
UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
CENTRO MULTIDISCIPLINAR DE PAU DOS FERROS
CURSO DE BACHARELADO EM TECNOLOGIA DA INFORMAÇÃO

GERSON BRENO FAGUNDES DE OLIVEIRA

# PROJETO DE UMA ARQUITETURA PARA A FERRAMENTA LOOP ACADEMIC: MÓDULO ALUNO

PAU DOS FERROS - RN
2024

GERSON BRENO FAGUNDES DE OLIVEIRA

# PROJETO DE UMA ARQUITETURA PARA A FERRAMENTA LOOP ACADEMIC: MÓDULO ALUNO

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Bacharel em Tecnologia da Informação.

Orientador: Prof. Dr. Reudismam Rolim de Sousa

PAU DOS FERROS - RN

2024

48p Oliveira, Gerson Breno Fagundes de.
Projeto de uma arquitetura para a ferramenta loop academic: módulo aluno / Gerson Breno Fagundes de Oliveira. - 2024.
51 f. : il.

Orientador: Reudismam Rolim de Sousa.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2024.

1. Back-end. 2. Front-end. 3. Api. 4. Algoritmo. I. Sousa, Reudismam Rolim de , orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

GERSON BRENO FAGUNDES DE OLIVEIRA

**PROJETO DE UMA ARQUITETURA PARA A FERRAMENTA LOOP ACADEMIC: MÓDULO ALUNO**

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Bacharel em Tecnologia da Informação.

Defendida em: **18/10/2024**.

**BANCA EXAMINADORA**

_____
Prof. Dr. Reudismam Rolim de Sousa, Prof. Dr. - (UFERSA)
Presidente

_____
Profa. Dra. Jarbele Cássia da Silva Coutinho, Profa. Dra. - (UFERSA)
Membro Examinador

_____
Prof. Me. José Ferdinandy Silva Chagas, Prof. Me. - (UFERSA)
Membro Examinador

*Dedico este trabalho à minha família, especialmente à minha mãe, Silva Cleide, e às minhas irmãs, Maria Lucina e Gessica Juliana. Elas nunca duvidaram da minha capacidade, sempre me apoiaram e estiveram ao meu lado nas fases mais desafiadoras desta jornada e continuam a fazer isso.*

## AGRADECIMENTOS

Primeiramente gostaria de agradecer a Deus, a principal base de princípios ao qual cresci e fui ensinado a respeitar e clamar em momentos de dificuldade; agradecer por ser o primeiro de minha família a se formar em uma universidade federal a uma longa distância de casa e que desde a minha chegada eu nunca me senti só de verdade.

De maneira a faltar palavras, sou grato pelos meus pais, que desde o início nunca disseram nenhuma palavra de desânimo ou fraqueza; a minha mãe Silva Cleide, que diante de dificuldades financeiras, problemas de saúde, distância e situações diversas mostraram que eu tenho a base forte fundamentada no verdadeiro valor de uma família. Foram nesses momentos meus conselheiros, cuidadores e sempre me motivaram a seguir em frente.

Agradeço ao meu irmão que sempre esteve disposto a me ajudar sem pedir nada em troca, sempre me apoiando em tudo que eu precisei, seja me dando conselhos.

Agradeço aqui, especialmente, a meus tios que sempre me deram suporte com transporte para que eu pudesse ir para faculdade sem pedirem nada em troca. Logo, sou muito grato a eles: Silvaci Fagundes e Antônio Silvanio.

Agora neste agradecimento não conterá laço de sangue, mas um forte laço de amizade, respeito, companheirismo, que cresceu em meio a dificuldades e desafios, mas que até hoje nos deu frutos; aqui cito Iandra Liz, Iris Matias, Maria Eduarda, Angelo Silvano, Jade Dafine e Luiz. Todos eles de maneira direta fizeram por mim coisas que nem todo amigo, colega ou parceiro faz; cada um do seu jeito, do seu próprio modo e com suas próprias capacidades; esses sim “cavaram trincheiras ao meu lado”.

Agradeço aos meus professores, mas tem alguns que o meu agradecimento transcende os muros da universidade, primeiramente ao Prof. Me. Ferdinandy e o Prof.Dr. Reudismam, seus exemplos como pessoas, carisma e dedicação serão lembrados e ficarão gravados na minha essência de universitário.

Agradeço a todo corpo docente da UFERSA campus Pau dos Ferros. Obrigado!

Agradeço ao meu orientador, o Prof.Dr. Reudismam, que sempre esteve à disposição para me ouvir e me orientar em cada passo dado dentro da universidade, e me guiou para a entrega deste trabalho de conclusão de curso. Por fim, agradeço a minha banca examinadora. Obrigado!

## RESUMO

Ao adentrar no meio acadêmico em cursos voltadas a Tecnologia da Informação (TI), alguns ingressantes costumam ter dificuldades com componentes curriculares básicos de programação, a exemplo de Algoritmos. Para amenizar essas dificuldades, abordagens são propostas, a exemplo da plataforma Loop Academic. Atualmente, dentro do ciclo da Engenharia de Software, essa plataforma se encontra no estágio de prototipação (*design* da interface de usuário). Dessa forma, o ambiente precisa ser implementado para que possa ser efetivamente utilizado por estudantes ingressantes no ensino superior. Neste trabalho, é desenvolvida uma arquitetura de software para essa ferramenta. O desenvolvimento da arquitetura foi baseada na proposta de divisão da arquitetura em diferentes visões proposta por Bass et al. (2012), que propõe três categorias de apresentação de artefatos arquiteturais, a visão de módulo, componente & conector e implantação. As três visões foram projetadas para o software, permitindo que se tenha diferentes perspectivas do ambiente a ser desenvolvido. Para avaliar a proposta de arquitetura foram desenvolvidas duas abordagens; a primeira delas baseada em cenários de uso da plataforma e a segunda voltada ao desenvolvimento dos artefatos de software gerados em linguagem de programação, voltada a parte *back-end* da aplicação, que pode ser integrada à parte *front-end* da aplicação. Como resultados, foi possível identificar que a arquitetura pode ser implantada e integrada a integrada a um ambiente externo, o *front-end* da aplicação. Como contribuições, a implantação do Loop Academic permite que essa ferramenta promissora possa atuar para reduzir as taxas de reprovação na disciplina, especialmente, entre estudantes ingressantes, proporcionando uma prática mais contínua e facilitando o entendimento dos conceitos.



**Palavras-chave:** back-end; front-end; api; algoritmo.

## ABSTRACT

On entering the academic environment in courses outside of Information Technology (IT), some new students often have difficulties with basic curricular components of the programming, such as Algorithms. To minimize these difficulties, approaches are proposed, such as the Loop Academic platform. Currently, within the Software Engineering cycle, this platform is in the prototyping stage (user interface design). Thus, the environment needs to be implemented so that it can be effectively used by students entering higher education. In this work, a software architecture for this tool is developed. The architecture development was based on the proposal of dividing the architecture into different views by Bass et al. (2012), which proposes three categories of presentation of architectural artifacts, the module view, component & connector, and implementation. The three views were designed for the software, allowing them to have different perspectives of the environment to be developed. To evaluate the architectural proposal, two approaches were developed; the first one was based on platform usage scenarios, and the second one focused on the development of software artifacts generated in a programming language, focused on the back-end part of the application, which can be integrated with the front-end part of the application. As a result, it was possible to identify that the architecture can be implemented and integrated with an external environment, or front-end of the application. As a contribution, implementing Loop Academic allows this promising tool to reduce failure rates in the discipline, especially among incoming students, providing more continuous practice and facilitating the understanding of concepts.



**Keywords**: back-end; front-end; api; Algorithm.

## LISTA DE FIGURAS

Figura 1 - Diagrama da Arquitetura do Django Rest Framework…….……...19
Figura 2 -Diagrama de Caso de Uso…….……...22
Figura 3 -Diagrama de Classes…….…….……...28
Figura 4 - Diagrama de Componentes…..…..…..…….……..……...31
Figura 5 - Diagrama de Implantação……32
Figura 6 - Diagrama de Sequência Resolução de exercício..…….……...……...34
Figura 7 - Diagrama de Sequência consultar Perfil e Desempenho…….……...….....35
Figura 8 Diagrama de Sequência Enviar Dúvidas e Consulta Emblemas……36
Figura 9 - Documentação da API - Interface para registro de novos Alunos…….47
Figura 10 - Documentação da API - Interface para Login de Alunos.…….…..38
Figura 11 - Documentação da API - Interface de Vinculação do Aluno à Turma…..…….38
Figura 12 - Documentação da API - Interface de Verificação de Cadastro em outra Turma...39
Figura 13 - Documentação da API - Interface de Visualização do Perfil.……....39
Figura 14 - Documentação da API - Interface de Edição de Foto de Perfil do Aluno…….40
Figura 15 - Documentação da API - Interface de Lista de Exercício.……41
Figura 16 - Documentação da API -Interface de Status da Lista de Exercício…….. .42
Figura 17 - Documentação da API - Interface para Responder Exercícios.……...…….42
Figura 18 - Documentação da API - Interface para Dica do Exercício.……..….43
Figura 19 - Documentação da API - Interface de Consulta de Material de Apoio. ……...44
Figura 20 - Documentação da API - Interface para Criar Forum. …….…….….45
Figura 21 - Documentação da API -Interface de Exibição de Fóruns…….…..…….45
Figura 22 - Documentação da API - Interface para Responder a Tópicos no Fórum...…….46
Figura 23 - Documentação da API - Interface para Visualizar Desempenho do Aluno…….47
Figura 24 - Documentação da API - Interface para Visualizar Emblema Desbloqueado do Aluno……47
Figura 25 - Documentação da API - Interface para Visualizar Emblema Todos os Emblemas Disponível……...48

## LISTA DE QUADROS

Quadro 1 - Aluno……..……...…….21
Quadro 2 - Realizar Cadastro do Aluno…….. ……...…….23
Quadro 3 - Login do Aluno…….…….……23
Quadro 4 - Solução de Exercícios…..……...…….……...……..23
Quadro 5 - Consulta Código de Apoio……..……...…....23
Quadro 6 - Consulta Dicas Do Exercício…….…....24
Quadro 7 - Consultar o Material de Apoio…….……..……....24
Quadro 8 - Consultar Desempenho do Aluno...……...……....24
Quadro 9 - Enviar as Dúvidas……..…….……...…….....24
Quadro 10 - Criar Tópico do Fórum..……..……...…...……..25
Quadro 11 - Responder Tópico do Fórum……..…...…….……..……...25
Quadro 12 - Consulta Emblemas do Aluno.……...…...…….……..……...25
Quadro 13 - Disponibilidade do Sistema……...……...…….…..25
Quadro 14 - Interface do Sistema.……...…….....…….…...26

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| *DRF* | *Django REST Framework* |
| *API* | *Application Programming Interface* |
| WEB | *World Wide Web* |
| HTML | Hypertext Markup Language |
| CSS | Cascading Style Sheets |

**SUMÁRIO**

**1. INTRODUÇÃO..... 13**
1.1 JUSTIFICATIVA..... 14
1.2 OBJETIVOS..... 14
**2. FUNDAMENTAÇÃO TEÓRICA..... 16**
2.1 TECNOLOGIAS WEB..... 16
2.2 PYTHON..... 16
2.3 DJANGO..... 17
2.4 PRINCÍPIOS ARQUITETURAIS DE LEN BASS..... 18
2.5 DJANGO REST FRAMEWORK..... 18
**3. PROPOSTA DE SOLUÇÃO..... 21**
3.1 LEVANTAMENTO DE REQUISITOS..... 21
3.2 PROPOSTA DE ARQUITETURA DA PLATAFORMA..... 26
**4. AVALIAÇÃO..... 34**
4.1 AVALIAÇÃO BASEADA EM CENÁRIOS DE USO E INTERAÇÃO COM O SISTEMA..... 34
4.2 AVALIAÇÃO BASEADA NA IMPLEMENTAÇÃO DE ARTEFATOS ARQUITETURAIS..... 37
4.3 CONSIDERAÇÕES SOBRE A API..... 48
**5. CONSIDERAÇÕES FINAIS..... 49**
5.1 CONSIDERAÇÕES FUTURAS..... 50
**REFERÊNCIAS..... 51**

# 1. INTRODUÇÃO

A constante evolução da Tecnologia da Informação (TI) tem transformado profundamente a maneira como as instituições educacionais operam e interagem com seus membros. Em cursos superiores, especialmente na área de TI, os estudantes frequentemente enfrentam dificuldades com disciplinas voltadas para programação. Um exemplo disso é o curso de Bacharelado em Tecnologia da Informação (BTI) da Universidade Federal Rural do Semi-Árido (UFERSA), Campus Pau dos Ferros, em que a taxa de insucesso nas componentes curriculares associadas a algoritmos é alta (Souza, 2019).

Para minimizar esse problema, algumas abordagens foram propostas, dentre eles o uso da plataforma Loop Academic. Essa plataforma visa reduzir as taxas de reprovação nas disciplinas de algoritmos e promover um aprendizado para os estudantes de TI e áreas relacionadas. O Loop Academic foi concebido para fornecer oportunidades interativas que aprimorem o conhecimento dos estudantes, especialmente aqueles do primeiro período (Souza, 2019). Ao utilizar o sistema, os alunos terão acesso a exercícios práticos, recursos adicionais, videoaulas e materiais de apoio, permitindo uma aprendizagem ativa e engajada, além das aulas convencionais e dos trabalhos sugeridos pelos professores.

No entanto, o Loop Academic se encontra como protótipo (*design* da interface), levando à necessidade de implementação para que a plataforma possa ser utilizada no ambiente educacional. Dessa forma, a proposta deste trabalho é projetar uma arquitetura de software para este ambiente. Para o projeto da arquitetura foram adotadas as visões de software propostas por Bass et al. (2012).

Para avaliar a arquitetura foram adotadas duas abordagens, sendo a primeira delas baseada em cenários de uso e a segunda na implementação do *back-end* do Loop Academic. Também  externo do ambiente, assegurando que o sistema seja adaptado às necessidades dos estudantes.

Embora o sistema ainda esteja em fase de prototipagem e não tenha sido implementado em uma linguagem de programação específica, os testes internos realizados até o momento têm fornecido *insights* valiosos para o desenvolvimento iterativo da solução (Souza, 2019).

