmo diante de mudanças no recurso ou transferência de propriedade, os links mantêm-se ativos e acessíveis, permitindo o acesso por meio de um navegador *web*.

Hoje em dia, existe uma variedade de identificadores e sistemas para sua atribuição estão disponíveis, podendo ser financiados por empresas privadas ou de acesso aberto ao público. Exemplos incluem URN, *Handle System*, DOI, ARK, *OpenURL*, entre outros.Esses citados acima, são PIDs ou empresas que utilizam identificadores no seu dia a dia.

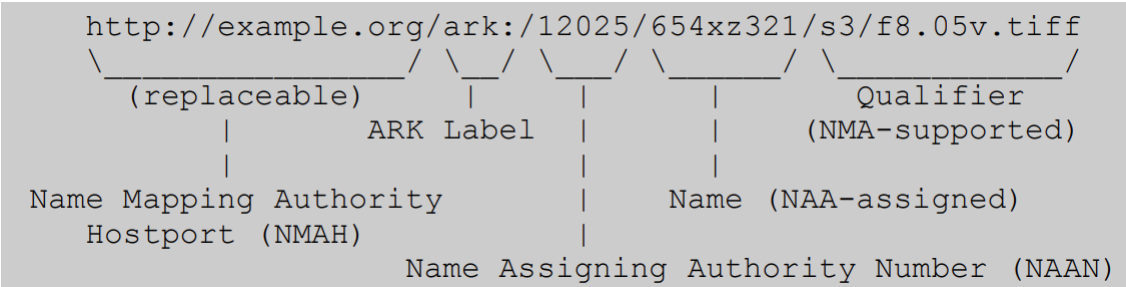
Digital Object Identifier (DOI) é um identificador único para objetos digitais, caracterizado por sua persistência, capacidade de resolução, metadados abrangentes e interoperabilidade semântica (Paskin, 2010). O DOI, comumente associado a agências de registro, está presente no contexto brasileiro principalmente por meio de duas entidades, CrossRef e DataCite, que são plataformas de gerenciamento ou consulta de identificadores. Este identificador possui uma característica única: embora sua estrutura básica permaneça constante, permitindo sua rastreabilidade, é possível atualizar sua localização e outros metadados (Pavão et al., 2018).

O *Archival Resource Key* (ARK) é outro tipo de identificador cujo propósito é conectar objetos a seus respectivos recursos. Ele foi concebido para integrar os pontos positivos de sistemas como URL e DOI, procurando combinar as vantagens dessas duas abordagens. Sua concepção tem como foco a persistência, representando uma alternativa de baixo custo e flexível para a atribuição de identificadores (Kunze; Rodgers, 2008).

A figura 1 ilustra a estrutura de um ARK:

Sua estrutura é sempre iniciada com "ark:/"quando incorporada a uma URL. Nesse caso, o protocolo é seguido pelo nome do serviço que sustenta o ARK. Vale ressaltar que, no ARK, o

Figura 1 – Estrutura do objeto ARK.



Fonte: Elaborada por (Kunze; Rodgers, 2008).

identificador é o único elemento imutável (Kunze; Rodgers, 2008). Antes do próprio nome, pode haver um *Name Mapping Authority Hotsport* (NMAH), que representa um possível endereço para o qual as solicitações e chamadas devem ser direcionadas ao utilizar o ARK (Kunze; Rodgers, 2008).

Pensando na segurança, o ARK adota um *naming scheme* que não apresenta riscos diretos para seu computador ou rede. No entanto, os implementadores devem priorizar medidas de segurança ao fazer requisições, mitigando assim possíveis riscos de *spoofing* e obtenção de informações incorretas (Kunze; Rodgers, 2008).

2.2 Blockchain

Atualmente, a normalização das criptomoedas é uma tendência predominante, acompanhada pelo contínuo avanço tecnológico. Nesse cenário, o mercado financeiro e outros setores são profundamente influenciados por previsões de quedas e avaliações otimistas sobre o futuro desse domínio tecnológico em constante evolução (Al-Saqaf; Seidler, 2017). A concepção da *blockchain* surgiu com a ideia do *bitcoin* por Satoshi Nakamoto, como um meio de pagamento online operado por uma rede P2P (*peer-to-peer*) (Nakamoto, 2008).

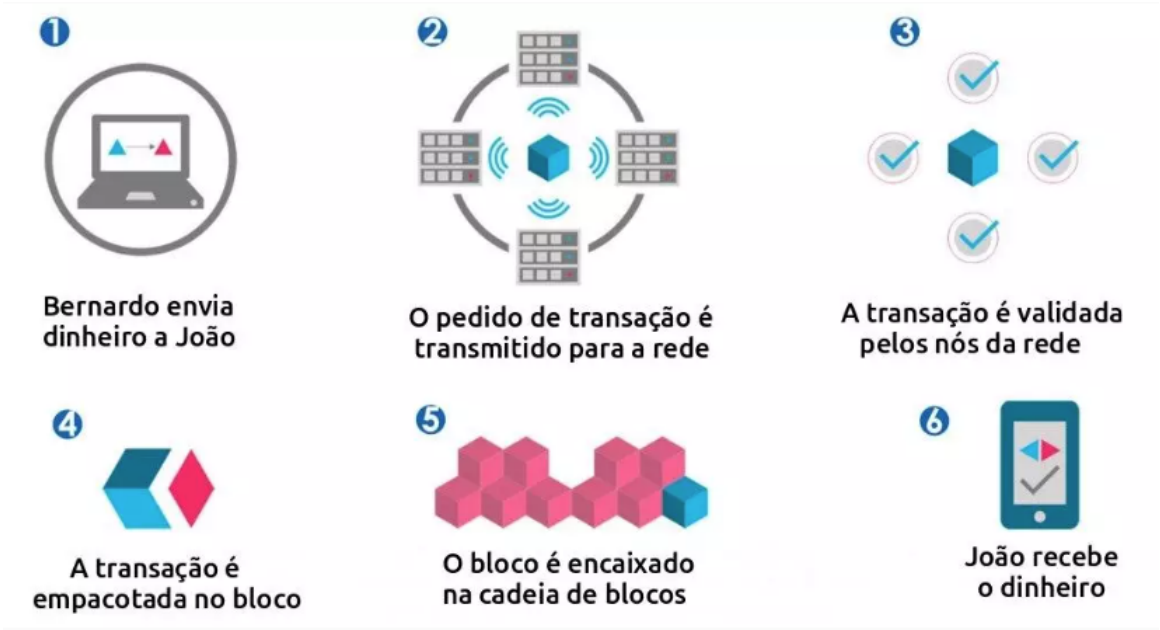
De acordo com Drescher (2018), a *blockchain* é classificada como uma tecnologia inovadora, proporcionando uma forma segura de trocar informações e ativos de valor. Sua característica de imutabilidade nas transações e eliminação de intermediários possibilita o rastreamento da origem das informações.

A *blockchain*, se configurada como pública, faz uso de primitivas criptográficas para assegurar a imutabilidade de suas transações já registradas, impedindo alterações ou exclusões. Essa imutabilidade é possível devido à sua distribuição em um conjunto de nós em uma rede P2P (Pierro, 2017).

Devido à sua estrutura descentralizada, a *blockchain* possibilita que suas transações sejam controladas e verificadas pelos próprios usuários. As transmissões ocorrem entre computadores individuais que atuam como nós na rede, conectando-se para formar uma cadeia de nós interconectados (Romanini; Ohlson, 2018).

A figura 2 representa a estrutura de uma transação na *blockchain*:

Figura 2 – Estrutura de transação na *blockchain*



Fonte: Elaborada por (Marcela Lima, 2020).

A Figura 2 apresenta uma representação do funcionamento de uma transação utilizando a *blockchain*. No cenário, Bernardo deseja realizar uma transação para João. Essa transação é enviada para uma rede P2P composta por vários computadores, representados como nós, interconectados de maneira descentralizada. Após a aprovação das partes, a transação é concluída e, em seguida, imutavelmente adicionada à *blockchain*.

Conforme destacado por Silva (2018), diante da diversidade de ações possíveis utilizando a *blockchain*, essa tecnologia se expandiu além de sua aplicação inicial em transações financeiras, tornando-se uma solução abrangente para a descentralização de diversos setores.

Apesar de sua origem na concepção de criptomoedas, a tecnologia *blockchain* tem revelado sua versatilidade e impacto positivo em inúmeros domínios, estendendo sua influência para setores industriais, serviços financeiros, governamentais, assistência médica e uma ampla gama de outras áreas (Carson et al., 2018). Seu potencial de transformação e suas aplicações em diferentes contextos têm atraído a atenção de organizações e pesquisadores, tornando-a uma tecnologia significativa em constante evolução.

Em contextos de serviços públicos, uma *blockchain* pública se distingue por sua natureza de acesso aberto, que permite a qualquer pessoa visualizar transações e participar no mecanismo de consenso (Alves et al., 2018). Esse grau de transparência e participação torna essa abordagem particularmente relevante para a esfera de serviços públicos, onde a confiabilidade e a integridade dos registros desempenham um papel crítico.

Conforme as análises de (Swan, 2015), a evolução da *blockchain* é categorizada em três principais fases, denominadas como *blockchain* 1.0, 2.0 e 3.0. Cada uma dessas fases representa uma progressão nas capacidades e nos usos da tecnologia *blockchain* ao longo do tempo.

