UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE RUSSAS

CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

PEDRO HENRIQUE DE SOUSA ALCÂNTARA

UM ESTUDO DE CASO DE DESENVOLVIMENTO DO *BACK-END* DO AMO (AMBIENTE DE MONITORIA ON-LINE)

RUSSAS

2025

PEDRO HENRIQUE DE SOUSA ALCÂNTARA

UM ESTUDO DE CASO DE DESENVOLVIMENTO DO *BACK-END* DO AMO (AMBIENTE DE MONITORIA ON-LINE)

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Orientadora: Prof. Dr. Patrícia Freitas Campos de Vasconcelos.

RUSSAS

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

A32e Alcântara, Pedro Henrique.
UM ESTUDO DE CASO DE DESENVOLVIMENTO DO BACK-END DO AMO (AMBIENTE DE MONITORIA ON-LINE / Pedro Henrique Alcântara. – 2025.
48 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Engenharia de Software, Russas, 2025.
Orientação: Prof. Dr. Patrícia Freitas Campos de Vasconcelos.

1. Desenvolvimento back-end. 2. Django REST Framework. 3. Projeto de extensão. 4. Monitoria acadêmica. 5. API REST. I. Título.

CDD 005.1

PEDRO HENRIQUE DE SOUSA ALCÂNTARA

UM ESTUDO DE CASO DE DESENVOLVIMENTO DO *BACK-END* DO AMO (AMBIENTE DE MONITORIA ON-LINE)

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software do Campus de Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Software.

Aprovada em: 06/03/2025.

BANCA EXAMINADORA

Prof. Dr. Patrícia Freitas Campos de Vasconcelos (Orientadora)
Universidade Federal do Ceará (UFC)

Profa. Ms. Valéria Maria da Silva Pinheiro
Universidade Federal do Ceará (UFC)

Profa. Dra. Janete Pereira do Amaral
Centro Universitário Maurício de Nassau
(UNINASSAU)

AGRADECIMENTOS

A Deus, pelo dom da vida.

Aos meus pais, Fabio Alcântara e Maria Aparecida de Sousa, por tudo e por todo apoio que me deram e dão para eu seguir minha graduação.

À minha namorada, Camila Fernanda Leite de Jesus, por todo apoio, suporte e noites e madrugadas adentro reescrevendo e revisando o texto. "Com você eu faria 20 TCC's."

À minha orientadora, Patrícia Freitas Campos de Vasconcelos, por ter me guiado neste trabalho.

RESUMO

Tendo em vista a necessidade de melhoria da comunicação entre os alunos do ensino superior, monitores acadêmicos e professores, foi idealizado o desenvolvimento do software Ambiente de Monitoria Online (AMO) pelo Projeto de Apoio ao Ensino (PAE) da Universidade Federal do Ceará (UFC) campus de Russas. A aplicação tem por objetivo estabeler mais uma forma de contato entre os membros do campus de Russas, para que os alunos possam tirar dúvidas sobre as disciplinas e aprimorar seu aprendizado. A ferramenta permite a realização de comentários em fóruns e agendamentos de atendimento com os monitores. O presente trabalho tem como propósito apresentar o desenvolvimento do *back-end* do aplicativo, que consiste em uma *Application Interface Program* (API) *Representational State Transfer* (REST). Para a realização do trabalho, foi necessário analisar o documento de requisistos do sistema, que consta todas as funcionalidades do mesmo. Para atender aos requisitos, o desenvolvimento do sistema contou com uso da linguagem de programação Python, utilizando o *Django Rest Framework* (DRF) para construir a API com uma arquitetura *Model View Controller* (MVC). Após a etapa de desenvolvimento, foram realizados testes para verificar se as funcionalidades foram implementadas corretamente e se estão retornando os dados corretos e esperados. O aplicativo *mobile* foi lançado para a comunidade acadêmica russana em fevereiro de 2025. Ao receber feedback dos usuários ou houver necessidade de melhorias, o aplicativo será atualizado, garantindo boa usabilidade e eficiência para os seus usuários.

Palavras-chave: monitoria acadêmica; projeto de extensão; desenvolvimento *back-end*; Django REST Framework; API REST.

ABSTRACT

Considering the need to improve communication between higher education students, academic monitors, and professors, the development of the Ambiente de Monitoria Online (AMO) software was conceived by the Projeto de Apoio ao Ensino (PAE) at the Federal University of Ceará (UFC), Russas campus. The application aims to establish an additional means of contact among campus members, allowing students to ask questions about topics within their courses and enhance their learning. The tool enables users to post comments in forums and schedule meetings with academic monitors. This work aims to present the development of the application's back-end, which consists of a Representational State Transfer (REST) Application Programming Interface (API). To carry out this work, it was necessary to analyze the system requirements document, which contains all its functionalities. To meet these requirements, the system was developed using the Python programming language, with Django Rest Framework (DRF) to build the API following a Model-View-Controller (MVC) architecture. After the development phase, tests were conducted to verify whether the functionalities were correctly implemented and if they returned the expected and accurate data. The mobile application was launched for the academic community of Russas in February 2025. Upon receiving user feedback or identifying the need for improvements, the application will be updated to ensure good usability and efficiency for its users.