## 1.1 JUSTIFICATIVA

Para minimizar esse problema, propõe-se a implementação de uma API (Application Programming Interface) para o sistema Loop Academic. Esta iniciativa visa melhorar o desempenho dos estudantes na disciplina de Algoritmos, promovendo um aprendizado mais interativo. A seguir, destacam-se os principais benefícios dessa implementação:

- **Acesso Facilitado a Materiais de Apoio:** A API integrará recursos educacionais diretamente na plataforma Loop Academic, como materiais de estudo, videoaulas e uma ampla gama de exercícios práticos. Isso eliminará a necessidade dos alunos buscarem materiais externos, proporcionando um aprendizado contínuo e focado.
- **Correção Instantânea de Exercícios:** Um dos principais desafios na disciplina de Algoritmos é a falta de correções imediatas. Com a API, os alunos poderão verificar seu desempenho em tempo real. Caso uma questão esteja errada, poderão realizar várias tentativas até acertar, assim promovendo o seu desempenho na disciplina.
- **Engajamento e Motivação dos Alunos:** A API também promoverá o engajamento dos estudantes por meio de um sistema de emblemas. À medida que alcançam boas notas e resolvem questões, desbloqueiam novos emblemas, incentivando-os a se dedicarem mais aos estudos.

Portanto, a implementação da API Loop Academic é essencial para promover a inclusão e o sucesso acadêmico dos discentes, especialmente nas disciplinas de Algoritmos. A plataforma não só suprir as lacunas do ensino tradicional, mas também modernizar a forma como o conteúdo é transmitido, oferecendo uma experiência de aprendizado mais dinâmica, acessível e personalizada.

## 1.2 OBJETIVOS

Este projeto visa desenvolver a arquitetura de software do Loop Academic com o intuito de aprimorar a aprendizagem dos alunos ingressantes, especialmente na disciplina de Algoritmos. A plataforma buscará oferecer uma experiência de aprendizado centralizada e acessível, promovendo a prática de exercícios interativos e permitindo que os estudantes acompanhem seu desempenho acadêmico. Além disso, o sistema incluirá funcionalidades para a criação de tópicos e respostas em um fórum, além de permitir o envio anônimo de

dúvidas, criando um ambiente de aprendizado inclusivo. O objetivo final é implementar o back-end da plataforma, preparando-a para futuras integrações com o front-end, visando ensinar conceitos de programação.

Para alcançar o objetivo geral descrito anteriormente, foram definidos os seguintes objetivos específicos:

- Desenvolver uma API capaz de gerenciar as tarefas importantes do Loop Academic.
- Vincular a API a recursos de ensino, como tutoriais, exercícios práticos e videoaulas, para que os alunos possam encontrá-los e usá-los facilmente.
- O aluno terá acesso a uma área dedicada para acompanhar seu desempenho à medida que resolve os exercícios

## 1.3 ORGANIZAÇÃO DO TRABALHO

O presente trabalho está organizado da seguinte forma: no Capítulo 2, é apresentado o embasamento teórico e alguns trabalhos relacionados a esta pesquisa; no Capítulo 3. é apresentada a proposta de solução e os resultados alcançados; no Capítulo 5, são expostas as conclusões obtidas.

# 2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão discutidos alguns conceitos e termos importantes para o entendimento deste trabalho.

## 2.1 TECNOLOGIAS WEB

As tecnologias da web constituem um conjunto de ferramentas e técnicas utilizadas para a comunicação entre dispositivos na Internet. Elas envolvem componentes como navegadores web, por exemplo, Google Chrome e Firefox, que exibem páginas HTML e interpretam conteúdos de multimídia (ICLOUD, 2017). A estrutura básica da web é composta pelas seguintes tecnologias:

- **HTML** (Hypertext Markup Language): define a estrutura das páginas web.
- **CSS** (Cascading Style Sheets): estiliza visualmente o conteúdo, definindo cores, fontes e layout.
- **JavaScript:** torna as páginas interativas e dinâmicas.

Os navegadores enviam solicitações a servidores web, que processam os pedidos e retornam páginas através do protocolo HTTP. Os servidores também interagem com bancos de dados, como MySQL ou MongoDB, que armazenam as informações exibidas nos sites (Icloud, 2017).

As APIs (Interface de Programação de Aplicações) são utilizadas para integrar sistemas, permitindo que desenvolvedores utilizem funcionalidades de outras aplicações sem precisar acessar diretamente o código-fonte. Tecnologias como REST e JSON são empregadas para a comunicação entre APIs (Icloud, 2017).

A evolução das tecnologias da web possibilitou o surgimento de *frameworks* e bibliotecas, como Bootstrap e jQuery, que simplificam o desenvolvimento de interfaces. Além disso, linguagens de programação como PHP, Python e Java são amplamente utilizadas no *back-end* para manipulação de dados e lógica de negócio.

## 2.2 PYTHON

Python é uma linguagem de programação simples e versátil, ideal tanto para iniciantes quanto para especialistas. Sua popularidade é reforçada por uma comunidade ativa, uma vasta

coleção de bibliotecas e *frameworks* úteis, além de uma documentação acessível, que facilita o aprendizado e o desenvolvimento de programas complexos. Embora linguagens como C++ possam oferecer maior controle sobre certos aspectos, a flexibilidade de Python, sua tipagem dinâmica e o amplo conjunto de ferramentas disponíveis compensam essas limitações, acelerando o processo de desenvolvimento e tornando a codificação mais intuitiva (Awari, 2024).

## 2.3 DJANGO

Django é um *framework* web baseado em Python, que facilita o desenvolvimento de sites seguros e de fácil manutenção. Com diversas ferramentas integradas, ele permite que os programadores foquem na criação de aplicativos sem a necessidade de reinventar funcionalidades já existentes (Mozilla, 2024).

Um dos aspectos mais destacados do Django é seu forte compromisso com a segurança, protegendo os aplicativos contra ataques comuns, como injeção de SQL e XSS. Além disso, sua estrutura segue o princípio DRY (*Don't Repeat Yourself*), incentivando a reutilização de código e evitando duplicações desnecessárias.

Django é altamente portátil, funcionando de maneira eficaz em sistemas operacionais como Linux, Windows e macOS. Criado por uma equipe de desenvolvedores voltada para o setor de notícias no início dos anos 2000, o *framework* foi lançado ao público em 2005. Desde então, passou por inúmeras melhorias, continuando a evoluir para atender às necessidades modernas do desenvolvimento web (Mozilla, 2024).

O ORM (Object-Relational Mapping) do Django é uma de suas funcionalidades mais poderosas, permitindo que os desenvolvedores interajam com bancos de dados de forma intuitiva, sem a necessidade de escrever comandos SQL diretamente. Com o uso de QuerySets, é possível realizar operações como criação, leitura, atualização e exclusão de registros, além de aplicar filtros e ordenações. Essa abordagem não apenas torna o código mais legível e fácil de manter, mas também garante portabilidade e segurança ao abstrair detalhes específicos do banco de dados, alinhando-se ao princípio DRY do framework (ALURA, 2022).

## 2.4 PRINCÍPIOS ARQUITETURAIS

Len Bass, Paul Clements e Rick Kazman (2012) destacam que a arquitetura de software deve ser orientada pelas qualidades desejadas do sistema, como desempenho, segurança e usabilidade. Esses atributos de qualidade são cruciais nas decisões de design e estrutura do sistema, influenciando a forma como o software é construído e mantido. Além disso, eles enfatizam a importância dos estilos arquitetônicos, como camadas e microsserviços, que oferecem vantagens e desvantagens. A escolha adequada do estilo arquitetônico facilita a manutenção e evolução do sistema, garantindo que ele atenda aos requisitos técnicos e agregue valor estratégico à organização. Estes princípios fornecem uma base sólida para o desenvolvimento de sistemas robustos, escaláveis e eficientes (Bass, Clements e Kazman, 2012).

## 2.5 DJANGO REST FRAMEWORK

O Django REST Framework (DRF) é uma biblioteca lançada em fevereiro de 2011, que facilita a criação de APIs RESTful, utilizando o *framework* Django. Amplamente utilizado no desenvolvimento de APIs em diversas plataformas, como Windows, macOS e Linux, o DRF simplifica o processo ao aproveitar os recursos nativos do Django, como roteamento e ORM (*Object-Relational Mapping*) (Vinícius, 2021).

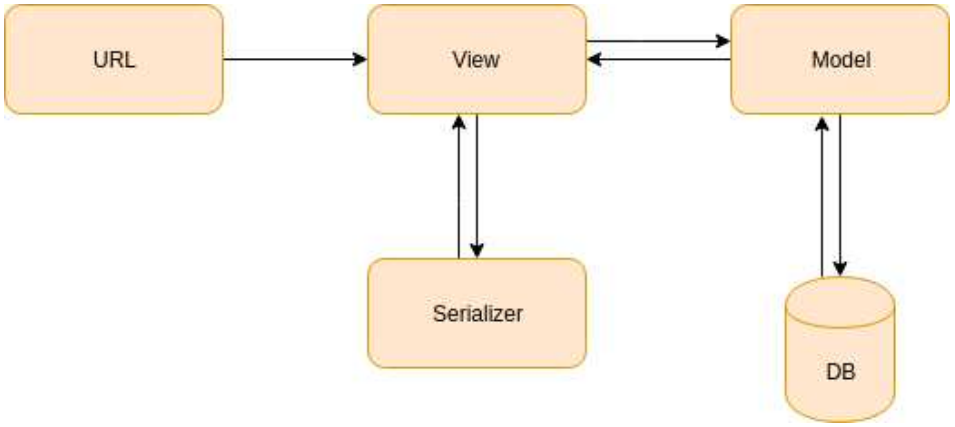
Uma das principais vantagens do DRF é o suporte a autenticação robusta através de *tokens*, o que garante segurança nas interações entre cliente e servidor. Além disso, o DRF permite a filtragem de dados, proporcionando uma visualização mais eficiente das informações e suporta a implementação de *cache*, aumentando a eficiência das aplicações (Treinaweb, 2020).

A integração completa com o Django oferece uma experiência de desenvolvimento fluida e segura. A vasta comunidade de desenvolvedores que apoia o DRF contribui constantemente com novos recursos e melhorias, o que mantém a biblioteca atualizada e relevante. A documentação abrangente do DRF facilita tanto o aprendizado quanto a implementação, tornando-o uma escolha popular entre desenvolvedores que buscam eficiência e praticidade na criação de soluções web.

A arquitetura do Django REST Framework (DRF) segue o padrão MVT (Model-View-Template), que organiza a aplicação em três componentes principais: Model (responsável pelos dados e interações com o banco de dados), View (gerencia as requisições e

respostas HTTP) e Template (define como as informações são apresentadas, embora seja menos utilizado quando o backend expõe APIs REST). Diferente do padrão MVC (Model-View-Controller), no Django, o próprio framework assume parte das responsabilidades de um controlador, simplificando o fluxo entre os dados e a interface. O DRF facilita a criação de APIs RESTful ao fornecer ferramentas para serialização, autenticação, roteamento e permissões, permitindo que o backend e o frontend operem de forma independente, com a troca de dados geralmente em formato JSON (Bastian, 2021) .

**Figura** 1 - Diagrama da Arquitetura do Django Rest Framework




**Fonte:** Documento de arquitetura (2018, online) [1]

## 2.6 BANCO DE DADOS

Um banco de dados é uma coleção organizada de informações estruturadas, geralmente armazenadas e controladas por um Sistema de Gerenciamento de Banco de Dados (SGBD). Os dados são dispostos em linhas e colunas para facilitar o processamento e a consulta, sendo que a maioria dos bancos de dados utiliza a Linguagem de Consulta Estruturada (SQL) para manipulação. O SQL surgiu na década de 1970 e é amplamente utilizado, embora novas linguagens estejam emergindo (Oracle, 2020).

Desde os anos 1960, os bancos de dados evoluíram de sistemas hierárquicos e de rede, passando por bancos de dados relacionais nos anos 1980 e orientados a objetos na década de 1990, até os recentes bancos de dados NoSQL, que tratam dados não estruturados. A tecnologia atual inclui bancos de dados na nuvem e autônomos, que aprimoram o gerenciamento e o uso de dados (Oracle, 2020).

[1]Disponível em: https://fga-eps-mds.github.io/2018.2-Integra-Vendas/docs/doc-arquitetura

Os bancos de dados diferem das planilhas em termos de capacidade, manipulação de dados e acesso simultâneo por vários usuários. Os principais tipos de bancos de dados incluem (Oracle, 2020):

- **Relacionais**: organizam dados em tabelas.
- **Orientados a objetos**: utilizam objetos, como na programação orientada a objetos.
- **Distribuídos**: armazenam dados em vários locais.
- **Data Warehouses**: projetados para consultas rápidas.
- **NoSQL**: para dados não estruturados.
- **Gráficos**: focam em entidades e seus relacionamentos.
- **OLTP**: otimizados para grandes volumes de transações.

# 3. PROPOSTA DE SOLUÇÃO

Neste capítulo, é apresentada a proposta de solução para este trabalho. A Seção 3.1 aborda o levantamento de requisitos necessários para o desenvolvimento do sistema. Na Seção 3.2, são apresentados os diagramas de classes, casos de uso, sequência e componentes, com o objetivo de facilitar o desenvolvimento do software. A Seção 3.3 descreve a arquitetura da API, incluindo imagens que ilustram suas funcionalidades. Por fim, a Seção 3.4 discute os resultados alcançados ao longo do processo.

## 3.1 LEVANTAMENTO DE REQUISITOS

Os requisitos de um sistema são a base fundamental para a documentação e desenvolvimento de software, desempenhando um papel crucial ao garantir que as necessidades e expectativas dos usuários sejam devidamente atendidas (Sommerville, 2011). De acordo com Sommerville (2011), os requisitos devem ser descritos de maneira clara e precisa para que possam orientar todas as etapas do ciclo de vida de desenvolvimento do sistema, desde o *design* até a implementação e manutenção.

### ATORES DO SISTEMA

Os atores envolvidos no uso da plataforma podem ser vistos no Quadro 1.