A primeira geração, conhecida como 1.0, se originou com o advento das criptomoedas e

seu foco em transações e sistemas de pagamento. A segunda geração, 2.0, introduziu os contratos inteligentes e a expansão das aplicações descentralizadas, ampliando significativamente suas aplicações em várias áreas. Por fim, a terceira geração, 3.0, visa criar uma sociedade digital com organizações autônomas e descentralizadas, operando com imutabilidade e transparência, eliminando a necessidade de intermediários.

2.3 dARK

O dARK é uma aplicação descentralizada (DApp) que opera em uma rede *blockchain*, integrando-se a um sistema de PID (consulte a Seção 2.1), mais precisamente o ARK (Segundo et al., 2023). Sua proposta é oferecer uma abordagem que permita a diversas instituições gerenciar seus sistemas de identificadores, incluindo a atribuição e reutilização de identificadores persistentes.

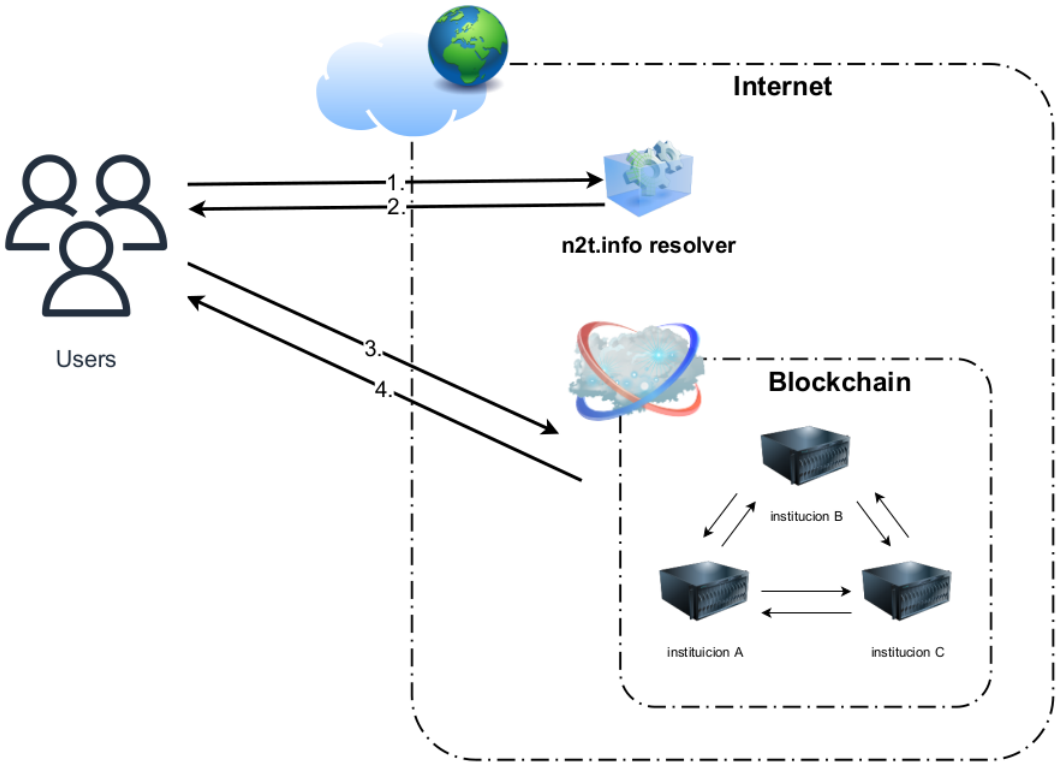
Uma de suas vantagens é a excelente compatibilidade com outros sistemas. É possível recuperar dados diretamente por meio de sua URI dARK ou através das URIs originalmente atribuídas por outros sistemas PIDs. Por exemplo, uma publicação que possui um ARK pode ser recuperada tanto pela URI do dARK quanto pela do ARK.

Conforme mencionado por (Segundo et al., 2023), um dos benefícios de estar presente na *blockchain* é que os dados do PID são replicados de forma segura, auditável e acessível em todos os nós da rede. Isso implica que, mesmo que uma instituição pare de utilizar o sistema por qualquer motivo, os dados permanecerão preservados nos outros nós da rede. Além disso, são empregadas técnicas de preservação e gestão de dados para garantir a integridade e disponibilidade contínua das informações.

A arquitetura do sistema ARK se baseia em um servidor HTTP relativamente simples, o qual consulta uma base de dados local preenchida pelo próprio identificador ARK do recurso em questão. Esse sistema também faz uso de um *resolver* público chamado *Name-to-Thing* (nt2), que direciona as requisições para um servidor *web* específico (Segundo et al., 2023).

No que diz respeito à sua arquitetura, de acordo com a fonte citada por Segundo et al. (2023), pode-se exemplificar seu funcionamento por meio da imagem a seguir:

Figura 3 – Arquitetira do dARK.



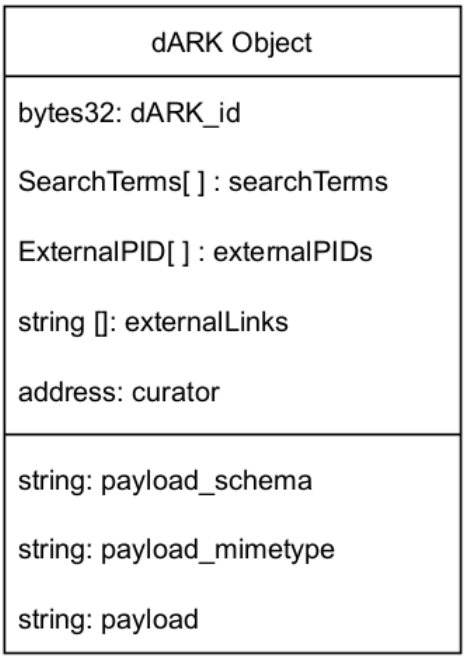
Fonte: Elaborada por (Segundo et al., 2023).

Conforme ilustrado na Figura 3, os usuários do sistema podem fazer uma requisição ao *resolver Name-to-Thing* (N2T), que é responsável por fornecer o endereço da rede *blockchain* associada à instituição do usuário, como demonstrado nos passos 1 e 2. Posteriormente, qualquer nó da rede que receber essa resposta irá processá-la e respondê-la, seguindo os passos 3 e 4.

Com base nessa arquitetura, Segundo et al. (2023) propõe a criação de um identificador mais generalista do que o ARK, adequado às necessidades do sistema dARK. Assim, surge o identificador dARK, que mantém uma estrutura semelhante à do ARK, mas inclui um prefixo destinado a identificar as novas cadeias de caracteres referentes a ele.

É relevante observar que o sistema dARK busca compatibilidade com diversos sistemas de Identificadores Persistentes (PIDs), incluindo, entre outros, o DOI, ORCID, PURL, ISSN, como mencionado na Seção 2.1. Sua estrutura é projetada para fornecer consultas confiáveis sobre os metadados que serão armazenados nele. A Figura 4 apresenta um diagrama simplificado de sua estrutura.

Figura 4 – Objeto padrão dARK.



Fonte: Elaborada por (Segundo et al., 2023).

A decisão de colaborar com o sistema dARK no desenvolvimento do Hyperdrive é respaldada por várias vantagens, incluindo a eficiente capacidade de armazenamento para gerenciar grandes volumes de Identificadores Persistentes (PIDs). O dARK oferece diagnósticos detalhados dos metadados associados aos PIDs, garantindo qualidade e integridade dos dados. Com mecanismos de consulta ágil e nativos de qualidade de dados, o dARK atende às necessidades dinâmicas de usuários. Sua compatibilidade com diversos sistemas de identificadores promove flexibilidade e interoperabilidade. Essas características fundamentam a escolha estratégica de integrar o Hyperdrive ao dARK para uma gestão abrangente e eficiente de PIDs.

2.4 Ferramentas e Plataformas do Projeto

No desenvolvimento de sistemas digitais contemporâneos, a escolha das tecnologias subjacentes é crucial não apenas para o funcionamento do sistema, mas também para sua sustentabilidade, escalabilidade e facilidade de manutenção. Este trabalho apoia-se em três pilares tecnológicos fundamentais: REST, Docker e Git. Cada um desses componentes desempenha um papel essencial na construção de uma infraestrutura robusta e adaptável.

REST, introduzido por Roy Fielding, define um conjunto de princípios arquiteturais para a criação de serviços *web* escaláveis e simples, usando os métodos padrões HTTP para comunicar de forma eficaz e eficiente (Fielding, 2000). Essa abordagem é vital para garantir que a API desenvolvida seja flexível e compatível com diversas plataformas e tecnologias.

Git, conforme descrito por Chacon e Straub (2014), é um sistema de controle de versão distribuído, fundamental para a gestão colaborativa em projetos de *software*. A utilização do Git

permite um controle eficiente de versões, garantindo integridade de dados e suportando fluxos de trabalho complexos. Este sistema é crucial para facilitar a colaboração entre desenvolvedores dispersos geograficamente, tornando-se uma ferramenta indispensável para projetos de *software* em escala global (Chacon; Straub, 2014).