Keywords: academic monitoring; extension project; back-end development; Django REST Framework; REST API.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AMO	Ambiente de Monitoria Online
API	*Application Interface Program*
DRF	*Django Rest Framework*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
MVC	*Model View Controller*
OMS	*Organização Mundial de Saúde*
ORM	*Object-Relational Mapping*
PAE	Projeto de Apoio ao Ensino
RAM	*Random Access Memory*
REST	*Representational State Transfer*
SaaS	*Software as a Service*
UFC	Universidade Federal do Ceará
URL	*Uniform Resource Locator*
WEB	*World Wide Web*
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, soluções em software estão se tornando cada vez mais presentes nas nossas vidas. Em seu livro *'The Technical and Social History of Software Engineering'*, Jones (2013) mostra que o advento do surgimento e polularização dos computadores pessoais na década de 1980 contribuiu pra uma enorme expansão do software para uso pessoal: softwares de escritório, calendários e finanças.

Durante a década de 1980, as soluções eram limitadas aos computadores e geralmente utilizados em empresas e fábricas. Após o surgimento de aparelhos celulares inteligentes — os smartphones —, começaram a surgir mais soluções para problemas, melhorias na qualidade de vida e facilitação de atividades diárias.

Em todos os âmbitos da vida — escola, trabalho, atividades de compra, faculdade — soluções de software deixam sua marca. Sempre facilitando uma atividade que antes podia exigir mais pessoas, tempo e esforço.

Queirós e Portela (2020) afirmam que *front-end* contempla tudo aquilo que é processado e exibido no navegador. Porém, eles complementam, o desenvolvimento para a *World Wide Web* (WEB) mudou radicalmente nos últimos anos: um bom exemplo é o JavaScript, há 10 anos, esta linguagem era usada somente para validar formulários, hoje em dia, extravasou os limites do navegador e pode ser usada em quase todos os cenários e plataformas para o desenvolvimento WEB (*front-end* e *back-end*), *desktop* ou *mobile*. O *back-end*, segundo Queirós e Portela (2020), consiste na programação WEB do lado do servidor e é responsável por implementar toda a camada de lógica de negócio de uma aplicação.

Este trabalho tem por objetivo documentar o processo de desenvolvimento de um *back-end* de um aplicativo *mobile* idealizado durante a execução de um projeto de extensão da UFC, o PAE.

O PAE, visando apoiar a realização das atividades relacionadas à monitoria pela comunidade acadêmica russana, disponibilizou um questionário para os estudantes da UFC do campus de Russas. O questionário consistia em o estudante descrever quais eram as principais dificuldades com a monitoria acadêmica e quais funcionalidades que ele acharia necessárias para uma aplicação *mobile* com o foco em melhorar a experiência dos estudantes ao utilizar da monitoria acadêmica. O resumo das respostas do questionário está disponível no Anexo - A. Com o resultado do questionários, além de percebida, foi comprovada a necessidade de uma solução em software — aplicação *mobile*, neste caso — para auxiliar os estudantes ao utilizarem

da monitoria acadêmica.

Este trabalho tem como objetivo documentar a solução — *back-end* — em software desenvolvida no PAE, com o intuito de melhorar a experiência de monitoria acadêmica na UFC do campus de Russas. A solução consiste no desenvolvimento *back-end* (API) para um aplicativo *mobile* chamado AMO.

2 OBJETIVOS

2.1 Objetivo geral

Desenvolver um *back-end* que forneça as funcionalidades necessárias para o funcionamento do sistema AMO.