**Quadro 1** - Aluno

| **Papel** | **Solicitante** |
|---|---|
| Solicitar | O aluno acessa o sistema com o objetivo de estudar, consultar materiais e realizar atividades relacionadas ao aprendizado, como resolução de exercícios e envio de dúvidas. |
| Recursos | O sistema deve fornecer ao aluno uma série de funcionalidades, como consulta de materiais de estudo, resolução de exercícios, participação em fóruns e acompanhamento de desempenho, facilitando o processo de aprendizado e oferecendo suporte em tempo real às suas necessidades acadêmicas. |

**Fonte**: Souza (2019)

### REQUISITOS FUNCIONAIS

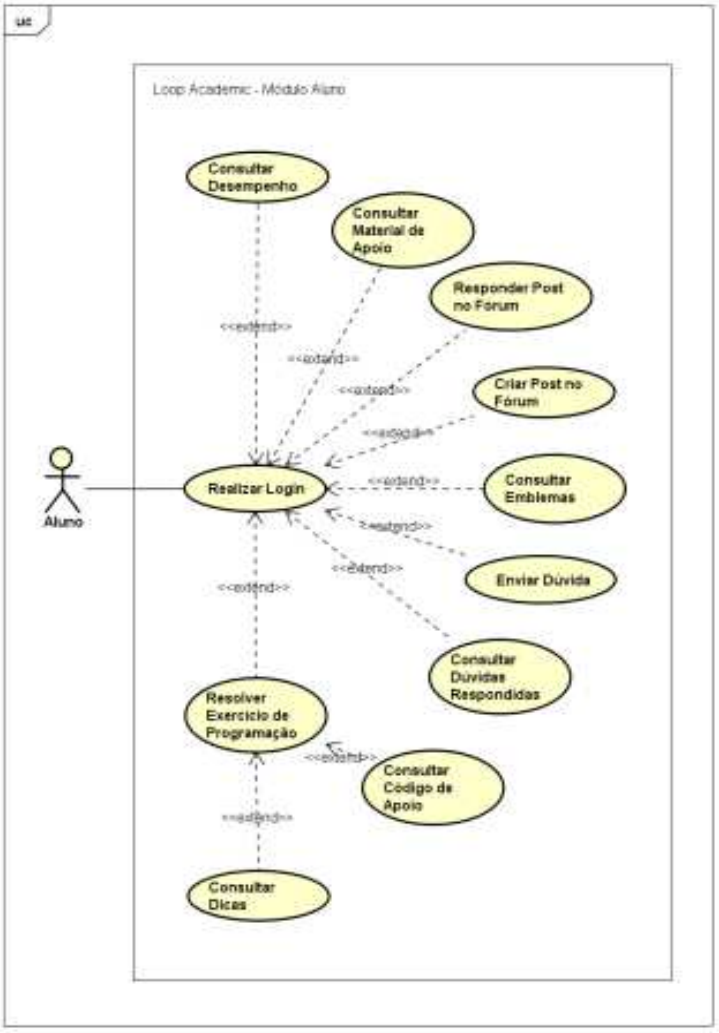
Na Figura 1 pode ser visto um diagrama de casos de uso, destacando as principais interações entre o Aluno e as funcionalidades disponíveis. Nesse sistema, o aluno pode

realizar o login para acessar suas informações e ferramentas de estudo. Entre as ações possíveis, ele pode consultar seu desempenho acadêmico, enviando dúvidas sobre o conteúdo, além de visualizar e baixar o material de apoio disponível.

O sistema também permite que o aluno participe de fóruns, em que pode criar tópicos de discussão e responder a tópicos existentes. Além disso, ele pode consultar dúvidas já respondidas, conferir os emblemas que conquistou ao longo de suas atividades, resolver exercícios de programação e acessar dicas e códigos de apoio relacionados aos exercícios.

Este diagrama foi criado com o software Astah UML (*Unified Modeling Language*), que permite organizar e detalhar de forma clara as interações entre os usuários e o sistema. Isso proporciona uma visão geral das funcionalidades que o aluno pode utilizar, enfatizando uma experiência intuitiva e eficiente para o usuário.

**Figura 2** - Diagrama de Caso de Uso




**Fonte**: Souza (2019)

Os requisitos funcionais para a plataforma Loop Academic podem ser vistos na sequência de quadros a seguir: Quadro 2 ao Quadro 12.

**Quadro 2 -** Realizar Cadastro do Aluno

| **RF001** | **Realizar Cadastro como Aluno** |
|---|---|
| Detalhes | O software deve permitir que o aluno realize o cadastro com os seguintes dados: nome, instituição de ensino, matrícula, curso, e-mail, senha. Para a matrícula, o software deve verificar a sua validade O software deve permitir que o aluno realize o cadastro com os seguintes dados: nome, instituição de ensino, matrícula, curso, e-mail, senha. Para a matrícula, o software deve verificar a sua validade perante a instituição de ensino informada. Para garantia de acesso, um e-mail de confirmação deve ser enviado ao e-mail do usuário, relatando suas credenciais de acesso. |
| Atores | Aluno |
| Importância | ALTA |

**Fonte**: Souza (2019)

**Quadro 3 -** Login do Aluno

| **RF002** | **Realizar Login** |
|---|---|
| Detalhes | O software deve permitir que o aluno realize o seu login a partir das suas credenciais de acesso, configuradas a partir da realização do seu cadastro. |
| Atores | Aluno |
| Importância | ALTA |

**Fonte**: Souza (2019)

**Quadro 4 -** Solução de Exercícios

| **RF003** | **Resolver Exercício** |
|---|---|
| Detalhes | O software deve permitir que o aluno busque, acesse e resolva um exercício pertencente à uma lista de exercícios. A resolução deve ser enviada através do software educacional e possibilitará mensurar o desempenho do aluno. No momento de resolução do exercício, o aluno pode consultar um exemplo de algoritmo com mesma temática (exemplo: algoritmo com implementação de estruturas de decisão) ou consultar dicas. |
| Atores | Aluno |
| Importância | ALTA |

**Fonte**: Souza (2019)

**Quadro 5 -** Consulta Codigo de Apoio

| **RF004** | **Consultar Código de Apoio** |
|---|---|
| Detalhes | O software deve permitir que o aluno consulte algoritmos com temática similar ao exercício no qual está resolvendo. Para tal, o sistema deve possuir uma opção que viabilize o acesso ao código de apoio no momento em que a resolução estiver sendo escrita. |

| | |
|---|---|
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

**Quadro 6 -** Consulta Dicas do Exercício

| **RF005** | **Consultar Dicas** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno acesse dicas que o permitam sanar dúvidas e dificuldades na resolução de um exercício. |
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

**Quadro 7 -** Consultar o Material de Apoio

| **RF006** | **Consultar Material de Apoio** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno consulte materiais de apoio sobre as temáticas relacionadas à programação. Os materiais apoiam os estudos. |
| Atores | Aluno |
| Importância | ALTA |

**Fonte**: Souza (2019)

**Quadro 8 -** Consultar Desempenho do Aluno

| **RF007** | **Consultar Desempenho** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno consulte o seu desempenho nas listas de exercícios resolvidas. Na ocorrência de desempenhos baixos, devem ser direcionados ao aluno materiais de apoio com temática similar a qual o aluno tem dificuldades. |
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

**Quadro 9 -** Enviar as Dúvidas

| **RF008** | **Enviar Dúvidas** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno envie dúvidas de forma anônima, que serão respondidas pelo professor ou monitor. |
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

**Quadro 10** - Criar Tópico do Fórum.

| **RF009** | **Criar Tópico no Fórum** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno crie tópicos de discussão, que estarão disponíveis em um fórum para todos os outros usuários (alunos, professores e monitores). |
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

**Quadro 11 -** Responder Tópico do Fórum

| **RF010** | **Responder Tópico no Fórum** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno responda tópicos no fórum. |
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

**Quadro 12 -** Consulta Emblemas do Aluno

| **RF011** | **O Consultar Emblemas** |
|---|---|
| Detalhes | O sistema deve permitir que o aluno consulte os emblemas com o uso do Loop Academic. Cada emblema deve abranger uma tarefa diferente. Os emblemas devem ser visíveis nos tópicos do fórum e cada usuário deverá ter um informativo com a quantidade de emblemas obtidos até o momento. |
| Atores | Aluno |
| Importância | MÉDIA |

**Fonte**: Souza (2019)

## REQUISITOS NÃO-FUNCIONAIS

Nos quadros 13 e 14 são elencados os requisitos não funcionais do ambiente.

**Quadro 13 -** Disponibilidade do Sistema

| **RNF001** | **Disponibilidade Somente Online** |
|---|---|
| Detalhes | O sistema opera de forma online, garantindo acesso contínuo a recursos atualizados. |
| Categoria | Disponibilidade |

**Fonte**: Souza (2019)

**Quadro 14 -** Interface do Sistema

| **RNF002** | **Interface** |
|---|---|
| Detalhes | A interface do sistema deve satisfazer as premissas da Engenharia de Software, garantindo a satisfação do usuário e respeito a sua experiência de uso. |
| Categoria | Usabilidade |

**Fonte**: Souza (2019)

## 3.2 PROPOSTA DE ARQUITETURA DA PLATAFORMA

Para modelar a plataforma do ambiente foi utilizada a divisão em visões arquiteturais proposta por Bass et al. (2012). Nesta modelagem, a arquitetura é dividida em três visões. Visão de módulo, que apresenta a organização estrutural dos elementos de software; a visão componente & conector, que organizar os elementos estruturais em componentes reutilizáveis, permitindo que se tenha uma visão desses elementos em tempo de execução; e, a visão de alocação, que permite visualizar como o software será implantada e que recursos serão alocados a eles.

### VISÃO DE MÓDULO

Todas as visões de um sistema podem ser modeladas usando diagramas UML, como o diagrama de classes, que representa a estrutura modular por meio de entidades e suas relações. O sistema Loop Academic é composto por várias classes que representam diferentes entidades e suas relações (Figura 3[2]). Para a figurar abaixo foi utilizado a ferramenta de modelagem Asta UML. A classe central é o “Aluno”, que possui atributos como nome, instituição de ensino, matrícula, curso, email, senha e a associação a uma turma.

Existem outras classes importantes que interagem diretamente com a classe Aluno. A classe “Forum” permite que o aluno crie tópicos de discussão relacionados a dúvidas acadêmicas, enquanto a classe “ResponderForum” possibilita que os alunos respondam aos tópicos criados. Além disso, a classe “Emblema” registra os emblemas que os alunos conquistam, conforme seu progresso no sistema, incentivando a participação ativa.

[2] Link para o diagrama de classes: https://abrir.link/bkMtW

A classe “Exercicio” representa os exercícios disponíveis no sistema, com atributos como título, descrição, dicas, exemplos de entrada e saída, além de um *status*, que indica se o exercício foi respondido ou não. O aluno pode enviar suas respostas para esses exercícios por meio da classe “ResponderExercicio”, que armazena o código submetido, a pontuação obtida e o resultado.

Há também a classe "Desempenho", que monitora o progresso do aluno nos exercícios, registrando informações como pontuação, tentativas e data de conclusão. A classe “MaterialApoio” contém recursos didáticos complementares, como vídeos do YouTube, arquivos PDF e mapas mentais, que auxiliam o aluno nos estudos.

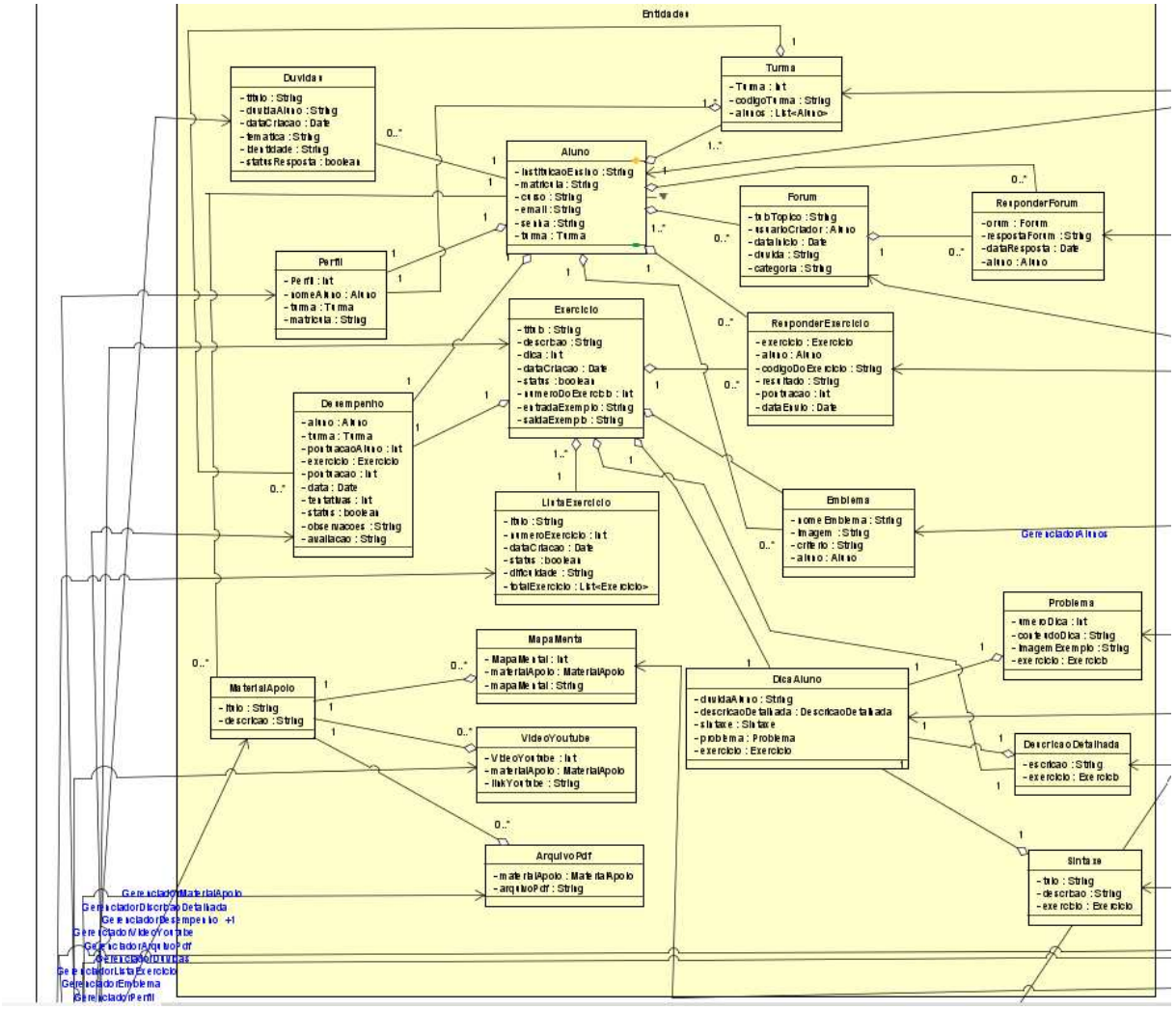
As setas entre as classes indicam as relações entre elas, como associação, agregação e dependência. Por exemplo, um “Aluno” pode estar associado a uma “Turma” e essa turma pode conter vários alunos. Da mesma forma, o “Aluno” pode resolver vários “Exercicios” e responder a diversos “Topicos” no “Forum”.