Docker, destacado por Merkel et al. (2014), representa uma inovação significativa na virtualização em nível de sistema operacional. A plataforma facilita a contêinerização de aplicações, garantindo consistência entre os ambientes de desenvolvimento e produção. Esta tecnologia simplifica os processos de desenvolvimento e implantação, ao mesmo tempo que minimiza problemas de incompatibilidade entre diferentes ambientes operacionais Merkel et al. (2014).

Python, uma linguagem de programação de alto nível, é conhecida por sua facilidade de aprendizado e por sua sintaxe clara e objetiva, o que a torna uma escolha popular entre desenvolvedores de todos os níveis de experiência. Projetada com o foco na legibilidade do código, resultando em programas que são visualmente limpos e logicamente projetados segundo seu criador (Rossum; Drake et al., 1995). Além disso, Python oferece uma vasta biblioteca padrão e uma ampla gama de *frameworks* e ferramentas, facilitando a integração com outras plataformas e tecnologias, como Git e Docker.

3 HYPERDRIVE

Neste capítulo, é apresentado em detalhes o Hyperdrive, a ferramenta que foi desenvolvida uma versão inicial, com o objetivo aprimorar a integração com o sistema dARK e otimizar a obtenção de PIDs na *blockchain*. É fornecida uma descrição sobre sua arquitetura, o funcionamento e as principais características do Hyperdrive, destacando que sua implementação está em andamento.

Ressalta-se que ao longo desta seção, os possíveis resultados são abordados, destacando como essa ferramenta pode impactar positivamente a integridade e a disseminação confiável do conhecimento acadêmico e afins. São abordados os benefícios e contribuições que se espera alcançar com o Hyperdrive no contexto da pesquisa acadêmica, bem como sua capacidade de proporcionar uma experiência customizada.

Por fim, este capítulo estabelece as bases para compreender o funcionamento do Hyperdrive e o impacto que sua implementação pode ter no avanço do acesso confiável ao conhecimento acadêmico. Além disso, é importante destacar a contribuição coletiva para este projeto.

3.1 Introdução à Implementação

O Hyperdrive, uma *Application Persistence Interface* (API), que desempenha o papel de intermediário na comunicação entre o lado do cliente e o serviço da *blockchain* com o uso dARK. Este projeto é desenvolvido em colaboração com a Rede Nacional de Pesquisa (RNP)e o Instituto Brasileiro de Informação em Ciência e Tecnologia (IBICT). Desenvolvido em equipe, a implementação da API tem enfoque principal otimizar o processo de obtenção de PIDs na *blockchain*.

No âmbito do objetivo de gerenciar registros e consultas de publicações acadêmicas por meio de seus PIDs, cada publicação é atribuída a um identificador único. O Hyperdrive, intermediará juntamente ao uso de identificadores as capacidades de armazenar, modificar e consultar esses recursos de forma eficiente e segura. Isso representa um passo significativo na otimização do acesso, na manutenção da integridade dos dados acadêmicos e na promoção da disseminação confiável de conhecimento. Está comprometido em fornecer uma solução robusta e acessível para atender às necessidades dos usuários na comunidade acadêmica.

No sistema, cada publicação ou artigo é associado a um identificador. Utilizando essa identificação e o sistema dARK, é possível transformá-la em um ARK ou um *hash*. Esses dados representam informações valiosas essenciais para diversos métodos do sistema. Ao gerar um identificador ARK ou seu *hash*, é possível associá-lo a três atributos essenciais.

- ***External URL***: é possível vincular o identificador a uma URL externa, permitindo a fácil referência ou acesso a recursos associados.
- ***External PID***: Esse parâmetro permite a vinculação a PIDs externos, ampliando a interoperabilidade do sistema com outras infraestruturas.

- ***Payload***: O campo de *payload* servirá para fornecer uma descrição detalhada do objeto ou conter informações valiosas relacionadas ao identificador.

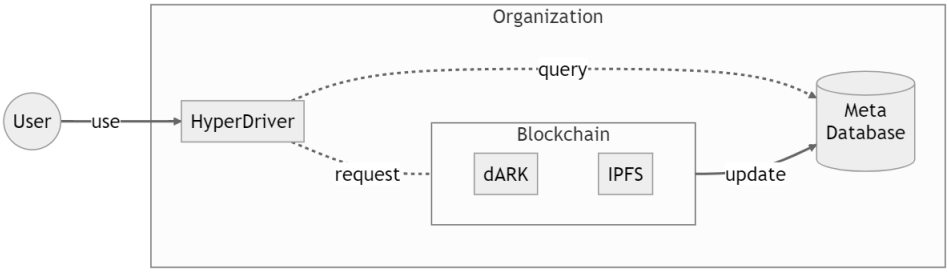
Prosseguindo o desenvolvimento do projeto, avançamos com a implementação do módulo central (*core module*). Este módulo inclui três *endpoints* principais. Cada um desses endpoints desempenha funções específicas no sistema, contribuindo para a funcionalidade geral da API.

3.2 Arquitetura do Hyperdrive

A API HyperDrive foi desenvolvida com base nos princípios do REST, um modelo amplamente reconhecido por sua escalabilidade, simplicidade e flexibilidade em sistemas distribuídos (Fielding, 2000). A implementação utilizando o framework Flask, escrito em Python, foi escolhida por sua capacidade de desenvolver rapidamente aplicações *web* com um conjunto mínimo de ferramentas, propiciando agilidade e eficiência no processo de desenvolvimento conforme enfatizado por (Grinberg, 2018).

A principal função desta API é servir como mediadora entre os usuários finais e a infraestrutura da *blockchain*, facilitando a adoção de PIDs descentralizados através do dARK. Estes identificadores desempenham um papel crucial na asseguração da integridade e permanência dos dados, o que é particularmente relevante no contexto de aplicações descentralizadas, uma abordagem em crescente evidência nos estudos mais recentes sobre sistemas distribuídos e *blockchain* (Nakamoto, 2008).

Figura 5 – Diagrama de interação usuário com Hyperdrive.

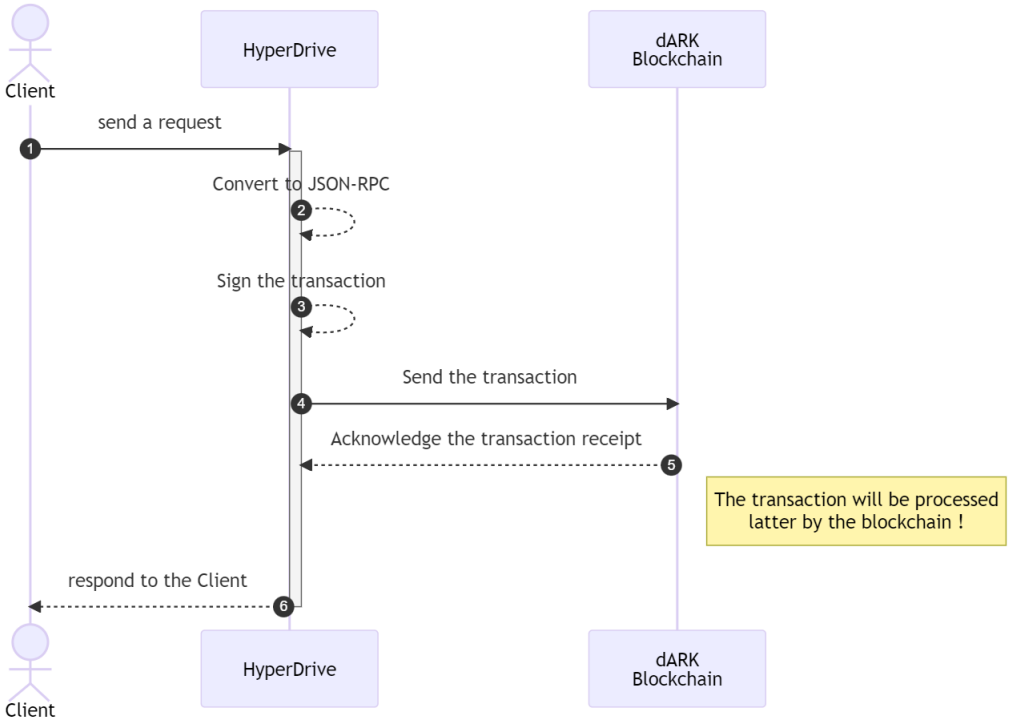


Fonte: Elaborada por grupo de desenvolvimento.

Na figura 5, é mostrada a interação do usuário com a API HyperDrive. As solicitações são feitas usando comandos URL, destacando a flexibilidade e a compatibilidade em diversos ambientes de desenvolvimento. O Hyperdrive processa essas solicitações e age como intermediário entre o usuário e a blockchain, utilizando dARK e IPFS. Além disso, realiza atualizações em um banco de dados de metadados.

Esta representação visual facilita o entendimento de como o HyperDrive integra-se à *blockchain* e ao sistema de arquivos interplanetários, assegurando a robustez e a resiliência do sistema.

Figura 6 – Diagrama de sequência do Hyperdrive.



Fonte: Elaborada por grupo de desenvolvimento.

O reconhecimento da recepção da transação é capturado, e a resposta é então enviada de volta ao cliente. Esse processo demonstra o compromisso do Hyperdrive com a transparência e segurança das operações realizadas. A Figura 6 ilustra o ciclo de vida de uma transação no HyperDrive. Inicia-se com o envio de uma requisição pelo cliente, que é então convertida para o formato JSON-RPC dentro da API. O próximo passo é a assinatura da transação, essencial para a segurança e autenticidade, seguida pelo envio desta transação para a *blockchain* dARK.