2.2 Objetivos específicos

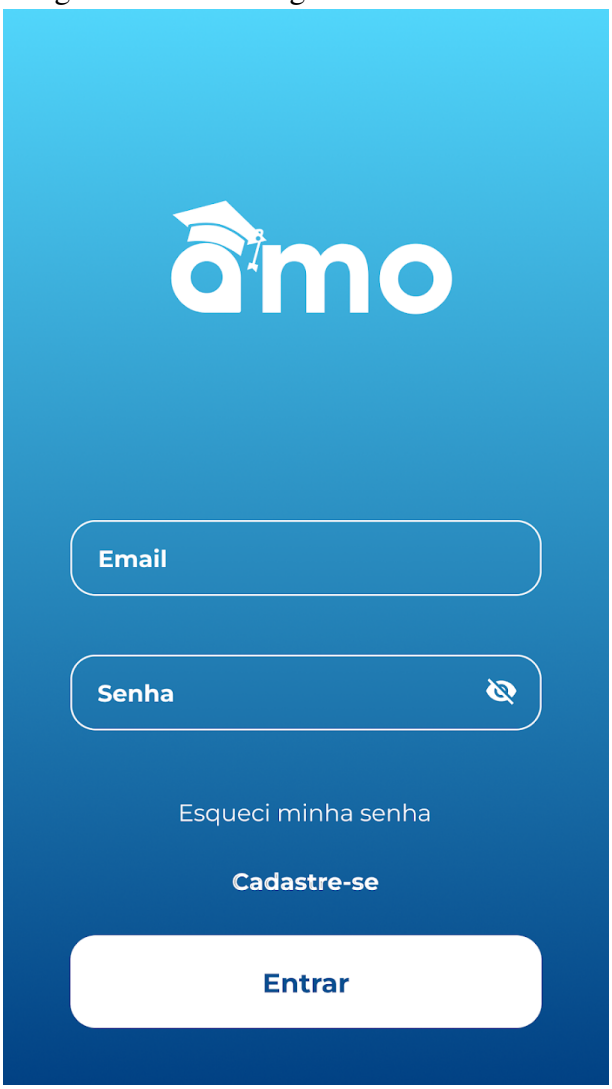
- Desenvolver uma API REST que possibilite a comunicação entre o aplicativo *mobile* e o servidor.
- Implementar funcionalidades essenciais para o funcionamento da aplicação.
- Documentar o processo de desenvolvimento do *back-end* dentro do PAE.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Contexto da aplicação

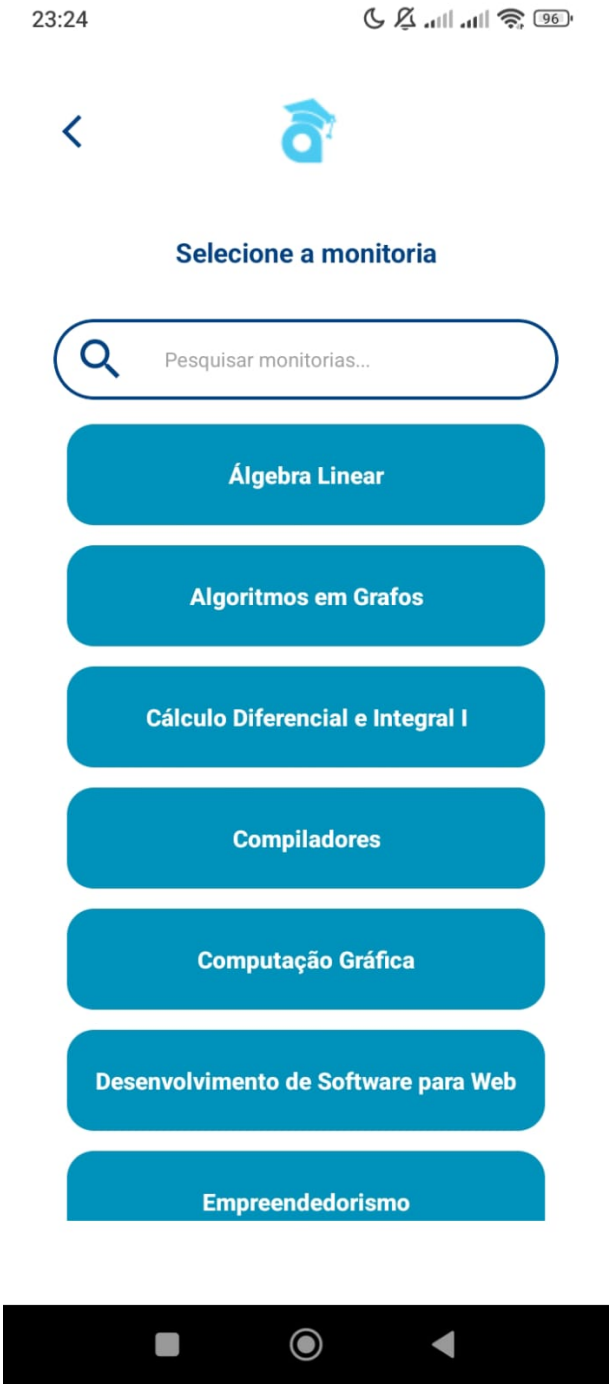
O AMO é um aplicativo *mobile* que visa auxiliar a monitoria acadêmica dentro da UFC do campus de Russas. O público-alvo da aplicação são alunos, monitores e professores. Com o aplicativo AMO, um aluno poderá fazer o seu próprio cadastro dentro da plataforma e uma vez realizado o *login*, ele poderá postar dúvidas dentro do fórum do aplicativo, onde outros alunos, monitores ou professores poderão o ajudar, fomentando uma discussão e troca de aprendizados dentro da aplicação. O aluno poderá também agendar um horário com um monitor de uma disciplina, podendo optar por atendimento remoto ou presencial. Foram escolhidos três módulos para serem desenvolvidos para a versão *beta* da aplicação a ser lançada ao público-alvo. O lançamento do aplicativo ocorreu no dia 05/02/2025 na UFC do campus de Russas. Esses três módulos — cadastro de usuários, fórum da aplicação e agendamentos — estão descritos na metodologia. Abaixo nas Figuras 1, 2, 3 e 4 estão algumas telas do aplicativo AMO.