As classes de entidades estão organizadas em um pacote que contém o gerenciador e interface, formando uma estrutura integrada. Nesse pacote, a classe Gerencia centraliza e executa os métodos das classes de entidade, estabelecendo uma conexão com a interface do Gateway. Esse Gateway atua como intermediário entre o gerenciador e o acesso ao banco de dados, sendo responsável por encaminhar as solicitações ao pacote de banco de dados. Dentro desse pacote, a classe AcessoBanco realiza o acesso direto ao banco de dados final, garantindo uma comunicação eficiente e coesa entre as diferentes camadas do sistema.

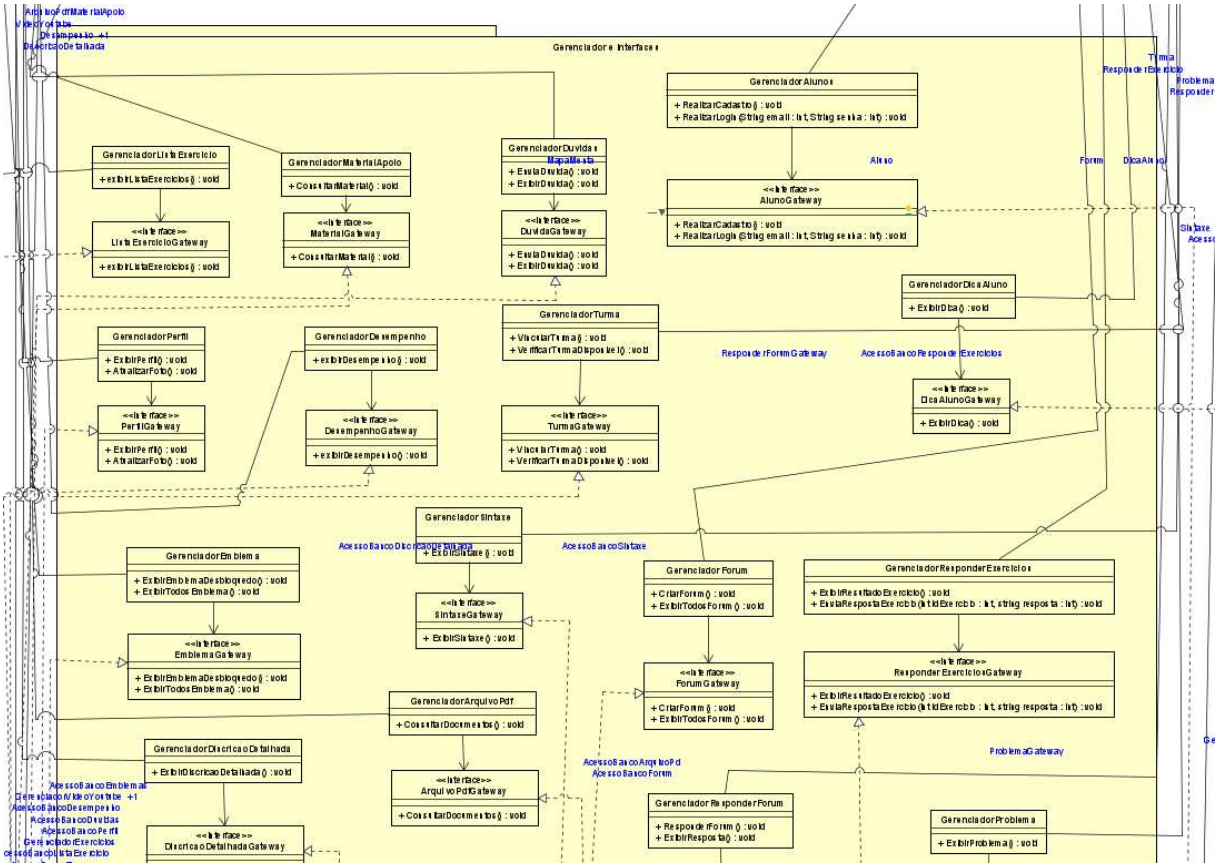
Essa modelagem de classes reflete a estrutura do sistema acadêmico Loop, facilitando a gestão de turmas, a resolução de exercícios, a participação em fóruns e o acompanhamento do desempenho acadêmico.

**Figura 3** - Diagrama de Classes

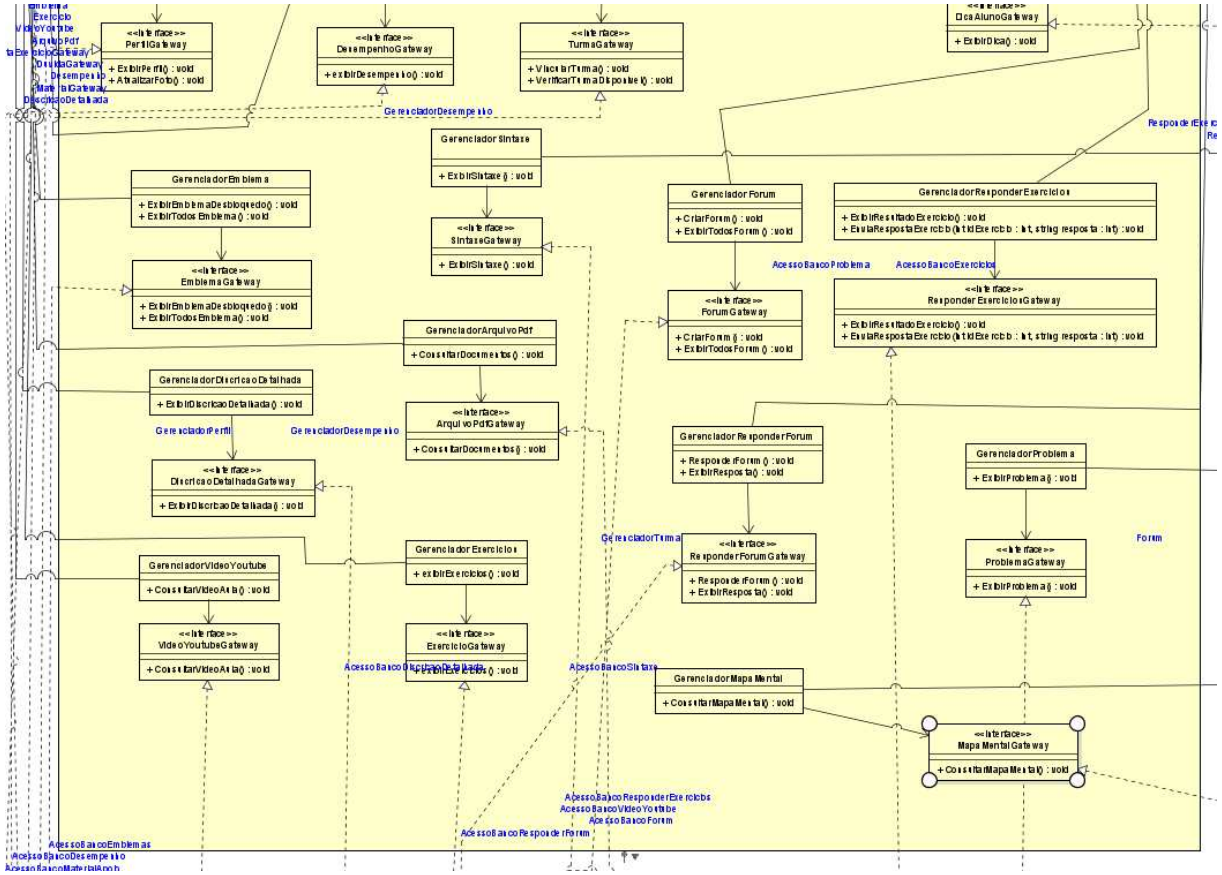
(a) Entidades



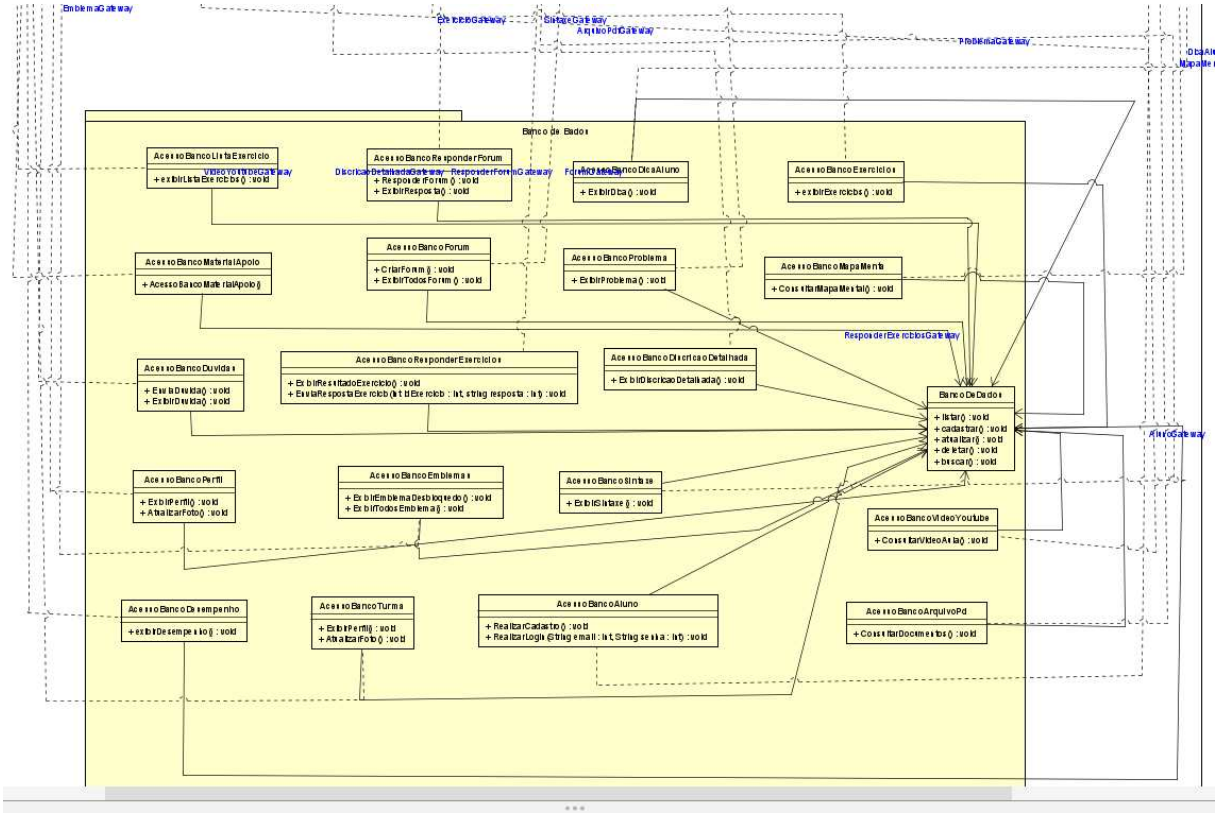
(b) Gerenciadores e interface



(c) Gerenciadores e interface(continuação)



(d) Banco de dados



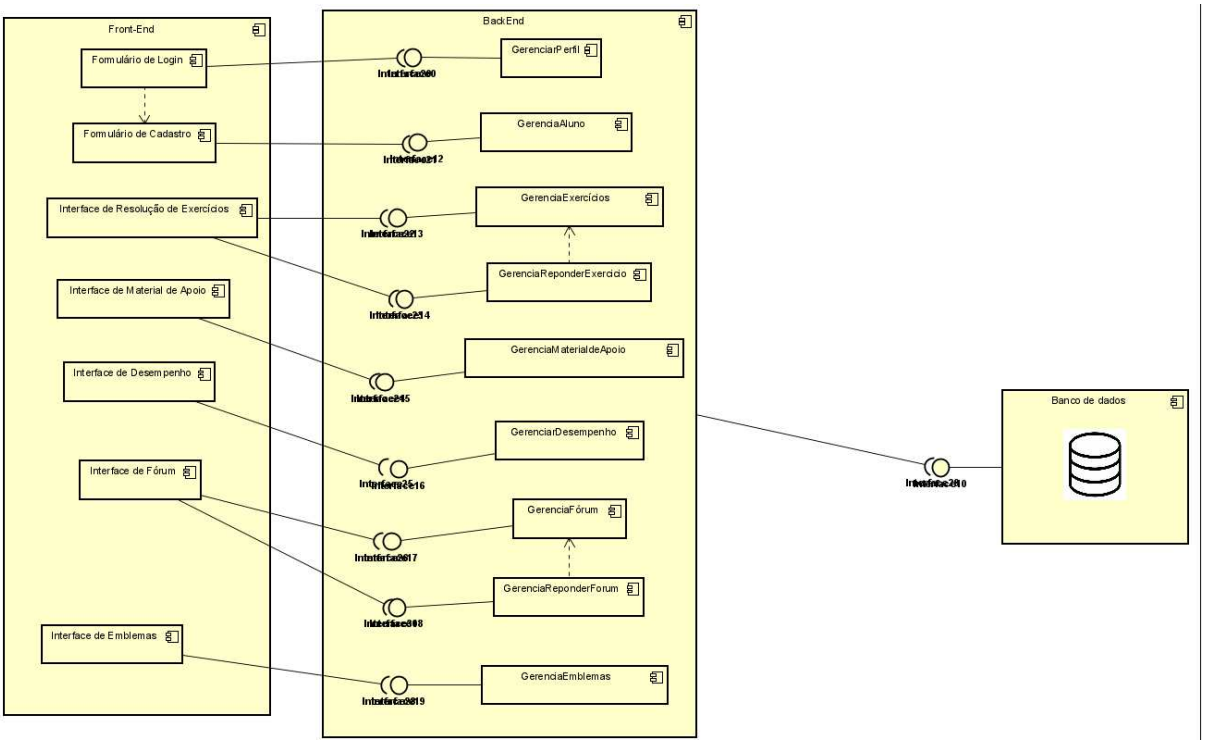

**Fonte**: Autoria própria(2024)

## VISÃO COMPONENTE & CONECTOR

O diagrama de componentes apresentado na Figura 4, modelado com a ferramenta Asta UML, mostra a estrutura do sistema dividida entre o *front-end,* responsável pelas interfaces com o usuário e o *back-end*, que gerencia a lógica de negócios. No *front-end*, o Formulário de Login e o Formulário de Cadastro permitem a autenticação e o registro de alunos. As interfaces de Resolução de Exercícios, Material de Apoio, Desempenho, Fórum e Emblemas possibilitam a interação dos alunos com as principais funcionalidades do sistema, como acessar exercícios, materiais didáticos, consultar o desempenho acadêmico.

No *back-end*, os componentes GerenciarPerfil, GerenciaAluno, GerenciaExercicios, ResponderExercicio, Material de Apoio, Desempenho, Fórum e Emblemas são responsáveis por processar e gerenciar as informações enviadas pelo *front-end*. Esses componentes interagem com o Banco de Dados, que armazena todas as informações essenciais, como dados de perfil, exercícios, respostas enviadas pelos alunos e outros registros necessários para o funcionamento do sistema.