Ainda que a integração com um banco de dados robusto não esteja presente na versão inicial da API, o uso do PostgreSQL está previsto para as próximas iterações. Tal melhoria visa não só a otimização na gestão dos dados, mas também a conexão com sistemas de gerenciamento de bancos de dados mais robustos e escaláveis.

3.3 Tecnologias Utilizadas

No contexto do desenvolvimento do projeto Hyperdrive, a seleção criteriosa das tecnologias é fundamental para garantir não apenas a funcionalidade, mas também a escalabilidade e a segurança da aplicação. Neste sentido, optou-se pela linguagem de programação Python, conhecida por sua versatilidade e eficiência em diversos paradigmas de programação. Além disso, a estruturação das rotas foi realizada através do *framework Hypertext Transfer Protocol* (HTTP) *Flask*, que oferece um ambiente propício para o desenvolvimento ágil e flexível de aplicações *web*. Esta seção detalha as tecnologias principais empregadas no projeto, enfatizando sua aplicabilidade e as razões de sua escolha, embasadas por literatura acadêmica relevante e

experiências práticas consolidadas. A segui a listagem das principais tecnologias utilizadas no projeto.

- **Python**: Framework Python leve e flexível para o desenvolvimento de aplicações *web*.
- **Flask**: Empregado na implementação das rotas da API, Flask proporciona um ambiente de desenvolvimento ágil e é altamente eficaz para prototipagem rápida de aplicações *web* (Grinberg, 2018).
- **cURL**: Utilizado para a transferência segura de informações entre partes na *web*, cURL é fundamental para autenticação e autorização de usuários em aplicações modernas. A biblioteca suporta diversas opções de protocolos de rede, o que a torna versátil para operações de rede (Stenberg, 2017).
- **Docker**: A plataforma Docker é empregada para o desenvolvimento, envio e execução de aplicativos em contêineres. Docker facilita a implantação da API e garante a consistência do ambiente de execução através da virtualização leve (Merkel et al., 2014).
- **Git**:Como um sistema de controle de versão distribuído, Git é crucial para o rastreamento e gerenciamento de mudanças no código-fonte. Sua capacidade de suportar fluxos de trabalho distribuídos é altamente valorizada em projetos colaborativos de *software* (Chacon; Straub, 2014).

A escolha dessas tecnologias foi cuidadosamente feita para atender às necessidades específicas do Hyperdrive. Cada ferramenta foi selecionada para maximizar a eficiência operacional, facilitar a manutenção e garantir a segurança do sistema. O uso de Python e Flask é especialmente alinhado com as práticas modernas de desenvolvimento ágil e responsivo, enquanto Docker e Git fornecem soluções robustas para gerenciamento de infraestrutura e controle de versão. Essa configuração tecnológica estabelece uma base sólida para o sucesso contínuo e a escalabilidade do projeto, posicionando-o na vanguarda das práticas contemporâneas de desenvolvimento de *software*.

3.4 Descrição dos Endpoints

Os endpoints são pontos de acesso fundamentais em qualquer API, determinando as operações disponíveis para os usuários. No contexto do projeto, cada endpoint oferece funcionalidades específicas para interagir com o sistema dARK e seus PIDs. A seguir, listamos as principais funcionalidades oferecidas pelos endpoints do Hyperdrive.

- **Get: Recuperar um PID**: Este endpoint recebe como parâmetro de rota um dARK ID para buscar um recurso com o mesmo identificador e assim retorná-lo em formato JSON. Caso o recurso associado ao dARK ID não seja encontrado, o endpoint retornará um erro.

- **SET: Definir atributos**: O endpoint SET é projetado para permitir a configuração de uma URL externa *(external_url)* ou de um *payload*. A operação específica a ser realizada seja a alteração de uma URL ou de um *payload* é determinada por meio do corpo da requisição, passando uma flag *"external_url"* ou *"payload"*. Essas operações podem ser realizadas tanto de maneira síncrona quanto assíncrona, o que é definido por meio de uma variável de ambiente.
- **ADD: Adicionar atributos**: O *endpoint ADD* é projetado para permitir a adição de atributos a um PID existente. Isso pode incluir uma URL externa *(external_url)* ou um PID externo *(external_pid)*. A operação específica a ser realizada é determinada pelo corpo da requisição, seja uma URL externa ou um PID externo, identificados por uma flag *"external_pid"* ou *"external_url"*.Essas operações podem ser realizadas tanto de maneira síncrona quanto assíncrona, o que é definido por meio de uma variável de ambiente.

Para as operações de SET e ADD, o usuário deve fornecer como parâmetro um PID ou um *hash* para a identificação do recurso. Além disso, o usuário deve enviar uma string informando qual campo deseja alterar. Em formato JSON, o usuário deve especificar o parâmetro que deseja modificar. Mostrando em imagens os tipos de parâmetros que podem receber e o que deve ser passado no corpo da requisição.

Figura 7 – Parâmetro por *hash* do ARK.

Fonte: Elaborada pelo Autor.

Figura 8 – Parâmetro por ARK ID.

Fonte: Elaborada pelo Autor.

Figura 9 – Corpo da requisição.

Fonte: Elaborada pelo Autor.

Vale ressaltar que, quando um PID é gerado, ele é inicialmente considerado como um rascunho (*draft*). Para que o PID não seja mais tratado dessa forma, a primeira operação a ser executada deve ser ou ADD ou SET em relação ao campo *"external_url"*. Isso garante que o

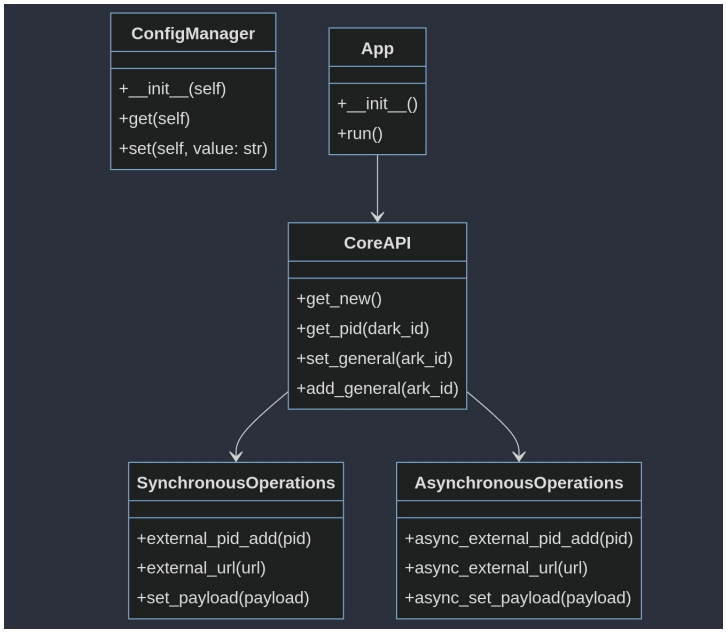
PID seja vinculado a algum recurso (desde que este recurso seja válido), tornando-o oficial e não mais um rascunho. Essa abordagem visa assegurar que os PIDs gerados estejam sempre associados a recursos válidos, contribuindo para a integridade e eficácia do sistema.

Com sua premissa descentralizada e a integração com a *blockchain*, o sistema oferece a capacidade de recuperar dados de forma eficiente, permitindo que os usuários realizem chamadas de forma síncrona ou assíncrona, dependendo de suas necessidades. As chamadas síncronas garantem que os dados estejam disponíveis com certeza no momento da resposta, embora possam levar mais tempo para serem completadas. Por outro lado, as chamadas assíncronas proporcionam flexibilidade e permitem que os usuários continuem com outras operações enquanto esperam pela resposta, otimizando assim o uso do tempo e dos recursos do sistema. Esta flexibilidade oferecida pelo Hyperdrive permite que os usuários otimizem suas interações com o sistema, escolhendo a abordagem mais adequada para diferentes cenários.

Por meio de variáveis de ambiente será configurado se as chamadas serão síncronas ou assíncrona, oferecendo flexibilidade na definição das operações e validações a serem realizadas. Isso permitirá que os usuários personalizem a execução da API de acordo com suas necessidades específicas, adaptando-a para atender aos requisitos de seus projetos ou aplicações.

A seguir, na Figura 10, ilustramos um diagrama simples de classe exemplificando os métodos do Hyperdrive, temos uma visão geral dos principais elementos e funcionalidades da API.

Figura 10 – Diagrama de classe.



Fonte: Elaborada pelo Autor.

No Hyperdrive, a arquitetura foi projetada para suportar totalmente requisições via cURL, uma biblioteca amplamente reconhecida no ambiente acadêmico e editorial pela eficiência em fazer solicitações HTTP. Essa compatibilidade completa com cURL é estratégica, pois facilita a integração do Hyperdrive com sistemas já existentes em universidades e periódicos científicos.

Isso promove uma interface de usuário coesa e eficiente, alinhada com as práticas predominantes nessas instituições.

Considerando os desafios enfrentados no desenvolvimento do Hyperdrive, é crucial conceber uma API robusta e otimizada. A expectativa é que a API não apenas suporte a integração com PIDs, mas também aproveite as vantagens da tecnologia *blockchain* para assegurar a integridade e a confiabilidade dos dados. A estrutura de classes, como ilustrada na Figura 10, revela uma arquitetura que contempla a execução de operações síncronas e assíncronas, refletindo a multifuncionalidade exigida por um sistema de tal calibre.