Figura 1 – Tela de login do AMO.



Fonte: Elaboração própria.

Figura 2 – Tela mostrando uma lista de fóruns para cada disciplina.

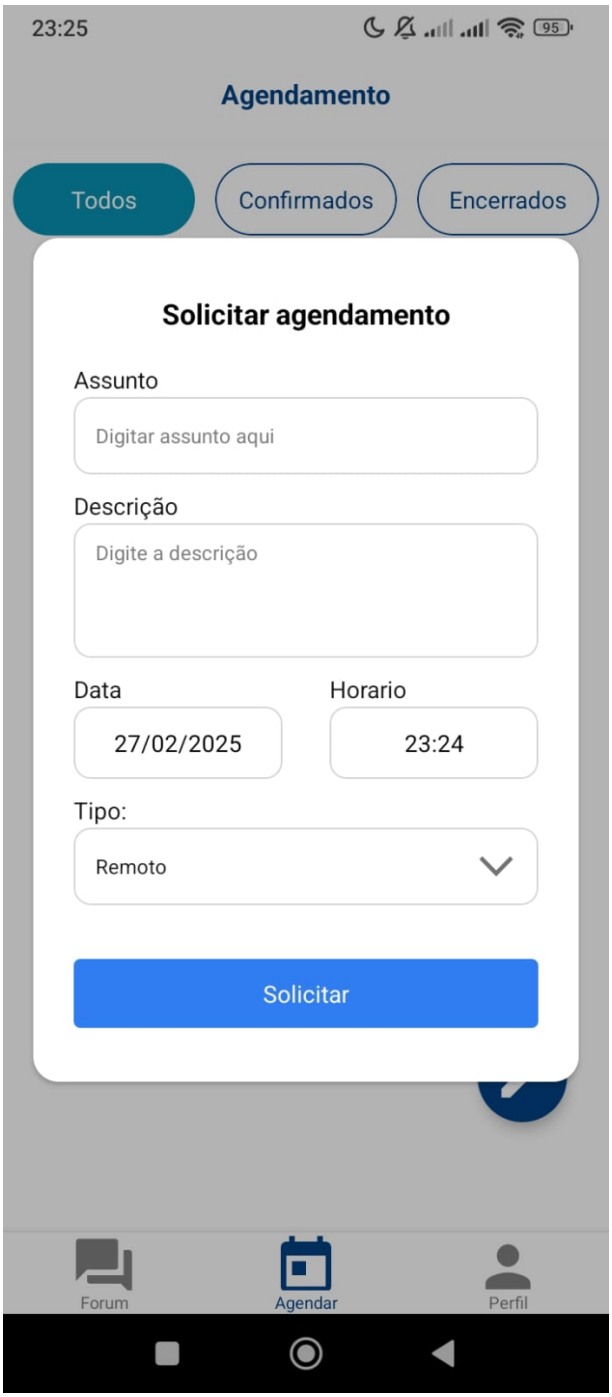


Fonte: Elaboração própria.

Figura 3 – Tela do fórum de uma disciplina.

Fonte: Elaboração própria.

Figura 4 – Tela para realização de um agendamento.



Fonte: Elaboração própria.

3.2 Linguagem de programação Python

Python é uma linguagem de programação multiparadigma criada na década de 1990. Ela é uma linguagem de alto nível amplamente utilizada em aplicações WEB, desenvolvimento de software, ciência de dados e *machine learning*, devido à sua simplicidade e versatilidade.

Ela é uma linguagem interpretada, imperativa, orientada a objetos, funcional — o que a torna uma linguagem que pode ser utilizada em vários paradigmas de programação —, de tipagem dinâmica e forte.