**Figura 4** - Diagrama de Componentes

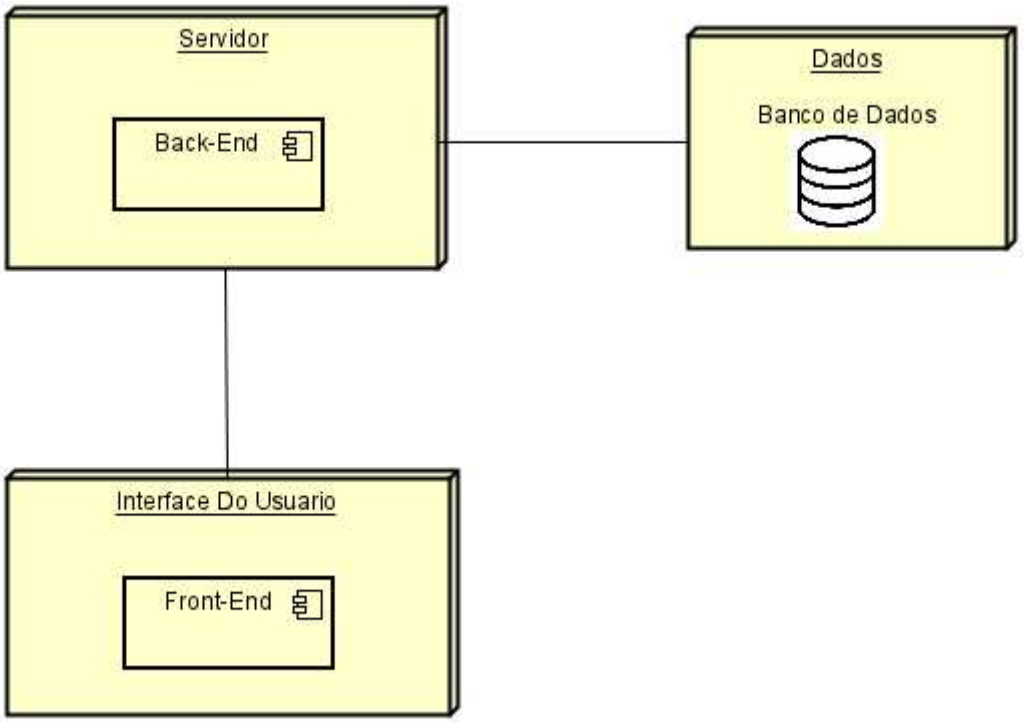



**Fonte**: Autoria própria(2024)

## VISÃO DE ALOCAÇÃO

O diagrama de implantação da Figura 5 modelado usando a ferramenta Asta UML, apresenta a visão de alocação do sistema do Loop Academic, composta por três componentes principais: a Interface do Usuário (*front-end*), o Servidor (*back-end*) e o Banco de Dados. O *front-end* interage diretamente com o usuário, permitindo ações como cliques e preenchimento de formulários, geralmente, utilizando tecnologias como HTML, CSS e JavaScript. O *back-end* é responsável pela lógica de negócios, processando requisições do *front-end*, interagindo com o banco de dados e retornando respostas. Ele pode ser implementado em diversas linguagens, como Python, Java ou Ruby. Por fim, o Banco de Dados armazena as informações do aplicativo, podendo ser relacional (como MySQL ou PostgreSQL) ou NoSQL (como MongoDB). O servidor acessa o banco de dados para realizar operações de criação, leitura, atualização e exclusão de dados. As interações entre esses componentes são fundamentais para o funcionamento eficiente do sistema.

**Figura 5** - Diagrama de Implantação




**Fonte**: Autoria própria(2024)

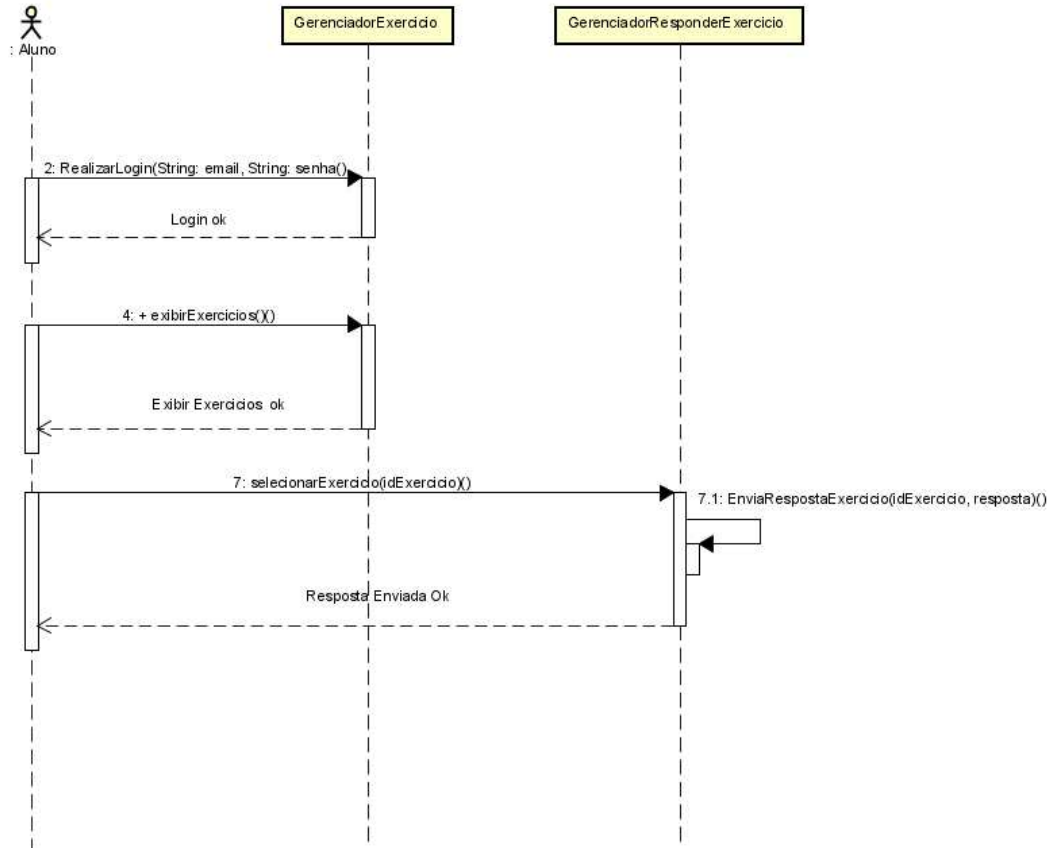
# 4. AVALIAÇÃO

Para avaliar a arquitetura proposta foram utilizadas duas abordagens. A primeira delas consiste em uma avaliação baseada em cenários de uso, uma abordagem comum empregada em avaliações arquiteturais, a exemplo do método *Architecture Tradeoff Analysis Method* — ATAM (Bass et al. (2012). O cenário de uso da arquitetura é modelado com o uso de diagramas de sequência da UML.

## 4.1 AVALIAÇÃO BASEADA EM CENÁRIOS DE USO E INTERAÇÃO COM O SISTEMA

O diagrama de sequência da Figura 6 modelado usando a ferramenta Asta UML descreve a interação entre o aluno, o sistema de exercícios e a funcionalidade de responder exercícios. Inicialmente, o aluno realiza o login no sistema, recebendo uma confirmação após a autenticação bem-sucedida. Em seguida, ele visualiza a lista de exercícios disponíveis para resolver. Após selecionar um exercício, o aluno envia sua resposta ao sistema. O sistema processa a resposta e confirma o envio. O aluno repete esse processo para cada exercício, acompanhando o progresso de suas respostas através do sistema.

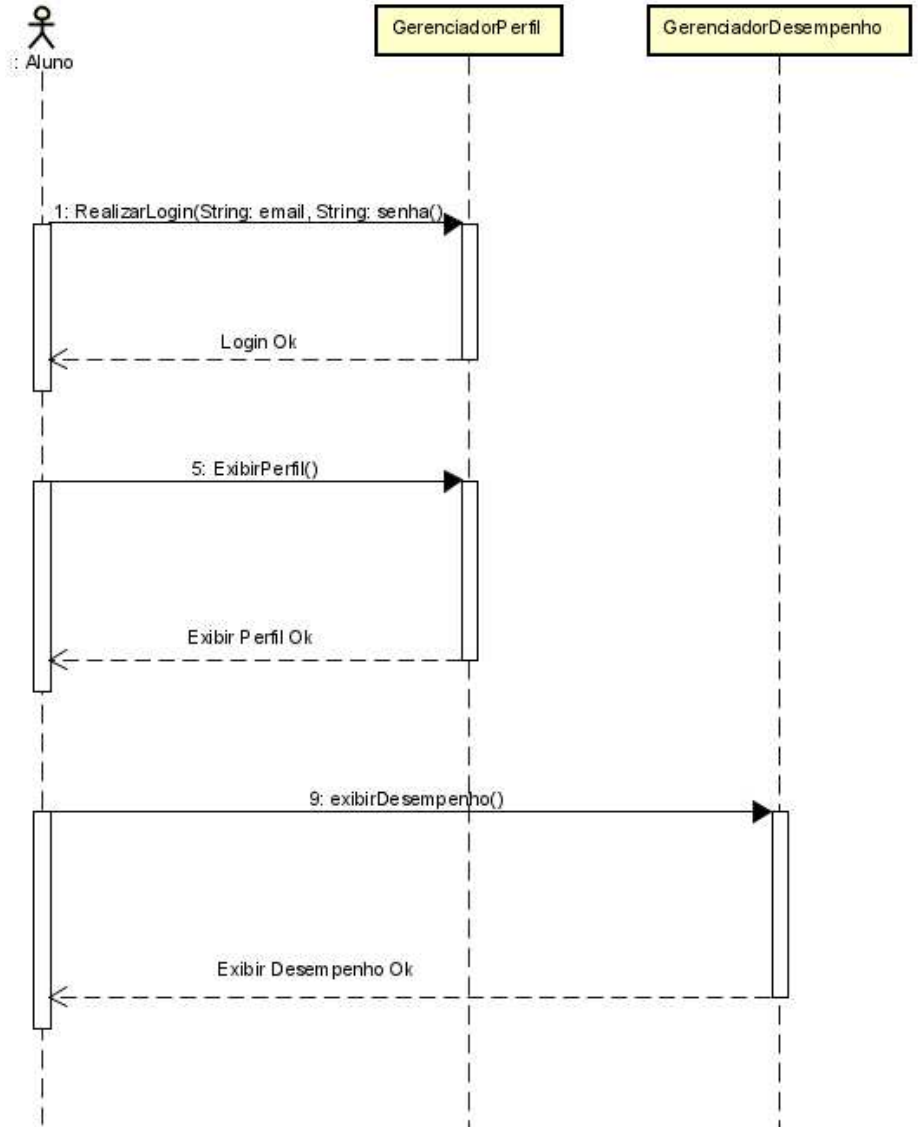
**Figura 6** - Diagrama de Sequência Resolução de exercício




**Fonte**: Autoria própria(2024)

O Diagrama de Sequência da Figura 7 modelado usando a ferramenta Asta UML demonstra a interação entre o aluno e os módulos do sistema responsáveis pelo login e pela exibição de perfil e desempenho. O fluxo começa quando o aluno faz login, fornecendo seu email e senha. O sistema verifica as credenciais do aluno. Se o login for validado com sucesso, o sistema consulta o gerenciador de desempenho para buscar as informações sobre o progresso acadêmico do aluno. Após essa verificação, o sistema envia de volta ao aluno os dados completos de seu perfil, juntamente com as informações de desempenho. Isso permite que o aluno visualize seu progresso acadêmico como parte da confirmação de um login bem-sucedido.

**Figura 7** - Diagrama de Sequência consultar Perfil e Desempenho



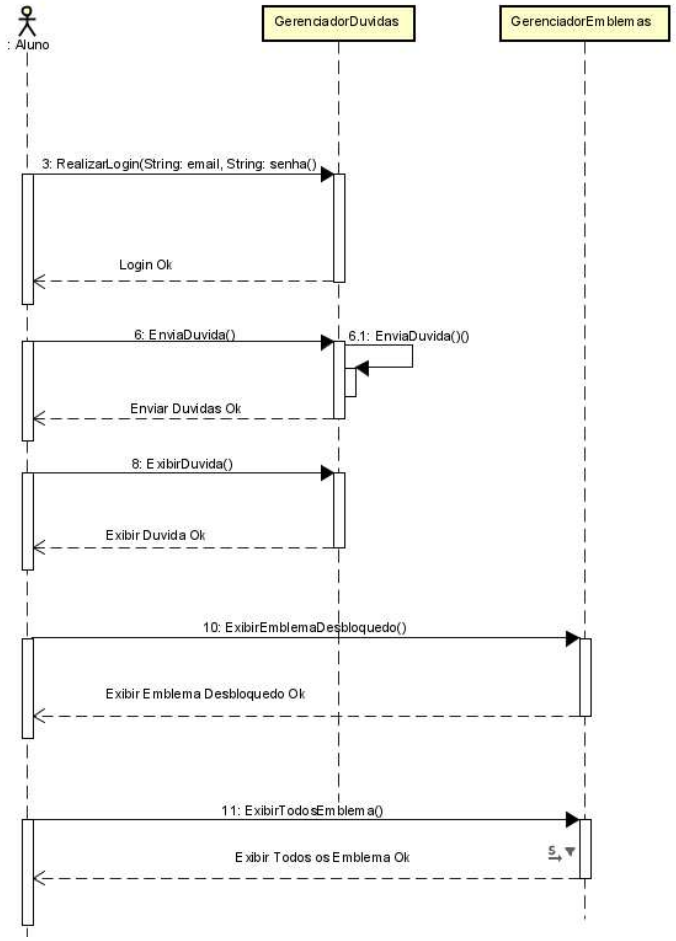

**Fonte**: Autoria própria(2024)

O Diagrama de Sequência da Figura 8 modelado usando a ferramenta Asta UML ilustra a interação do aluno com o sistema para gerenciar dúvidas e emblemas. Inicialmente, o aluno faz login utilizando seu email e senha, e o sistema valida essas informações, confirmando com sucesso o login. Com o acesso garantido, o aluno pode enviar uma dúvida ao sistema, que é processada e registrada pelo Gerenciador de Dúvidas. Além disso, o aluno tem a opção de visualizar as dúvidas que já enviou, recebendo informações detalhadas sobre o conteúdo e o status delas.