A classe "CoreAPI"é central no sistema, coordenando as operações e interagindo diretamente com as subclasses "*SynchronousOperations*"e "*AsynchronousOperations*". As operações síncronas são usadas quando é crucial garantir execução imediata e feedback rápido, enquanto as operações assíncronas são importantes para maximizar a eficiência, permitindo a execução paralela sem bloquear a *thread* principal. Dessa forma, a infraestrutura planejada para o Hyperdrive, com base na sólida "*CoreAPI*", está bem preparada para a próxima fase de implementação. Isso será fundamental para alcançar os resultados desejados e garantir que o sistema seja viável em aplicações do mundo real.

4 DEMONSTRAÇÃO DA FERRAMENTA HYPERDRIVE

Este capítulo oferece uma análise detalhada sobre o que foi desenvolvido do Hyperdrive até então, enfocando sua implementação, configuração e funcionalidades. Inicialmente, aborda-se a implementação do módulo core, que constitui a espinha dorsal da ferramenta, explorando sua arquitetura inicial e as principais funcionalidades. Esta seção é fundamental para compreender como a API foi construída desde sua concepção.

Segue-se com a gestão de variáveis de ambiente, discutindo-se as estratégias adotadas para a configuração dessas variáveis e como elas influenciam a execução da API em diferentes ambientes. Este aspecto é crucial para garantir a segurança e a adaptabilidade da ferramenta. Posteriormente, examinam-se os padrões de mensagens utilizados, tanto de erro quanto de sucesso, detalhando como esses padrões contribuem para uma comunicação eficaz e uma experiência de usuário consistente. A padronização das mensagens é vital para a manutenção e escalabilidade.

A análise contínua com a descrição das funcionalidades e operações disponíveis nos endpoints da aplicação. Cada endpoint é analisado em termos de seus métodos, parâmetros e respostas, proporcionando-se exemplos concretos de uso para ilustrar como o Hyperdrive pode ser efetivamente integrada e utilizada.

Finalmente, a análise das respostas da API sob diferentes cenários de testes é apresentada, evidenciando a robustez e confiabilidade da ferramenta. Esta seção destaca os resultados obtidos e discute como eles validam as capacidades técnicas da aplicação. Portanto, este capítulo não apenas demonstra as funcionalidades técnicas, mas também enfatiza sua aplicabilidade prática, permitindo aos leitores uma compreensão clara de seu impacto e relevância no campo de aplicação específico.

4.1 Implementação do módulo Core

O módulo *Core* do HyperDrive desempenha um papel vital no ecossistema do dARK, atuando como o principal facilitador de operações relacionadas a Identificadores Persistentes Digitais (PID). Este módulo é meticulosamente projetado para lidar com o registro, a atualização e a manutenção de metadados associados aos PIDs. Sua capacidade de integrar-se suavemente com a *blockchain* o torna um componente essencial para a gestão de identificadores, a espinha dorsal de qualquer sistema de armazenamento e recuperação de dados digitais.

A resolução de PID é uma das funcionalidades críticas providas pelo HyperDrive. Quando um PID é requisitado ou consultado, o módulo entra em ação, processando a solicitação e devolvendo as informações relevantes ao usuário. Em circunstâncias onde os objetos digitais mudam de localização, o HyperDrive assegura que o acesso aos recursos não seja interrompido, redirecionando os usuários para as novas localizações sem esforço. Isso é imperativo para manter a consistência e acessibilidade dos dados em um ambiente dinâmico, onde os ativos digitais podem ser frequentemente movimentados ou atualizados.

Além disso, o módulo *Core* possibilita que os usuários registrem e gerenciem PIDs

de forma programática por meio de suas APIs. Essas interfaces permitem que as aplicações automatizem o processo de atribuição de PIDs a objetos digitais, associem metadados essenciais e atualizem os registros de PID conforme necessário. Esse gerenciamento programático é essencial para o ciclo de vida dos PIDs, abrangendo eventos como a desativação ou reatribuição, garantindo que a integridade dos dados seja preservada ao longo do tempo.

Essencialmente, o módulo *Core* do HyperDrive é construído sobre a premissa de eficiência e confiabilidade. Ao abstrair a complexidade inerente à interação com a *blockchain* e ao fornecer um sistema coeso para o gerenciamento de identificadores persistentes, o módulo eleva a experiência do desenvolvedor e a funcionalidade do usuário final. Com esse suporte robusto, o HyperDrive não só simplifica as operações críticas relacionadas aos PIDs, mas também se estabelece como um pilar de interoperabilidade e sustentação dentro do ambiente descentralizado do dARK.

4.2 Gestão de variáveis de ambiente

O sistema utiliza várias variáveis de ambiente para gerenciar sua configuração e validar as operações realizadas. Essas variáveis são essenciais para garantir que o sistema funcione de maneira eficiente e segura, proporcionando flexibilidade na gestão das operações e na integração com outros sistemas. Abaixo, detalham-se as principais variáveis de ambiente utilizadas no HyperDrive e como elas influenciam o comportamento do sistema:

- **HYPERDRIVE EXTERNAL PID VALIDATION**: Esta variável determina o método de validação para os Identificadores de Processo Externo (PID). Os valores possíveis incluem *"NONE"*, indicando que nenhuma validação será realizada, e *"BASIC"*, que ativa uma validação simples baseada em formatos pré-definidos. A escolha do método de validação influencia diretamente na segurança e integridade dos dados processados, assegurando que apenas PIDs válidos sejam aceitos e processados pelo sistema.

- **HYPERDRIVE URL VALIDATION**: Controla como as URLs fornecidas ao sistema são validadas. Semelhante à validação de PID, admite os valores *"NONE"* e *"BASIC"*. A validação *"BASIC"* geralmente envolve a verificação da URL contra uma expressão regular específica, garantindo que apenas URLs bem formadas sejam aceitas. Esta variável é crucial para prevenir erros de operação e ataques cibernéticos que podem ser iniciados através de URLs malformadas ou maliciosas.

- **HYPERDRIVE PAYLOAD VALIDATION**: Especifica o método de validação para os payloads submetidos ao sistema. Os valores *"NONE"* e *"BASIC"* estão disponíveis, onde *"BASIC"* implica uma verificação de que o payload está em um formato JSON *(JavaScript Object Notation)* válido. Esta validação é vital para manter a consistência dos dados e a estabilidade operacional, evitando erros de processamento causados por dados incorretamente formatados.

- **HYPERDRIVE OPERATION MODE**: Define o modo de operação do sistema, podendo ser *"SYNC"* (síncrono) ou *"ASYNC"* (assíncrono). O modo assíncrono permite que o sistema execute operações sem bloquear outras atividades, melhorando o desempenho geral e a escalabilidade. Esta variável é fundamental para configurar como as operações são gerenciadas internamente, afetando diretamente a eficiência e a resposta do sistema a cargas de trabalho variadas.
- **HYPERDRIVE AUTH**: Gerencia a autenticação para métodos que necessitam proteção por meio de tokens de acesso. Os valores *"TRUE"* e *"NONE"* controlam se a autenticação está ativada ou desativada, respectivamente. Quando ativada, essa variável garante que apenas usuários autorizados possam executar operações críticas, reforçando a segurança do sistema.

4.2.1 Detallhamento das validações no sistema

Com base nas variáveis citadas acima, será ou não realizada uma série de validações essenciais para o controle de qualidade e segurança no processamento de dados dentro da aplicação. É importante destacar que, sendo estas validações parte do primeiro módulo de implementação, elas são relativamente simples. Estas validações iniciais são configuradas para garantir que tanto as entradas quanto as operações dentro do sistema atendam a padrões específicos, mitigando possíveis riscos e assegurando a integridade dos dados manipulados. A seguir, detalha-se cada tipo de validação implementada, descrevendo como elas influenciam diretamente na operacionalidade e segurança do sistema, proporcionando uma base sólida para futuras expansões e aprimoramentos.

A validação de URL, regulada pela respectiva variável de ambiente, determina como as URLs submetidas ao sistema são verificadas. Sob a configuração *"BASIC"*, esse processo se dá por meio de uma expressão regular (*regex*) projetada para assegurar que as URLs correspondam a um formato específico. O padrão *regex* empregado é:

Figura 11 – *Regex* de validação URL.

```
regex = ("((http|https)://)(www.)?" +
         "[a-zA-Z0-9@:%._\\+~#?&//=]" +
         "{2,256}\\.[a-z]" +
         "{2,6}\\b([-a-zA-Z0-9@:%" +
         "._\\+~#?&//=]*)")
```

Fonte: Elaborada pelo Autor.

A Figura 11 exibe o código da expressão regular utilizada para a validação de URLs. Este padrão *regex* assegura que as URLs atendam aos critérios de formatação especificados, conferindo sua estrutura adequada antes de serem processadas pelo sistema.

- Começa com "http"ou "https".
- Pode incluir "www.".
- Segue com caracteres permitidos para nomes de domínio e caminhos.
- Termina com um domínio de topo de 2 a 6 letras.

Sobre a validação de identificadores persistentes digitais (PID), determina como os Identificadores de Processo Externo (PID) são validados. Com a configuração "BASIC", espera-se que o PID siga o formato:

Figura 12 – Formato PID.