Criada em 1991 por Guido van Rossum, e desde o começo teve como norte um conjunto de príncipios — conhecidos como filosofia "Zen do Python-– alguns desses princípios são: "Simples é melhor que complexo"; "Legibilidade conta"; "Ao se deparar com a ambiguidade, se recuse a tentar adivinhar"(SOUZA, 2023).

Sobre o crescimento do Python, Souza (2023, p.12) diz que:

> Nos últimos anos, Python viu uma explosão de popularidade em diversas áreas. Seu ecossistema robusto e bibliotecas ricas tornaram-no uma escolha atraente para desenvolvimento web, automação, análise de dados, aprendizado de máquina e inteligência artificial. Frameworks populares como Django e Flask facilitaram o desenvolvimento de aplicativos web, enquanto bibliotecas como NumPy, pandas e matplotlib revolucionaram a análise de dados e a visualização.

Para o desenvolvimento do trabalho foi utilizada a linguagem Python. A escolha foi feita pois é uma linguagem de fácil aprendizado por possuir uma sintaxe clara, concisa e de fácil compreensão próxima à linguagem humana e muitos *frameworks* são construídos utilizando Python (AVARI, 2023). Além do mais, esta linguagem possui uma enorme comunidade ativa para suporte e uma vasta documentação das suas ferramentas e de como utilizá-las. No presente trabalho, o DRF foi utilizado para o desenvolvimento de uma API — sendo ela utilizada como *back-end* de uma aplicação *mobile*.

3.3 API REST

Conforme Massé (2011), os serviços da WEB são servidores projetados com o objetivo de atender às demandas específicas de um site ou de outra aplicação. Programas clientes se comunicam com esses serviços por meio de APIs. Uma API é um conjunto de serviços implementados em um programa de computador que são disponibilizados para que outros sistemas/aplicativos — aplicativos *mobile*, sendo o caso desse projeto — possam utiliza-los diretamente de forma simplificada, dessa forma, os aplicativos que consomem essa API, não precisarão se envolver com as implementações desses serviços — este isolamento entre estas partes é chamado de encapsulamento.

A arquitetura de uma API é cliente-servidor e segue um modelo de design de software no qual as funções são separadas entre dois componentes principais: cliente, responsável por solicitar serviços ou recursos (por exemplo, um navegador acessando uma página WEB) e

o servidor, responsável por processar as solicitações e retornar as respostas apropriadas (por exemplo, um servidor WEB enviando uma página *Hypertext Markup Language* (HTML)).

> Em termos gerais, uma API (Interface de Programação de Aplicações) expõe um conjunto de dados e funções para facilitar as interações entre programas de computador e permitir que eles troquem informações (MASSÉ, 2011, p. 5).

Já uma API REST, é uma API que segue o estilo arquitetural REST, no qual impõe restrições à forma que sistamas distribuídos — neste caso APIs — se comunicam, permitindo a criação de uma API com interfaces bem definidas — de fácil entendimento para os desenvolvedores. O *back-end* do aplicativo AMO possui uma arquitetura cliente-servidor e foi desenvolvido utilizando o estilo arquitetural REST por oferecer vantagens tais como protocolos HTTP bem definidos que, semânticamente, são compreensíveis e próximos da nossa realidade.

3.4 Métodos HTTP

O protocolo HTTP define um conjunto de métodos de requisição responsáveis por indicar a ação a ser executada para um dado recurso (DOCS, 2024). Esses métodos possuem nomes de verbos na lingua inglesa, alguns deles são: *GET* (o usuário pede um dado ao servidor); *PUT* (o usuário envia um dado ao servidor); *DELETE* (o usuário exclui um dado do servidor). Após a execução de um método, o servidor retorna um ou mais dados e informações sobre o status dessa solicitação: informando se ocorreu tudo bem com essa solicitação. São através desses métdos que ocorre a comunição com a API.

3.5 Django REST Framework

DRF é um conjunto de ferramentas — o que implica na própria definição de *framework*, segundo AWS (2024):

> Em engenharia e programação de software, um *framework* é uma coleção de componentes de software reutilizáveis que tornam mais eficiente o desenvolvimento de novas aplicações."

É possível criar APIs robustas e seguras, utilizando o DRF. O DRF foi desenvolvido utilizando o *framework* Django como coluna principal. Ao contrário do DRF, que sua biblioteca oferece ferramentas para a criação de uma API, o Django está mais voltado para o desenvolvimento WEB como um todo, desde lidar com as rotas URL, renderização de templates, gerenciamento de bancos de dados, entre outras (APIDOG, s.d.). No trabalho realizado, a API foi desenvolvida utilizando o DRF.