Em relação aos emblemas, o aluno também pode consultar os emblemas disponíveis, que são reconhecimentos e recompensas que refletem suas conquistas dentro do sistema. O

aluno pode ver tanto os emblemas que já conquistou quanto os que estão disponíveis para obtenção. O Gerenciador de Emblemas, por sua vez, retorna as informações solicitadas sobre todos os emblemas, garantindo que o aluno tenha uma visão completa de suas conquistas e do que ainda pode alcançar.

**Figura 8** - Diagrama de Sequência Enviar Dúvidas e Consulta Emblemas



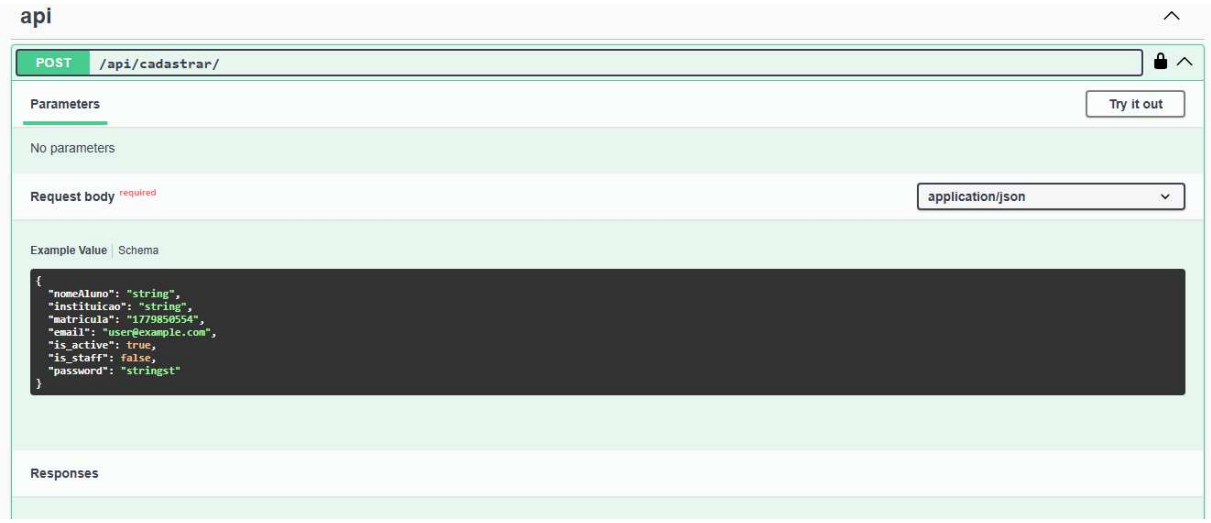

**Fonte**: Autoria própria(2024)

## 4.2 AVALIAÇÃO BASEADA NA IMPLEMENTAÇÃO DE ARTEFATOS ARQUITETURAIS

Nesta seção, é apresentada a avaliação baseada na implementação de artefatos arquiteturais. Para isso, foi desenvolvida a parte *back-end* da aplicação, utilizando o *framework* DJANGO REST. Para exemplificar a implementação, é mostrada a API

desenvolvida, que pode ser consumida pelo *front-end* da aplicação. A Figura 9 ilustra a parte da API para cadastro de aluno, apresentando um *endpoint* configurado para o método POST. Este *endpoint* permite a inclusão de novos alunos no sistema, aceitando dados em formato JSON, que contém todas as informações necessárias para o cadastro. O formulário deve incluir dados como nome, e-mail, matrícula e outros detalhes relevantes, garantindo que o aluno seja adicionado corretamente à base de dados.

**Figura 9 -** API - Cadastro de novos Alunos.




**Fonte**: Autoria própria(2024)

A Figura 10 ilustra a parte da API referente ao *endpoint* de Login do Aluno. Esse *endpoint* é configurado para o método POST e tem como função gerar um *token* de acesso para autenticar o aluno no sistema de maneira segura. O processo envolve a validação dos dados fornecidos, como e-mail e senha, garantindo que apenas usuários com credenciais válidas possam acessar suas respectivas contas. A implementação é projetada para oferecer uma conexão confiável e protegida, utilizando padrões de autenticação, como JWT (JSON Web Token), para manter a integridade e a segurança dos dados do usuário durante o login.

**Figura 10** - API - Interface para Login de Alunos.


**Fonte**: Autoria própria(2024)

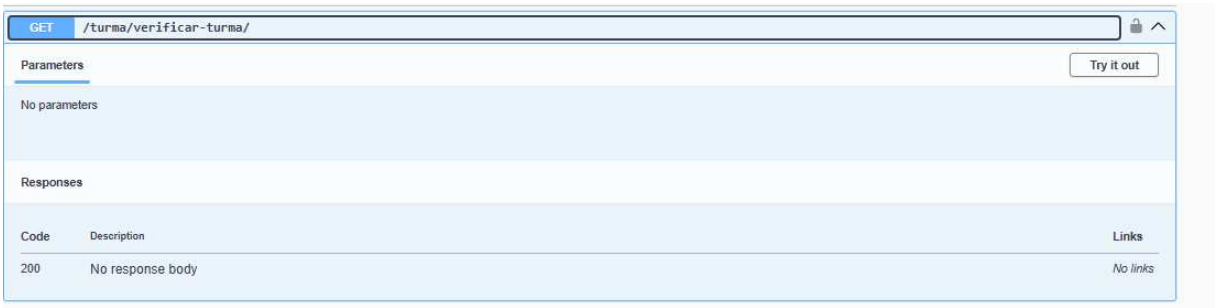
A Figura 11 apresenta a parte da API referente ao processo de vinculação de um aluno a uma turma, por meio de um código específico. Esse *endpoint* é configurado para associar o aluno à turma, utilizando o código fornecido. O sistema valida a autenticidade do código e, ao confirmar sua validade, vincula o aluno à turma correspondente, garantindo que ele seja registrado corretamente conforme as regras estabelecidas.

**Figura 11** - API - Interface de Vinculação do Aluno à Turma.


**Fonte**: Autoria própria(2024)

A Figura 12 apresenta a parte da API referente ao processo de validação do vínculo de um aluno com uma turma. Esse *endpoint* verifica se o aluno já está associado a uma turma e retorna uma resposta, indicando se o vínculo existe ou não. Caso o aluno não esteja vinculado a nenhuma turma, a resposta indica que ele ainda não é associado com a turma em questão.

**Figura 12** - API - Interface de Verificação de Cadastro em outra Turma.




**Fonte**: Autoria própria(2024)

A Figura 13 apresenta a parte da API referente à exibição dos dados do perfil do aluno. Esse *endpoint* é acessado por meio do método GET e retorna informações do aluno atualmente logado no sistema, como nome, foto de perfil, matrícula e outros dados pertinentes. Através desse recurso, o aluno pode visualizar seus dados de forma segura e personalizada.

**Figura 13** - API - Interface de Visualização do Perfil.


Fonte: Autoria própria(2024)

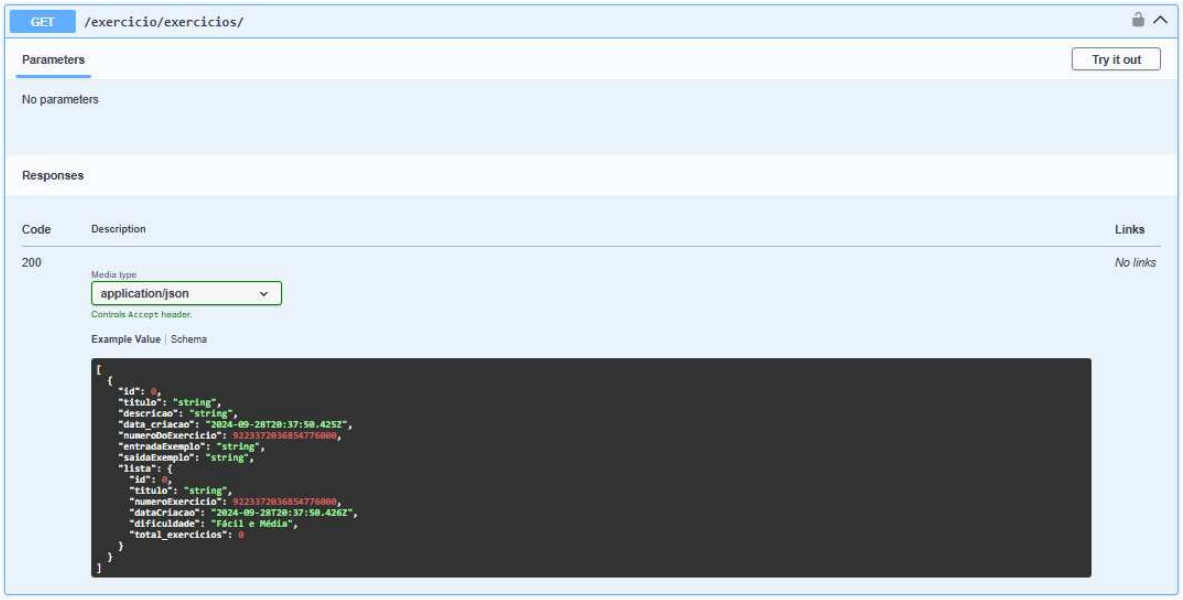
A Figura 14 apresenta a parte da API referente à atualização da foto de perfil do aluno. Esse *endpoint* utiliza o método PATCH, permitindo que o aluno logado no sistema altere sua imagem de perfil de forma simples e eficiente. A funcionalidade assegura que os alunos possam personalizar sua presença no sistema, contribuindo para uma experiência mais individualizada e representativa.

**Figura 14** - API - Interface de Edição de Foto de Perfil do Aluno.


**Fonte**: Autoria própria(2024)

A Figura 15 apresenta a parte da API relacionada à exibição de todos os exercícios disponíveis. Esse *endpoint* utiliza o método GET para retornar todos os dados dos exercícios, permitindo que os alunos tenham acesso às informações necessárias para realizar suas atividades. Através desse endpoint, os usuários podem visualizar todos os exercícios que estão prontos para serem respondidos.

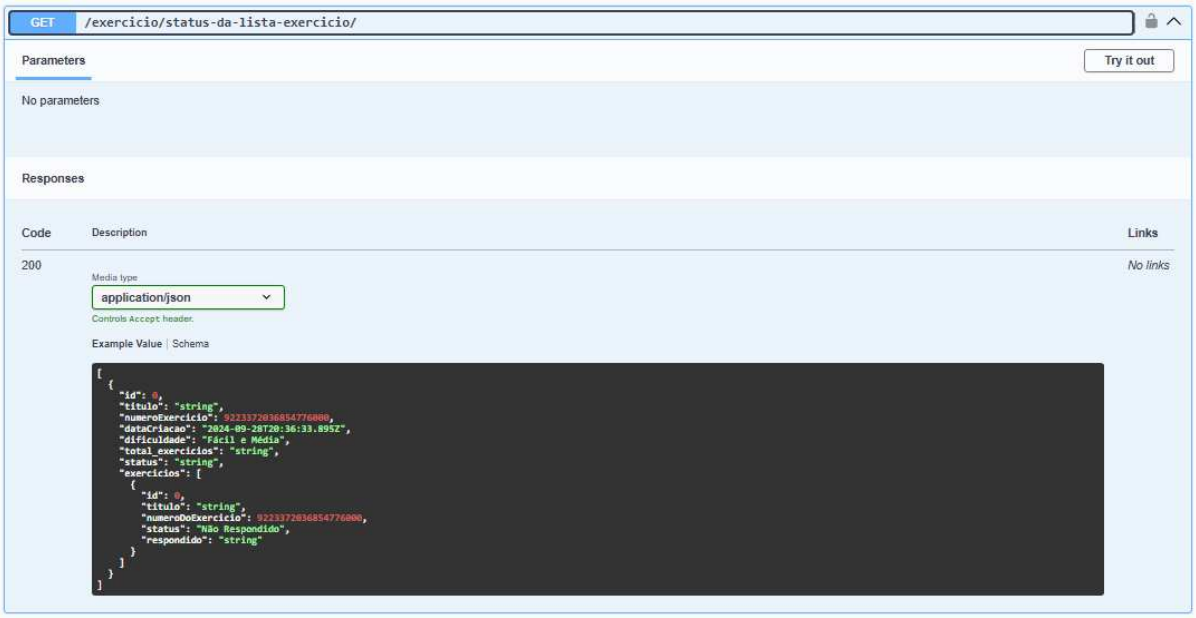
**Figura 15** - API - Interface de Lista de  Exercício.




**Fonte**: Autoria própria(2024)

A Figura 16 apresenta a parte da API referente à exibição da lista de exercícios. Esse *endpoint* utiliza o método GET para fornecer informações detalhadas sobre os exercícios disponíveis, classificados por nível de dificuldade. Dentro dessa lista, cada exercício é acompanhado de seu *status*, que pode indicar se foi "respondido" ou "não respondido". Essa estrutura facilita a visualização e o acompanhamento do progresso do aluno em relação às atividades propostas, permitindo uma gestão eficiente do aprendizado.

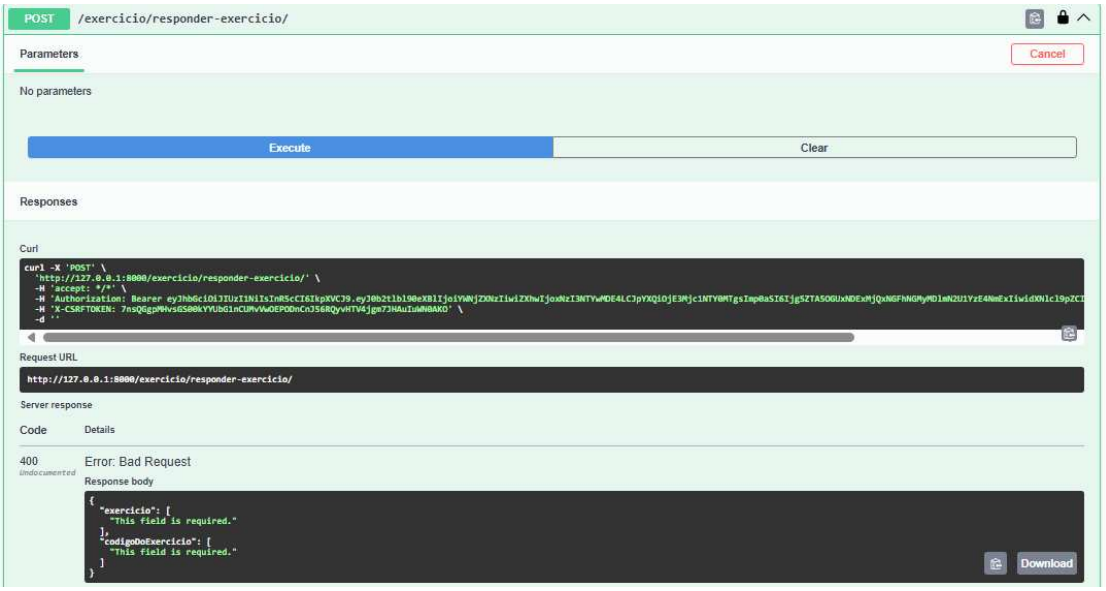
**Figura 16** - API -Interface de Status da Lista de Exercício.