Fonte: Elaborada pelo Autor.

Por exemplo, um PID válido poderia ser "doi:/10.1016/j.datak.2023.102180", onde "doi:/"é o protocolo , indicando que o sistema de identificação usado é o DOI, já a outra parte posterior "10.1016/j.datak.2023.102180"é o identificador específico. Esta validação assegura que apenas PIDs que sigam o padrão estabelecido sejam aceitos, mantendo a integridade dos dados e a consistência do sistema.

Para a validação do *payload*, a opção *"BASIC"* envolve a verificação da validade do formato JSON dos dados submetidos. Esta validação é crucial para garantir que o sistema processe apenas *payloads* que estejam corretamente formatados como JSON, evitando problemas de interpretação de dados ou falhas na operação. O JSON válido deve ser bem estruturado, conforme exemplificado na imagem abaixo:

Figura 13 – JSON válido.

Fonte: Elaborada pelo Autor.

4.3 Padrões de mensagens

As mensagens são um componente crucial para entender as comunicações da API, sejam elas indicativas de sucesso ou de erro. Esta seção explora os padrões de resposta adotados nesta

versão inicial da API. É importante observar que alguns parâmetros podem ser opcionais nas mensagens, surgindo apenas se forem efetivamente utilizados em uma determinada solicitação. Essa característica de design flexível permite que as respostas da API se adaptem mais especificamente às necessidades da operação realizada.

Figura 14 – Padrão de mensagens.

Fonte: Elaborada pelo Autor.

- **pid**: Representa um identificador único para um objeto específico na API, retornado como *string*. Este parâmetro é essencial para rastrear e manipular objetos dentro do sistema HyperDrive.
- **pid hash index**: Armazena o índice interno da *blockchain* do PID, proporcionando uma maneira de verificar e localizar o PID dentro da *blockchain*.
- **hyperdrive operation mode**: Indica o modo operacional do HyperDrive, podendo ser "sync"para operações sincronizadas ou *"async"* para operações assíncronas. Este parâmetro define como as ações são processadas pela API.
- **action**: Especifica a ação solicitada pela API, como *"set_payload"*, *"new_pid"*, *"add_url"*, e *"add_external_pid"*. Cada opção representa uma tarefa diferente que pode ser executada através da API.
- **parameters**: Inclui informações adicionais necessárias para a ação solicitada. É formatado em JSON e pode conter valores como ’pid’, ’external_url’, e ’payload’, dependendo da ação.
- **status**: Reflete o estado da solicitação à API, com valores como "executed", "queued", ou "rejected", indicando, respectivamente, execução bem-sucedida, aguardando processamento ou erro na solicitação.

- **transaction hash**: No modo assíncrono, este parâmetro contém o hash da transação na *blockchain*, uma informação vital para rastrear a transação.
- **error code**: Disponível apenas se ocorrer um erro, esse parâmetro fornece o código do erro, seguindo as boas práticas do REST.
- **error message**: Disponível apenas se ocorrer um erro, esse parâmetro fornece a mensagem de erro, seguindo as boas práticas do *REST*, informar a mensagem é essencial para o usuário entender melhor do que se trata o erro.

As mensagens na API são cruciais para a compreensão das interações, com a opcionalidade de certos parâmetros nas respostas, que aparecem baseadas na sua utilização em solicitações específicas. Essa abordagem permite que as respostas sejam mais adaptáveis e pertinentes ao contexto da operação. A compreensão desses padrões de resposta é vital para a eficiência na utilização da API do HyperDrive. Este detalhamento não só clarifica as expectativas durante as interações com a API, mas também garante que os usuários possam manejar adequadamente as respostas em diferentes cenários, contribuindo para uma integração mais eficaz e um diagnóstico preciso de problemas.

4.4 Funcionalidades e Operações

Esta seção detalha a execução prática dos endpoints da API, explicando como cada um opera de forma síncrona ou assíncrona e como as respostas são geradas em diferentes cenários. Isso inclui exemplos de sucesso e de erro para cada operação, proporcionando uma compreensão abrangente da robustez e confiabilidade da API.

4.4.1 Endpoints Síncronos

Endpoints síncronos esperam a conclusão da operação requisitada antes de enviar uma resposta ao cliente.

4.4.1.1 GET: Recuperar um PID

Descrição: Este endpoint é usado para recuperar um PID associado a um dARK ID específico.

Cabeçalho da requisição: `GET /core/get/{dark_id}`

Resposta de Sucesso:

```
{
    "pid": "examplePID123",
    "op_mode": "sync",
    "status": "executed",
    "action": "get_pid",
    "parameter": "{dark_id}"
}
```

Resposta de Erro:

```
{
    "error_message": "block_chain_error : {str(e)}",
    "error_code": 400,
    "op_mode": "sync",
    "status": "rejected",
    "action": "get_pid",
    "parameter": "{dark_id}"
}
```

4.4.1.2 SET: Definir uma URL externa

Descrição: Este endpoint é usado para adicionar uma URL externa ao sistema de forma assíncrona.

Cabeçalho da requisição: `POST /core/set/{ark_id}`

Parâmetros:

```
{
  "external_url": "http://example.com"
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "sync",
    "status": "executed",
    "action": "set_url",
    "parameter": {"external_url": "http://example.com"},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "sync",
    "status": "rejected",
    "action": "set_url",
    "parameter": {"external_url": },
}
```

4.4.1.3 SET: Definir um payload

Descrição: Este endpoint é usado para adicionar um payload ao sistema de forma assíncrona.

Cabeçalho da requisição: `POST /core/set/{ark_id}`

Parâmetros:

```
{
  "payload": {"example" : "content" }
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "sync",
    "status": "executed",
    "action": "set_payload",
    "parameter": {"payload": {"example" : "content" }},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "sync",
    "status": "rejected",
    "action": "set_payload",
    "parameter": {"": {"example" : "content" }},
}
```

4.4.1.4 ADD: Adicionar uma URL

Descrição: Este endpoint é usado para adicionar uma URL externa ao sistema.

Cabeçalho da requisição: `POST /core/add/{ark_id}`
Parâmetros:

```
{
  "external_url": "http://google.com"
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "sync",
    "status": "executed",
    "action": "add_url",
    "parameter": {"external_url": "http://google.com"},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "sync",
    "status": "rejected",
    "action": "add_url",
    "parameter": {"external_url": "x"},
}
```

4.4.1.5 ADD: Adicionar uma PID externo

Descrição: Este endpoint é usado para registrar um novo PID no sistema.
Cabeçalho da requisição: `POST /core/add/{ark_id}`
Parâmetros:

```
{
    "external\_pid": "newPID123"
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "sync",
    "status": "executed",
    "action": "add_pid",
    "parameter": {"external_pid": "newPID123"},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "sync",
    "status": "rejected",
    "action": "add_pid",
    "parameter": {"external_pid": },
}
```

4.4.2 Endpoints Assíncronos

Endpoints assíncronos iniciam uma operação e retornam uma resposta imediata, permitindo que a operação continue sendo processada em segundo plano. Para entrar no modo assíncrono deve mudar a variável de ambiente.

4.4.2.1 SET: Definir uma URL externa

Descrição: Este endpoint é usado para adicionar uma URL externa ao sistema de forma assíncrona.

Cabeçalho da requisição: `POST /core/set/{ark_id}`

Parâmetros:

```
{
  "external_url": "http://example.com"
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "async",
    "status": "executed",
    "action": "set_url",
    "parameter": {"external_url": "http://example.com"},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "async",
    "status": "rejected",
    "action": "set_url",
    "parameter": {"external_url": },
}
```

4.4.2.2 SET: Definir um payload

Descrição: Este endpoint é usado para adicionar um payload ao sistema de forma assíncrona.

Cabeçalho da requisição: `POST /core/set/{ark_id}`

Parâmetros:

```
{
  "payload": {"example" : "content" }
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "async",
    "status": "executed",
    "action": "set_payload",
    "parameter": {"payload": {"example" : "content" }},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "async",
    "status": "rejected",
    "action": "set_payload",
    "parameter": {"": {"example" : "content" }},
}
```

4.4.2.3 ADD: Adicionar uma URL

Descrição: Este endpoint é usado para adicionar uma URL externa ao sistema.

Cabeçalho da requisição: `POST /core/add/{ark_id}`
Parâmetros:

```
{
  "external_url": "http://google.com"
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "async",
    "status": "executed",
    "action": "add_url",
    "parameter": {"external_url": "http://google.com"},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "async",
    "status": "rejected",
    "action": "add_url",
    "parameter": {"external_url": "x"},
}
```

4.4.2.4 ADD: Adicionar uma PID externo

Descrição: Este endpoint é usado para registrar um novo PID no sistema.
Cabeçalho da requisição: `POST /core/add/{ark_id}`
Parâmetros:

```
{
    "external\_pid": "newPID123"
}
```

Resposta de Sucesso:

```
{
    "pid": "{dark_id}",
    "op_mode": "async",
    "status": "executed",
    "action": "add_pid",
    "parameter": {"external_pid": "newPID123"},
}
```

Resposta de Erro:

```
{
    "message": "{error_message}",
    "error_code": "400 | 404 | 500 | 501",
    "op_mode": "async",
    "status": "rejected",
    "action": "add_pid",
    "parameter": {"external_pid": },
}
```

4.5 Análise de respostas

A avaliação das respostas geradas pelos endpoints de uma API é crucial para assegurar sua funcionalidade e confiabilidade. Esta seção dedica-se a uma análise meticulosa das respostas retornadas pelos diversos endpoints da API, com o objetivo de verificar sua aderência aos requisitos especificados e sua capacidade de manejar corretamente tanto as situações de sucesso quanto as de falha. Através desta análise, busca-se compreender o comportamento da API sob diferentes condições de operação, proporcionando insights valiosos sobre a robustez do sistema e identificando possíveis áreas para aprimoramento.