3.6 Arquitura *Model-View-Controller*

O padrão arquitetural MVC é uma abordagem amplamente utilizada no desenvolvimento de software, especialmente em aplicações WEB. Essa arquitetura promove a separação de preocupações, permitindo que diferentes componentes do sistema sejam desenvolvidos, testados e mantidos de forma independente. Segundo Gamma *et al.* (1995), uma aplicação possui três componentes principais:

- *Model* (Modelo): é responsável pela lógica de dados e regras de negócio. O *model* interage com a base de dados, gerenciando a criação, leitura, atualização e exclusão de dados. Ele também representa o estado da aplicação e notifica a *view* sobre quaisquer alterações.
- *View* (Visão): representa a interface do usuário. A *view* é responsável pela apresentação dos dados ao usuário, geralmente em forma de interfaces gráficas ou páginas WEB. Em aplicações WEB, a *view* é responsável por renderizar o conteúdo de acordo com os dados fornecidos pelo *model*.
- *Controller* (Controlador): atua como um intermediário entre o *model* e a *view*. O controlador processa as entradas do usuário, invoca as operações apropriadas no *model* e, em seguida, determina a *view* que deve ser exibida como resposta.

No contexto do DRF, a arquitetura pode ser entendida da seguinte forma:

- *Model* (Figura 5): o *model* no DRF funciona da mesma forma que no padrão MVC tradicional. Ele é responsável por interagir com a base de dados através do *Object-Relational Mapping* (ORM) do Django, permitindo que os desenvolvedores definam e manipulem a estrutura dos dados de forma intuitiva.

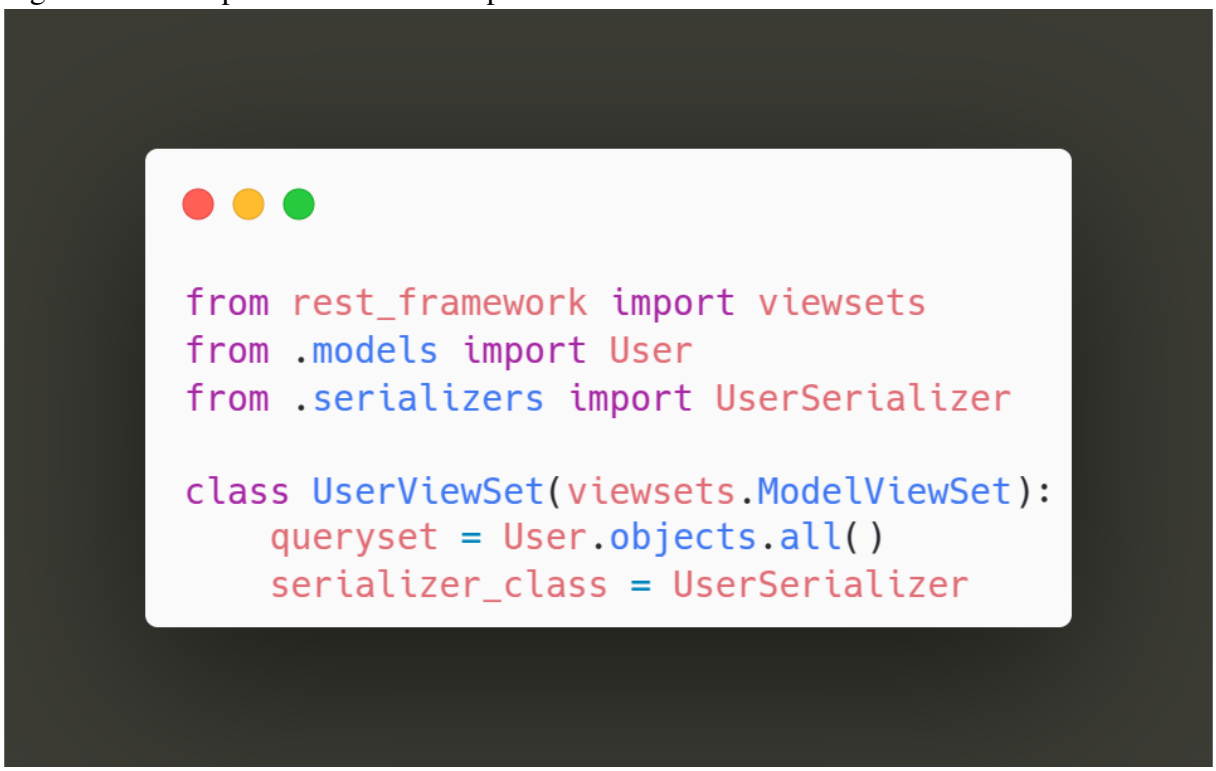
Figura 5 – Exemplo de um modelo de usuário

```python
from django.db import models

class User(models.Model):
    username = models.CharField(max_length=150, unique=True)
    email = models.EmailField(unique=True)
    password = models.CharField(max_length=256)

    def __str__(self):
        return self.username
```

Fonte: Elaboração própria.