**Fonte**: Autoria própria(2024)

A Figura 17 apresenta a parte da API referente à funcionalidade de resposta de exercícios. Esse *endpoint* utiliza o método POST para enviar a resposta do exercício, identificando-o pelo seu ID e incluindo a resposta em código C. Essa implementação permite que os alunos respondam de forma prática e eficiente, contribuindo para o seu aprendizado ao aplicar o conhecimento de programação em situações reais.

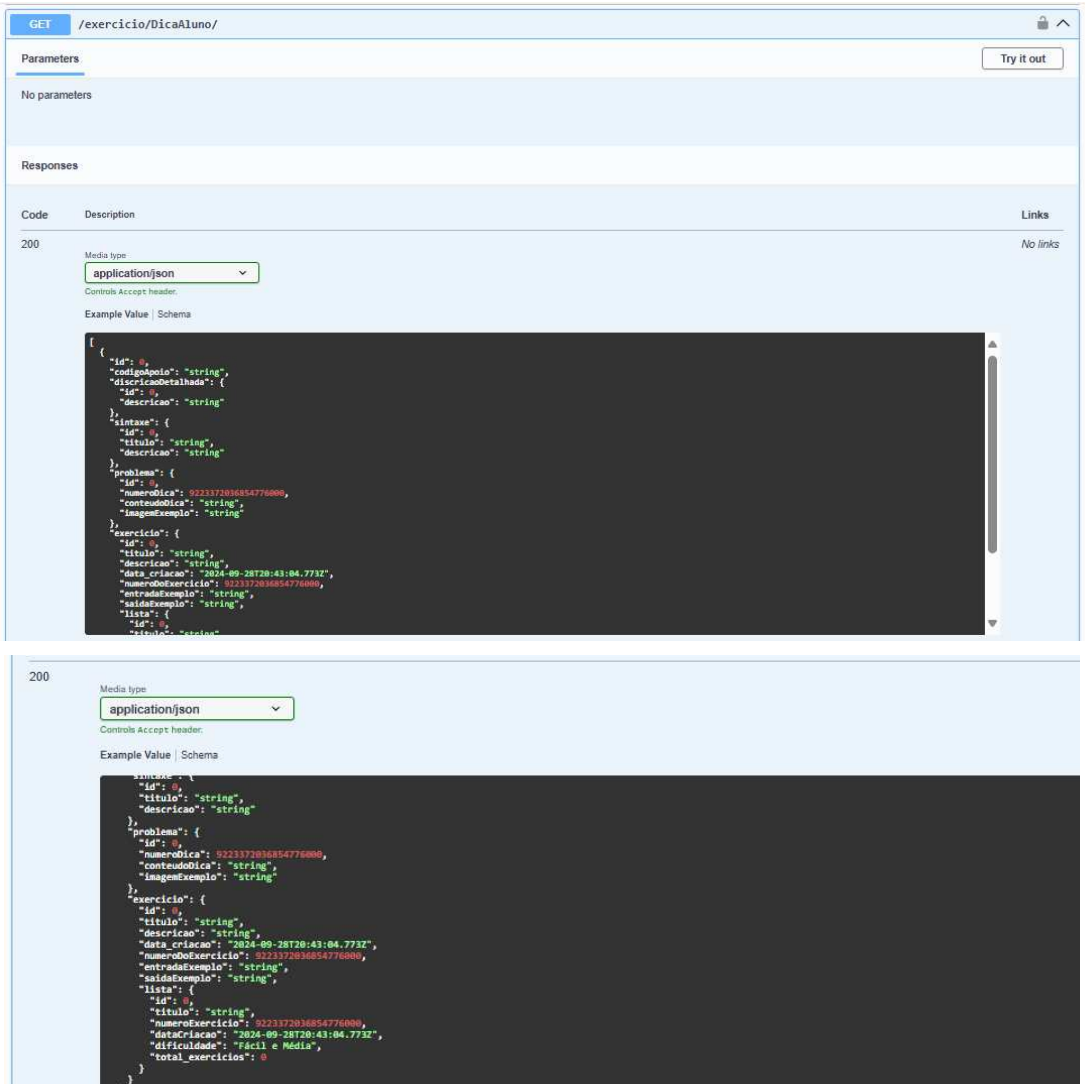
**Figura 17** - API - Interface para Responder Exercícios.




**Fonte**: Autoria própria(2024)

A Figura 18 apresenta a parte da API referente à funcionalidade de exibição das dicas para o aluno. Esse *endpoint* utiliza o método GET para fornecer todos os dados relacionados às dicas, incluindo informações sobre sintaxe, descrição detalhada, códigos de apoio e problemáticas associadas. Essa funcionalidade visa auxiliar os alunos no entendimento e na resolução de suas dúvidas, promovendo um aprendizado mais eficiente e orientado.

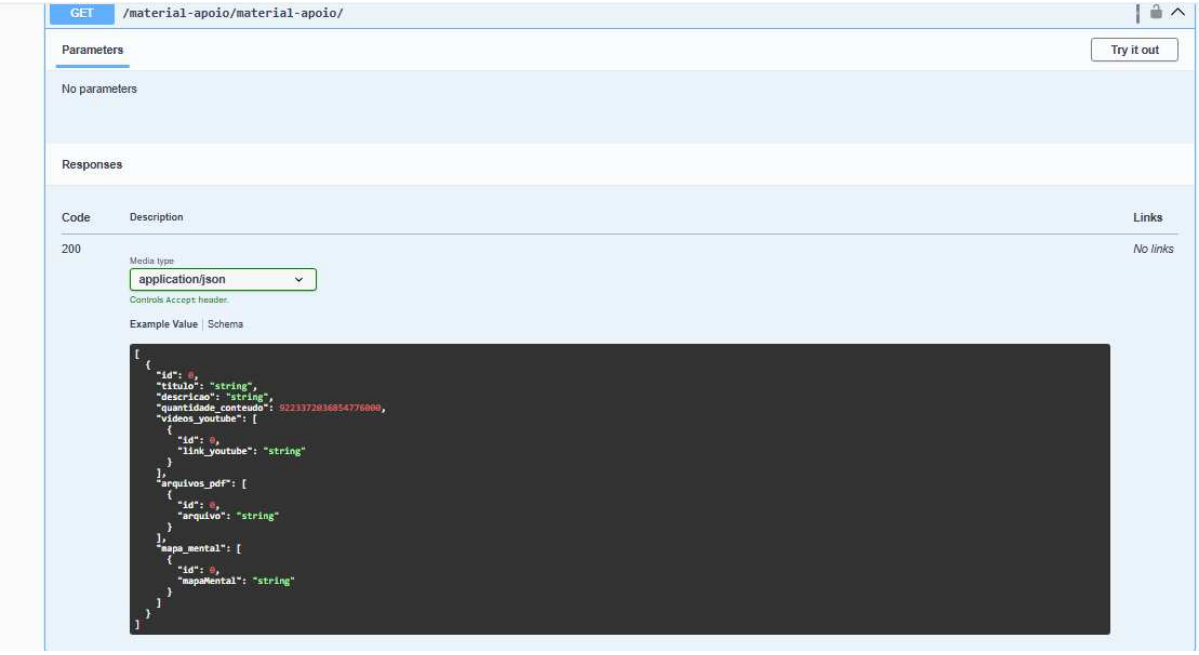
**Figura 18** - API - Interface para Dica do Exercício.




**Fonte**: Autoria própria(2024)

A Figura 19 apresenta a parte da API relacionada à funcionalidade de exibição dos materiais de apoio. Essa funcionalidade abrange recursos diversos, como vídeos do YouTube, livros em formato PDF e mapas mentais, permitindo que os alunos acessem facilmente uma variedade de conteúdos educativos. A apresentação organizada desses materiais facilita a consulta e o aprendizado, tornando as informações mais acessíveis e contribuindo para uma melhor experiência do usuário.

**Figura 19** - API - Interface de Consulta de Material de Apoio.




**Fonte**: Autoria própria(2024)

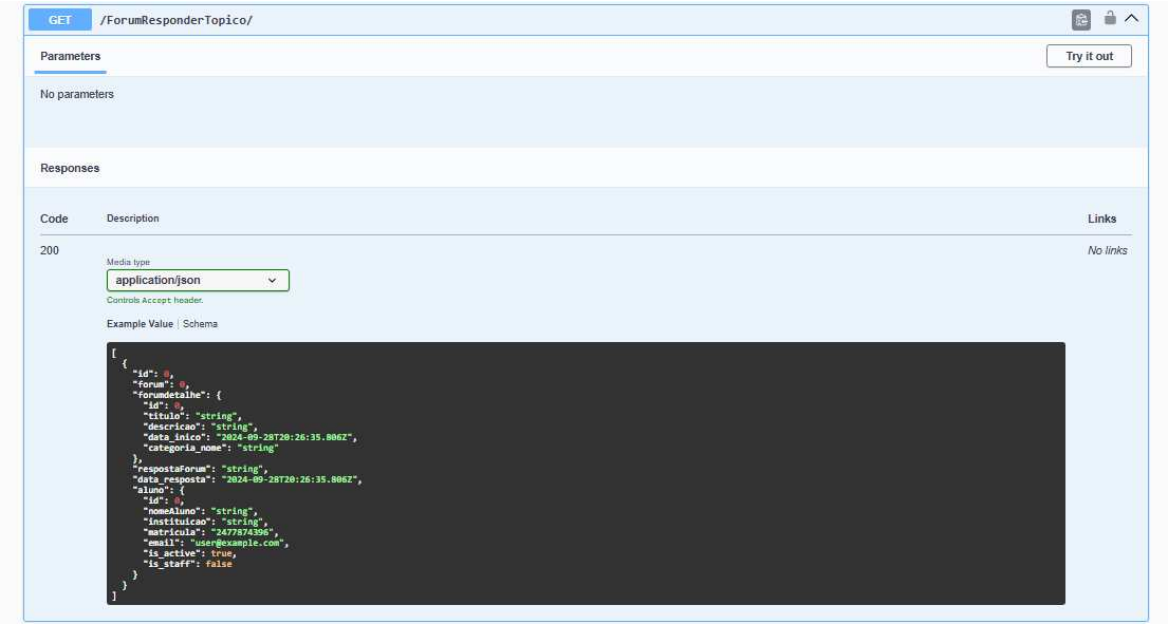
A Figura 20 apresenta a documentação referente à funcionalidade de criação de um tópico no fórum, a qual é implementada por meio de um método POST. Nessa funcionalidade, qualquer aluno cadastrado e autenticado no sistema tem a permissão de criar um novo tópico de discussão dentro do fórum.

**Figura 20** - API - Interface para Criar Forum.


**Fonte**: Autoria própria

A Figura 21 apresenta a parte da API referente à funcionalidade de exibição de todos os fóruns criados pelos usuários. Essa funcionalidade é implementada através do método GET, permitindo que qualquer aluno autenticado no sistema visualize os tópicos de discussão previamente criados no fórum.

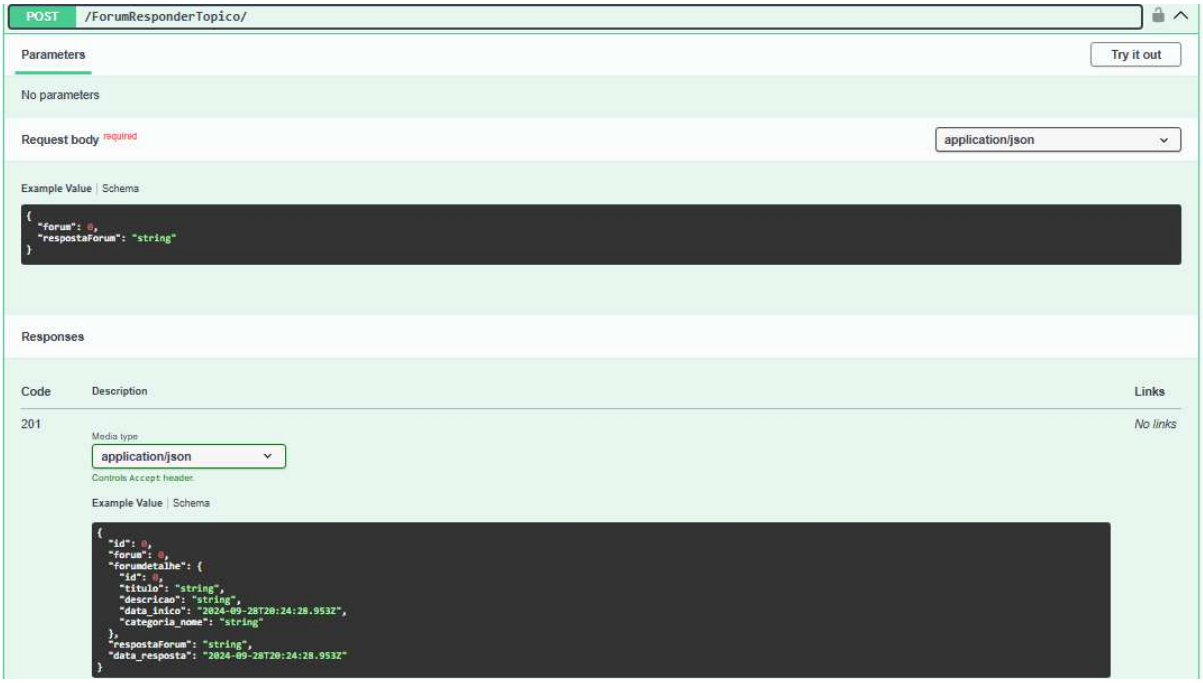
**Figura 21** - API -Interface de Exibição de Fóruns.




**Fonte**: Autoria própria (2024)

A Figura 22 apresenta a parte da API sobre a funcionalidade de resposta a um tópico no fórum, acessada pelo método POST. Para registrar uma resposta, o ID do tópico e o conteúdo da resposta devem ser enviados. Após a validação da autenticidade do usuário, o sistema armazena a resposta, associando-a ao tópico. Essa funcionalidade facilita a interação entre os alunos, promovendo o diálogo e a resolução de dúvidas no fórum.