4.5.1 Análise de Respostas de Sucesso

O propósito desta análise é assegurar que a API esteja cumprindo suas funções de maneira eficaz, retornando dados precisos e completos, conforme o esperado. A integridade dos dados, a correspondência de conteúdo e o tempo de resposta são as principais métricas avaliadas.

- **Integridade dos Dados:** Verifica-se se os dados retornados estão completos e intactos.
- **Correspondência de Conteúdo:** Confirma-se se os dados retornados correspondem exatamente aos solicitados.
- **Tempo de Resposta:** Mede-se o tempo necessário para que a API responda à requisição.

Observa-se que cada *endpoint* retornou respostas adequadamente estruturadas em JSON, indicando que a serialização dos dados está funcionando conforme esperado. Os tempos de resposta mostraram-se, em geral, adequados, mas variáveis conforme a carga do servidor e o volume de dados processados.

4.5.2 Análise de Respostas de Erro

O propósito desta seção é entender como a API lida com entradas inválidas ou condições anormais, o que é essencial para a robustez do sistema. As métricas focadas incluem a adequação das mensagens de erro e o uso apropriado dos códigos de status HTTP.

- **Adequação das Mensagens de Erro:** Avalia-se se as mensagens de erro são claras e fornecem informações suficientes para entender o motivo da falha.
- **Código de Status HTTP:** Verifica-se se os códigos de status correspondem adequadamente às diferentes situações de erro.

As mensagens de erro demonstraram ser específicas e informativas, esclarecendo as causas exatas das falhas nas requisições. Por exemplo, uma falha foi relatada como *"Blockchain is down"*, refletindo um problema na infraestrutura e não na lógica da aplicação. Os códigos de status HTTP foram utilizados de maneira apropriada, como 404 para "PID not found"e 500 para falhas internas do servidor.

A análise sugere que a API está configurada de maneira eficiente para lidar com cenários de sucesso e de falha. As respostas são consistentes e os códigos de erro estão alinhados com as práticas padrão do protocolo HTTP. Recomenda-se, contudo, a implementação de monitoramento contínuo das respostas para identificar e resolver prontamente qualquer inconsistência ou degradação no desempenho.

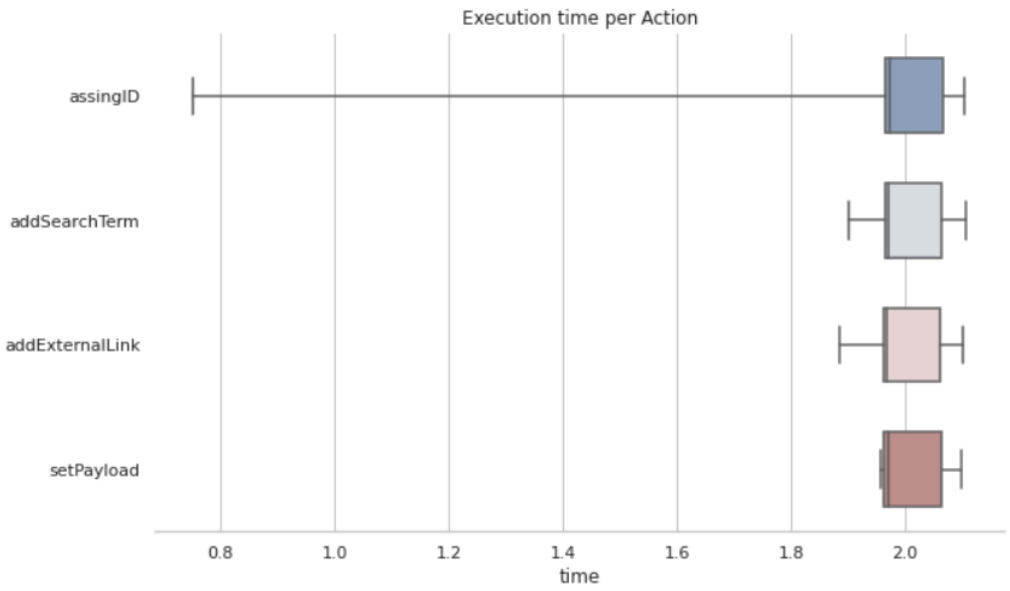
5 RESULTADOS

O desenvolvimento do módulo core do projeto Hyperdrive foi concluído com sucesso, demonstrando a viabilidade e eficácia da solução proposta. O *web service* desenvolvido, denominado Hyperdrive, cumpriu integralmente sua função como ponte de comunicação entre o dARK e os sistemas de repositórios utilizados por instituições de ensino e pesquisa. Este serviço mostrou-se eficiente ao encapsular as funções on-chain do dARK em mensagens REST, conforme proposto no objetivo específico A. Isso permitiu uma integração mais simplificada e transparente entre as partes envolvidas, garantindo uma comunicação fluida e eficaz. Todo o código fonte pode ser encontrado no repositório https://github.com/dark-pid/hyperdrive.

5.1 Evolução do Tempo de Resposta do dARK

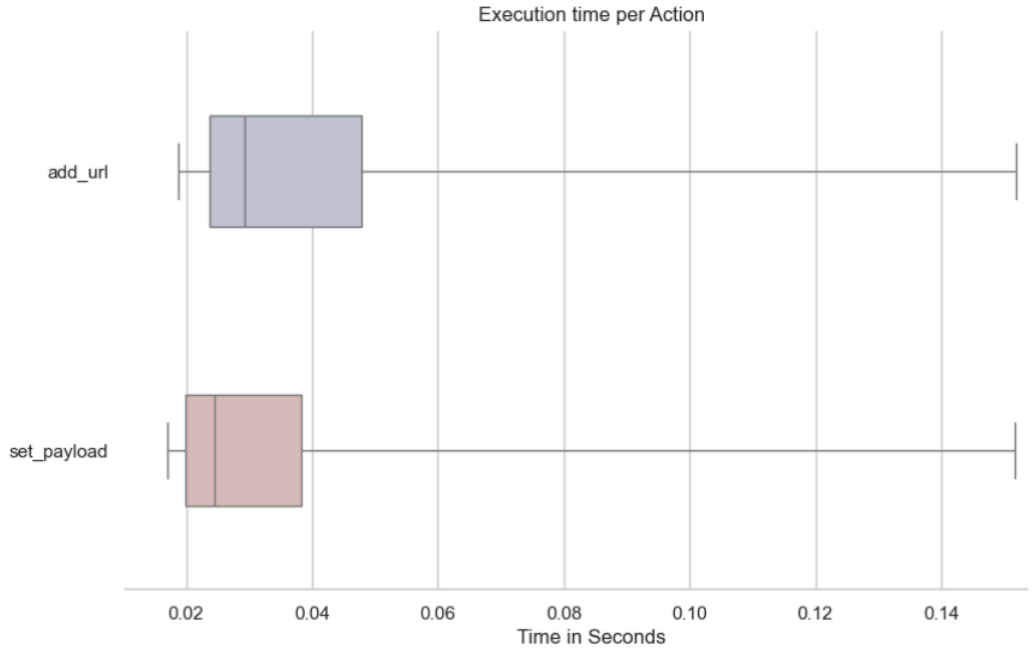
A aliança das contribuições do Hyperdrive com iniciativas financiadas pelo IBICT e LA Referencia resultou em uma redução significativa na latência das operações realizadas por contratos inteligentes na blockchain, conforme estabelecido no objetivo do trabalho. As figuras abaixo ilustram os resultados.

Figura 15 – Tempo de resposta das ações em 2022 (sem melhorias).



Fonte: Elaborada por grupo de desenvolvimento.

Figura 16 – Tempo de resposta das ações em 2023/2024 (após melhorias).



Fonte: Elaborada por grupo de desenvolvimento.

Os gráficos comparativos evidenciam a melhoria significativa no desempenho das ações após a implementação da API REST. Antes das melhorias, as ações "assignID", "addSearchTerm", "addExternalLink"e "setPayload"apresentavam variações significativas nos tempos de execução, especialmente na ação "assignID", que registrava o maior tempo de execução médio.

Após a implementação, observou-se uma redução notável nos tempos de execução das ações, com especial destaque para add_url e set_payload, que apresentaram melhorias substanciais. Esta redução pode ser atribuída à eficiência da API REST, que otimiza a comunicação entre os componentes do sistema. No entanto, é importante ressaltar que o tempo de resposta ainda poderá ser melhorado à medida que o desenvolvimento do projeto avança e novos módulos são implementados.

5.2 Impacto das alterações

É importante destacar que a função "addSearchTerm"foi removida do dARK e que a operação "assignID"foi fundida com "add_url". Essas alterações resultaram em uma atribuição mais rápida e eficiente de identificadores persistentes para objetos, contribuindo para uma melhor experiência do usuário final e otimizando o fluxo de trabalho das instituições envolvidas.