– *Serializer* (Figura 6): o *serializer* desempenha um papel semelhante ao da *View* no padrão MVC. Ele é responsável por converter os dados do *model* em formatos adequados para a transferência via API, como *JavaScript Object Notation* (JSON) ou Extensible Markup Language (XML). Este também valida as entradas do usuário antes de serem processadas pelo *model*.

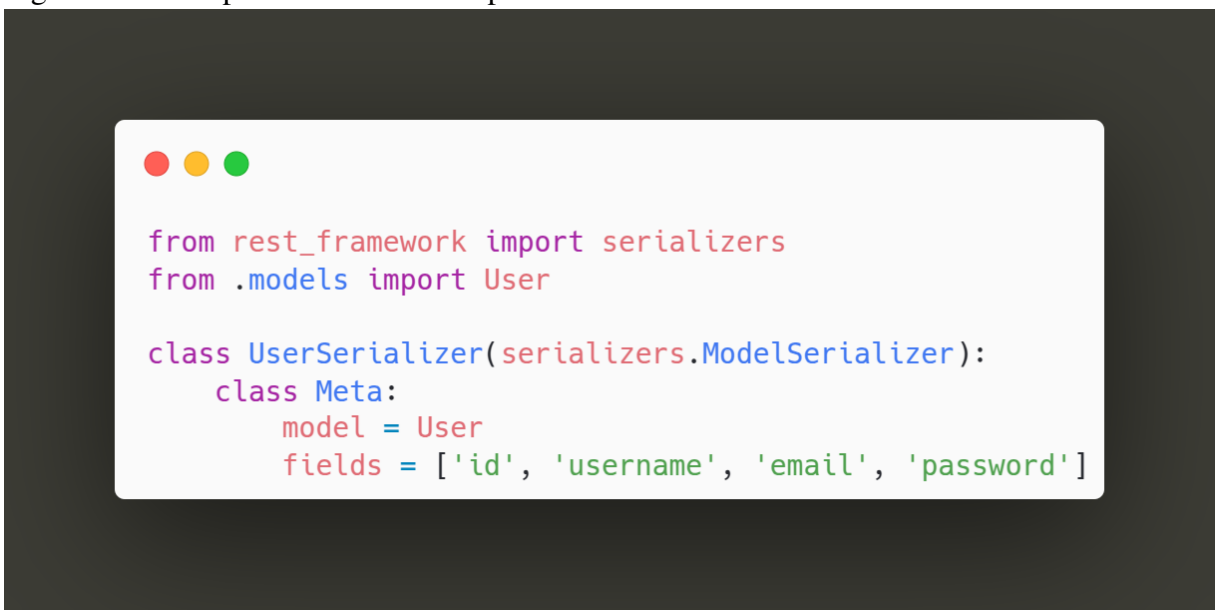
Figura 6 – Exemplo de uma ViewSet para usuários.



Fonte: Elaboração própria.

– *ViewSet/APIView* (Figura 7): o *ViewSet* ou *APIView* no DRF atua como o *controller*. Ele gerencia as requisições HTTP, direcionando as operações apropriadas (como listagem, criação, atualização ou deleção) para os *Models*, com base na lógica de negócio implementada. Além disso, o *ViewSet* determina qual *serializer* deve ser utilizado para formatar a resposta.

Figura 7 – Exemplo de uma ViewSet para usuários.



```python
from rest_framework import serializers
from .models import User

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = ['id', 'username', 'email', 'password']
```

Fonte: Elaboração própria.

A implementação no DRF seguiu essa estrutura, utilizando *models* para manipulação de dados, *ViewSets* para intermediar as requisições HTTP e *serializers* para converter os dados, garantindo uma API eficiente e bem estruturada para atender às necessidades da aplicação.

3.7 Git/Github

Git é um sistema de controle de versão que permite o desenvolvedor manter um registro das mudanças que ocorreram no código (ELLIS, 2022). Segundo Silverman (2013), as principais funcionalidades do Git são: possibilitar que o desenvolvedor examine o estado do projeto em pontos anteriores no tempo; mostrar as diferenças entre os estados do projeto e permitir que pessoas trabalhem simultaneamente, compartilhando e combinando seus códigos conforme necessário. Durante o desenvolvimento, funcionalidades ou novos pedaços de código são adicionados — ou removidos — e isso pode causar falhas no sistema, logo, é importante existir um *backup* — ponto de restauração — de um momento durante a vida do sistema no qual tudo estava funcionando corretamente, onde o desenvolvedor pode recorrer e voltar a este ponto. Esse é apenas uma das funcionalidades do Git. O Github é uma plataforma WEB de hospedagem on-line para manter os repositórios Git. Com mais de 35 milhões de projetos já hospedados, o GitHub é a principal ferramenta para gerenciamento de controle de versão online (COSENTINO *et al.*, 2017). Foi utilizado o Git para o versionamento do código do *back-end* da aplicação e o Github para hospedagem do código-fonte on-line possibilitando que os outros membros da

equipe de desenvolvimento possam trabalhar no sistema.