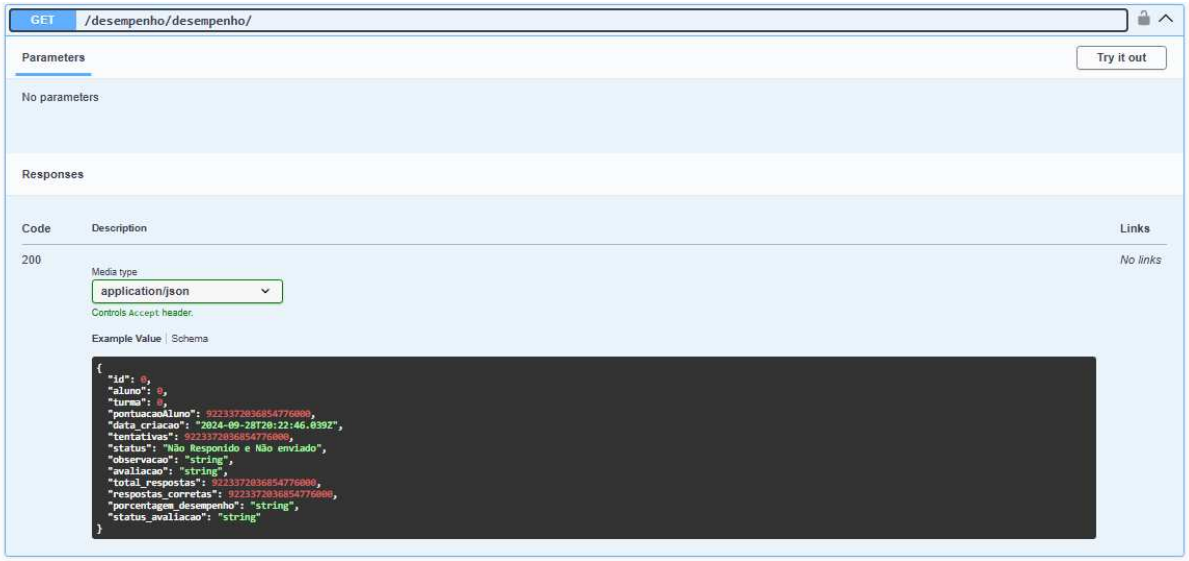
**Figura 22** - API - Interface  para Responder a Tópicos no Fórum.




Fonte: Autoria própria

A Figura 23 apresenta a parte da API referente à funcionalidade de consulta de desempenho. Esse *endpoint* permite exibir todas as informações relacionadas ao desempenho do aluno, incluindo a verificação de seu progresso por meio de porcentagens, além de outras métricas relevantes. A funcionalidade oferece uma visão detalhada do desempenho ao resolver exercícios, permitindo que o aluno acompanhe seu progresso e identifique áreas de melhoria.

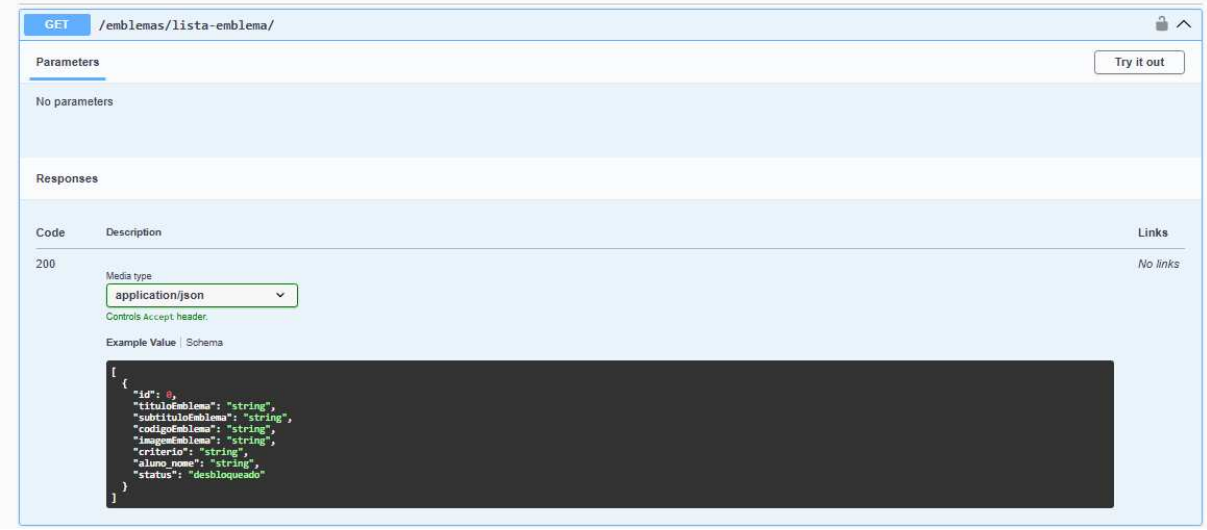
**Figura 23** - API - Interface para Visualizar Desempenho do Aluno.




**Fonte**: Autoria própria(2024)

A Figura 24 apresenta a parte da API da funcionalidade que exibe os emblemas desbloqueados pelo aluno. Esse *endpoint* é responsável por listar todas as informações dos emblemas que o aluno conseguiu desbloquear, proporcionando uma visão clara de suas conquistas no sistema.

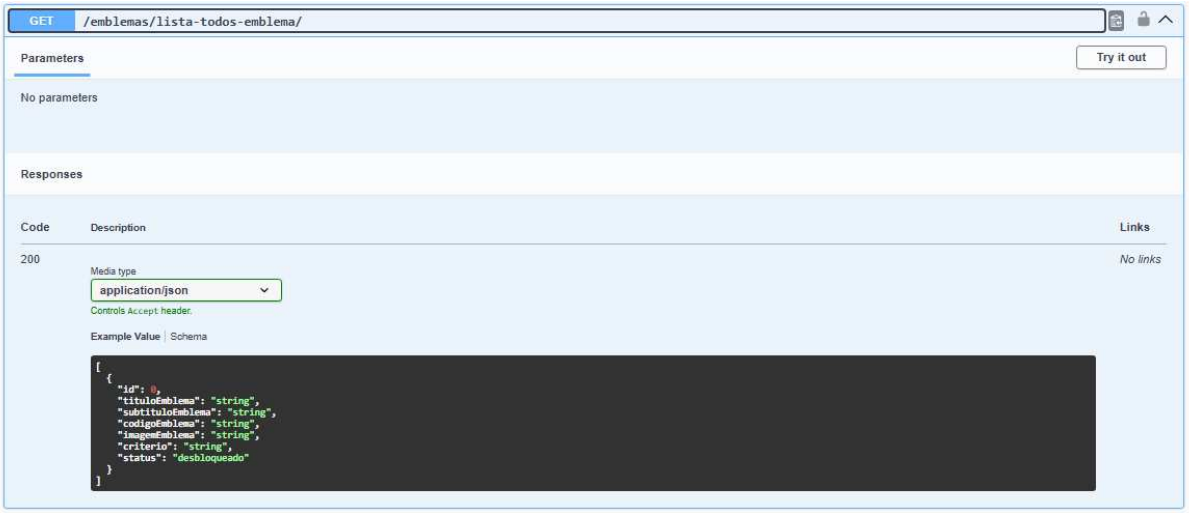
**Figura 24** - API - Interface para Visualizar Emblema Desbloqueado do Aluno.




**Fonte**: Autoria própria(2024)

A Figura 25 apresenta a parte da API da funcionalidade que exibe todos os emblemas disponíveis no sistema. Esse *endpoint* lista todos os emblemas cadastrados, mostrando seu status, indicando se foram desbloqueados ou não pelo aluno.

**Figura 25** - API - Interface para Visualizar Emblema Todos os Emblemas Disponível.




**Fonte**: Autoria própria(2024)

## 4.3 CONSIDERAÇÕES SOBRE A API

Com relação ao número de reprovações na disciplina de Algoritmos, as pesquisas realizadas indicam que o Loop Academic possui o potencial de obter resultados positivos, uma vez que o sistema pode motivar os alunos a praticarem mais programação por meio da resolução de exercícios. Essa prática contínua é fundamental para melhorar a compreensão dos conceitos e, consequentemente, reduzir os índices de reprovação, especialmente entre os estudantes que estão no início do curso.

No entanto, é importante ressaltar que o sistema precisa ser validado por meio de coletas de *feedback* dos usuários. Essa etapa é essencial para garantir que o software atenda às necessidades dos estudantes e facilite o processo de ensino-aprendizagem. Além disso, uma bateria de testes extensivos é necessária antes que o sistema esteja completamente pronto para uso pelos usuários finais. Portanto, a continuidade no processo de análise e planejamento, aliada à futura coleta de *feedbacks* dos usuários e a implementação de melhorias iterativas, será essencial para garantir o sucesso e a eficácia do Loop Academic no apoio ao ensino de programação.

Diante disso, uma das etapas necessária e de grande importância refere-se ao feedback dos usuários e à realização de testes. Essas ações são essenciais para garantir um sistema funcional, especialmente para a comunidade universitária.

## 5. CONSIDERAÇÕES FINAIS

Ao longo do projeto foi proposta uma arquitetura de software para o ambiente educacional Loop Academic. As visões essenciais da aplicação foram modeladas, possibilitando uma visualização de diferentes perspectivas dos elementos do software. Para avaliar a arquitetura foram utilizadas duas abordagens: a primeira delas referente a uma avaliação baseada em cenários, para levantar como os elementos interagem para implementar as funcionais do software; a segunda abordagem consistiu na implementação do *back-end* da plataforma, possibilitando a integração futuras, com o *front-end* do ambiente. No desenvolvimento deste projeto, buscou-se integrar o *front-end* com o *back-end* do sistema Loop Academic, visando testar e implantar este sistema na Universidade Federal Rural do Semi-Árido, Campus Pau dos Ferros. A necessidade de apoiar os calouros, que enfrentam grandes dificuldades em aprender programação, tornou-se evidente diante do elevado número de reprovações na disciplina. O foco do projeto foi desenvolver apenas um módulo inicial, buscando oferecer uma solução que potencialmente proporcionasse alguma experiência positiva para esses alunos em início de curso.

Embora o sistema ainda não esteja completo, contando apenas com o módulo de alunos, não tenha sido implantado na universidade, nem coletado feedback dos usuários finais. Portanto, apesar da implementação final e a coleta de feedback dos usuários ainda estejam pendentes, a fase de planejamento e desenvolvimento do sistema Loop Academic já representa um avanço significativo na modernização do ensino de programação para as universidades que oferecem curso relacionado com programação. O desenvolvimento desse módulo inicial, voltado especificamente para apoiar os ingressantes, revela o potencial da solução em reduzir as dificuldades enfrentadas nas disciplinas de programação, proporcionando uma experiência de aprendizado acessível.

## 5.1 TRABALHOS FUTURAS

A implementação do Loop Academic na UFERSA apresenta uma ótima oportunidade para apoiar os alunos que enfrentam dificuldades na disciplina de Algoritmos. Como destaca Souza (2019), essa disciplina é frequentemente considerada desafiadora e resulta em altas taxas de reprovação. Com o Loop Academic, espera-se incentivar o estudo da programação, desenvolvendo a lógica essencial para a disciplina de maneira interativa. Além de oferecer recursos como exercícios e videoaulas, a plataforma permitirá que os alunos enviem dúvidas de forma anônima, o que pode aumentar significativamente seu engajamento.

Com essas iniciativas, espera-se que o número de alunos aprovados aumente e que as taxas de reprovação diminuam consideravelmente, proporcionando um aprendizado mais sólido e inclusivo na universidade. Em perspectivas futuras, o Loop Academic poderá ser expandido para outras disciplinas, beneficiando alunos com dificuldades em aprender exclusivamente com aulas tradicionais. Para isso, a ferramenta deve passar por melhorias e adaptações, tornando-se versátil para diferentes áreas de conhecimento.

Além disso, a plataforma deverá evoluir com novos modelos de usuário, incluindo perfis específicos para professores e monitores, o que permitirá um suporte ainda mais direcionado. A inclusão desses perfis possibilitará que os alunos recebam assistência de monitores e feedback dos professores diretamente na plataforma, promovendo um ambiente colaborativo e integrador para o aprendizado contínuo. Com essas futuras implementações, o Loop Academic estará bem posicionado para se tornar uma ferramenta essencial no apoio ao ensino e aprendizado dentro das universidades.

## REFERÊNCIAS

ALURA. Django QuerySets e ORM: aprenda a manipular consultas em banco de dados. Disponível em: https://www.alura.com.br/artigos/django-query-sets-e-orm. Acesso em: 28 out. 2024.

AWARI. Python vs outras linguagens: entenda as diferenças. 2023. Awari. Disponível em: https://awari.com.br/python-vs-outras-linguagens-entenda-as-diferencas. Acesso em: 24 set. 2024.

BASTIAN, Sara. Learning Django REST Framework and MVT Architecture. Geek Culture, Medium, 21 jun. 2021. Disponível em: https://medium.com/geekculture/learning-django-rest-framework-and-mvt-architecture-1ca173163f1c. Acesso em: 29 out. 2024.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. Software architecture in practice (3rd ed.). Addison-Wesley Professional, 2012.

DOCUMENTO de Arquitetura – Integra Vendas. GitHub Pages, 2018. Disponível em: https://fga-eps-mds.github.io/2018.2-Integra-Vendas/docs/doc-arquitetura. Acesso em: 29 out. 2024.

ICLOUD. Tecnologia da web: definição e exemplos. iCloud. Disponível em: https://www.icloud.com.br/tecnologia-da-web-definicao. Acesso em: 27 set. 2024.

MOZILLA. Introdução ao Django - Aprendendo desenvolvimento web. MDN, 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Learn/Server-side/Django. Acesso em: 24 set. 2024.

ORACLE. O que é um banco de dados? Oracle, 2020. Disponível em: https://www.oracle.com/br/database/what-is-database/. Acesso em: 24 set. 2024.

SOUZA, Dyego Magno Oliveira. Desenvolvimento e avaliação do protótipo do Loop Academic: um software educacional para o auxílio no processo de ensino-aprendizagem de programação introdutória. 2019. Trabalho de Conclusão de Curso (Bacharelado em Tecnologia da Informação) – Universidade Federal Rural do Semi-Árido, Pau dos Ferros, 2019. Orientadora: Profa. Mãe. Jarbele Cássia da Silva Coutinho.

SOUZA, Mariana. Construção de APIs com Django REST Framework. Python Academy, 2019. Disponível em: https://pythonacademy.com.br/blog/construcao-de-apis-com-django-rest-framework. Acesso em: 7 out. 2024.

TREINA WEB. O que é o Django Rest Framework. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-django-rest-framework. Acesso em: 24 set. 2024