Com as alterações implementadas, a velocidade de execução das operações melhorou significativamente. Anteriormente, a ação "assignID"tinha um tempo de resposta médio de aproximadamente 2 segundos. Após a fusão com "add_url", o tempo de resposta caiu para cerca de 0,05 segundos. Isso representa uma melhoria de quase 40 vezes na velocidade de execução.

Além disso, a remoção da função "addSearchTerm"eliminou um gargalo significativo no processo, permitindo uma execução mais fluida e rápida das outras operações. O tempo de resposta da função "addExternalLink"também foi otimizado, passando de uma média de 1,8 segundos para aproximadamente 0,04 segundos. Estas melhorias evidenciam a eficácia das alterações no código e a eficiência do novo módulo core do Hyperdrive.

6 CONCLUSÃO E TRABALHOS FUTUROS

No cenário acadêmico, onde a produção e disseminação do conhecimento são fundamentais, a gestão eficaz de Identificadores Persistentes (PIDs) desempenha um papel crucial. A enorme quantidade de pesquisas e publicações exige uma infraestrutura robusta e eficiente, capaz de lidar com a complexidade inerente a essa vasta rede de informações. Nesse contexto, o desenvolvimento do Hyperdrive, integrando a tecnologia *blockchain*, surge como uma resposta inovadora e estratégica para atender às crescentes demandas desse ecossistema dinâmico.

A implementação inicial do Hyperdrive, centrada na versão *core*, estabelece uma base sólida para a gestão de PIDs, mas ainda há um vasto potencial para expansão e aprimoramento. Este trabalho focou na criação de uma interface robusta e confiável para operações básicas com PIDs, utilizando tecnologias como o *framework* Flask e a linguagem Python, que provaram ser decisivas para um desenvolvimento ágil e eficiente. Além disso, a tecnologia *blockchain* foi escolhida devido à sua capacidade de proporcionar transparência, segurança e descentralização, qualidades essenciais para a gestão de dados acadêmicos.

A integração da *blockchain* no ambiente acadêmico oferece várias vantagens significativas, incluindo a imutabilidade dos registros, que é crucial para a preservação da integridade acadêmica. Além disso, a descentralização proporcionada pela *blockchain* facilita a colaboração e o compartilhamento de dados entre instituições sem comprometer a segurança ou a privacidade. Esta tecnologia promove também uma maior transparência nos processos acadêmicos, possibilitando um novo paradigma para o rastreamento e a verificação da autenticidade das publicações científicas.

Apesar dos avanços, a implementação de funcionalidades adicionais como um sistema de banco de dados robusto e autenticação detalhada estão previstas para módulos futuros. Essa abordagem modular permite uma concentração na estabilidade e eficácia da versão *core* antes de adicionar camadas adicionais de complexidade, como integração com sistemas de gestão de banco de dados mais sofisticados e métodos de autenticação segura.

À medida que a pesquisa acadêmica continua a expandir-se, a evolução contínua do Hyperdrive promete desempenhar um papel central na facilitação da identificação, recuperação e integração de recursos acadêmicos. A perspectiva de colaboração mais estreita com instituições acadêmicas e editoras pode abrir novas oportunidades para aprimoramentos e customizações específicas ao domínio, aumentando a interoperabilidade com outras plataformas de gestão de PIDs.

Este trabalho não apenas responde às necessidades imediatas da comunidade acadêmica, mas também lança as bases para um futuro em que a gestão de Identificadores Persistentes se torna cada vez mais integrada, eficiente e adaptável aos desafios em constante evolução da pesquisa. O Hyperdrive não é apenas uma solução tecnológica, mas uma contribuição significativa para a progressão do ecossistema acadêmico em direção a uma era mais conectada e acessível.

A implementação futura deve considerar a adição de análises de desempenho em tempo

real, suporte para operações em grande escala e uma interface mais intuitiva para os usuários finais. Além disso, explorar parcerias estratégicas com outras iniciativas tecnológicas e acadêmicas pode potencializar o impacto e a eficácia do Hyperdrive, estendendo seu alcance e funcionalidade.

Em conclusão, a Hyperdrive está bem-posicionada para uma expansão significativa que poderá explorar plenamente o potencial das tecnologias descentralizadas em aplicações de armazenamento e recuperação de dados acadêmicos. O caminho adiante é rico em oportunidades para inovação e melhoria, prometendo transformar os processos acadêmicos e melhorar a integridade, acessibilidade e confiabilidade dos dados acadêmicos em uma escala global.

REFERÊNCIAS

AL-SAQAF, W.; SEIDLER, N. Blockchain technology for social impact: opportunities and challenges ahead. *Journal of Cyber Policy*, Taylor & Francis, v. 2, n. 3, p. 338–354, 2017. Citado na página 16.

ALVES, P. H. et al. Desmistificando blockchain: conceitos e aplicações. *Computação e Sociedade", Sociedade Brasileira de Computação. Disponível em: http://www-di. inf. puc-rio. br/˜ kalinowski//publications/AlvesLNRLK20. pdf*, 2018. Citado na página 17.

CARSON, B. et al. Blockchain beyond the hype: What is the strategic business value. *McKinsey & Company*, v. 1, p. 1–13, 2018. Citado na página 17.

CHACON, S.; STRAUB, B. *Pro git*. [S.l.]: Springer Nature, 2014. Citado 3 vezes nas páginas 20, 21 e 25.

DRESCHER, D. *Blockchain Básico: uma introdução não técnica em 25 passos*. [S.l.]: Novatec Editora, 2018. Citado na página 16.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine, 2000. Citado 2 vezes nas páginas 20 e 23.

GRINBERG, M. *Flask web development*. [S.l.]: "O'Reilly Media, Inc.", 2018. Citado 2 vezes nas páginas 23 e 25.

IBICT, E. Atribuição de identificadores digitais para publicações científicas: Doi para o seer/ojs. *Boletim Técnico do PPEC*, v. 1, n. 1, p. 39p–39p, 2016. Citado na página 12.

JOHNSON ANTHONY WATKINSON, M. M. R. The stm report: An overview of scientific and scholarly journal publishing. p. 214, 2018. Citado na página 13.

JUNIOR, R. F. G. et al. Acesso aberto a dados de pesquisa no brasil: identificadores persistentes para dados de pesquisa. 2020. Citado na página 12.

KUNZE, J.; RODGERS, R. The ark identifier scheme. 2008. Citado 2 vezes nas páginas 15 e 16.

Marcela Lima. *Blockchain: O Que É? Como Funciona Essa Tecnologia?* 2020. Disponível em: https://criptofy.com/blockchain-o-que-e/. Acesso em: 13 de outubro de 2023. Citado na página 17.

MEADOWS, A.; HAAK, L. L.; BROWN, J. Persistent identifiers: the building blocks of the research information infrastructure. *Insights*, United Kingdom Serials Group (UKSG), v. 32, p. 1–6, 2019. Citado na página 15.

MERKEL, D. et al. Docker: lightweight linux containers for consistent development and deployment. *Linux j*, v. 239, n. 2, p. 2, 2014. Citado 2 vezes nas páginas 21 e 25.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008. Citado 2 vezes nas páginas 16 e 23.

PASKIN, N. Digital object identifier (doi®) system. *Encyclopedia of library and information sciences*, Citeseer, v. 3, p. 1586–1592, 2010. Citado na página 15.

PAVÃO, C. M. G. et al. Acesso aberto a dados de pesquisa no brasil: práticas e percepções dos pesquisadores: relatório 2018. Universidade Federal do Rio Grande do Sul, 2018. Citado na página 15.

PIERRO, M. D. What is the blockchain? *Computing in Science & Engineering*, IEEE, v. 19, n. 5, p. 92–95, 2017. Citado 2 vezes nas páginas 12 e 16.

ROMANINI, A. V.; OHLSON, M. P. De elos bem fechados: o pragmatismo e a semiótica peirceana como fundamentos para a tecnologia blockchain utilizada no combate às fake news. *Volume 18-Edição 2 2º Semestre de 2018 ISSN 1676-3475*, v. 15354, p. 107, 2018. Citado na página 16.

ROSSUM, G. V.; DRAKE, F. L. et al. *Python reference manual*. [S.l.]: Centrum voor Wiskunde en Informatica Amsterdam, 1995. v. 111. Citado na página 21.

SAYÃO, L. F. Interoperabilidade das bibliotecas digitais: o papel dos sistemas de identificadores persistentes-urn, purl, doi, handle system, crossref e openurl. *Transinformação*, SciELO Brasil, v. 19, p. 65–82, 2007. Citado 2 vezes nas páginas 12 e 15.

SEGUNDO, W. et al. *dARK: A decentralized blockchain implementation of ARK Persistent Identifiers*. [S.l.], 2023. Citado 4 vezes nas páginas 12, 18, 19 e 20.

SILVA, C. C. d. Blockchain: um estudo da descrentralização da tecnologia da computação na quarta revolução industrial e seu impacto sócio-ambiental. Universidade Federal Fluminense, 2018. Citado na página 17.

STENBERG, D. Everything curl. *GitBook: Lyon, France*, 2017. Citado na página 25.

SWAN, M. *Blockchain: Blueprint for a new economy*. [S.l.]: "O'Reilly Media, Inc.", 2015. Citado na página 17.