3.8 Visual Studio Code

O Visual Studio Code é um editor de código aberto desenvolvido pela Microsoft, oferecendo um ambiente altamente customizável, com recursos como depuração, destaque de sintaxe, correspondência de colchetes e um vasto ecossistema de extensões, que contribuem significativamente para aumentar a produtividade dos desenvolvedores (VIT, 2023). O Visual Studio Code foi utilizado como ambiente de desenvolvimento neste trabalho para a edição de código.

3.9 Insomnia

Insomnia é um aplicação *open-source* do lado do cliente que permite que desenvolvedores testem e interajam com uma API (INSOMNIA, n.d.). Por meio desta ferramenta, é possível testar *endpoints* da API, e verificar se está funcionando corretamente e retornando dados esperados. Neste trabalho, essa aplicação foi utilizada para a realização de testes, a fim de verificar se a funcionalidade implementada estava operando conforme o esperado, por meio das *endpoints* da API.

4 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados trabalhos e pesquisas realizadas por outros autores, nos quais foram propostas soluções semelhantes à deste trabalho. Essas propostas envolvem o desenvolvimento de softwares como resposta a problemáticas do mundo real.

4.1 6Leva

O trabalho de Melo (2023) apresenta o desenvolvimento do *back-end* para uma aplicação de fretes baseada na arquitetura *Software as a Service* (SaaS). O objetivo foi criar uma API eficiente, compatível com dispositivos *desktop* e *mobile*, garantindo boa lógica, funcionamento e comunicação. A pesquisa seguiu um estudo bibliográfico fundamentado em obras sobre padrões de design no desenvolvimento *back-end*.

> O objetivo principal do presente estudo é desenvolver e obter conhecimento das técnicas e conceitos necessários para criar o back-end (serviço responsável pelas operações do sistema do lado do servidor, no qual o usuário não tem acesso) de um sistema de fretes que usa a base arquitetônica de um software como serviço, ressaltando a importância de ter uma boa comunicação do cliente com o servidor, e ter um back-end escalável, seguro e robusto para que sua aplicação possa lidar com o crescimento do tráfego e das necessidades de negócio para que tudo funcione de maneira eficiente (MELO, 2023, p. 9).

Embora o objetivo de ambos os estudos seja semelhante — desenvolver um *back-end* eficiente para uma aplicação WEB — a principal diferença está na tecnologia escolhida para a implementação.

O sistema 6Leva utilizou Node.js com NestJS, que oferece uma estrutura modular e se baseia em TypeScript para proporcionar maior escalabilidade e organização no código. Este trabalho, por sua vez, utiliza Python com DRF.

4.2 OnTime

Em seu trabalho, Pinheiro (2021, p. 17) propõe uma solução em software para o atendimento em cantina da UFC do campus Pici, onde o atendimento presencial:

> Continua sendo a única forma de esses estabelecimentos lidarem com as filas, cujos problemas são agravados pelo baixo número de funcionários, o que resulta na demora no atendimento e na perda de tempo dos seus clientes, em sua maioria estudantes e professores, que não raramente desistem ou mudam de estabelecimento, visto que muitos não dispõem de tempo para esperar em filas, gerando perda de lucratividade e, também, má reputação, o que afasta mais clientes.

A partir desta problemática, o autor define o objetivo do trabalho como sendo a implementação um sistema *open-source* de compra e venda de produtos alimentícios, baseado em tecnologia WEB e acessível por dispositivos móveis. Para tal implementação, é desenvolvido uma API de um software. Pinheiro (2021) desenvolve a API utilizando a linguagem de programação Java, enquanto o presente trabalho a API é desenvolvida utilizando a linguaguem de programação Python.

4.3 CUIDAIDOSO

Erse (2021), propôs uma solução em software, dentro do projeto CuidaIdoso, para ajudar a disseminação de informções úteis — e também combater a proliferação das fake news por meio da tecnologia — durante o período da pandemia da COVID-19.

Segundo Erse (2021, p. 1):

> Neste cenário nasceu o projeto CuidaIdoso que tem o objetivo de disseminar informações confiáveis e verificadas por profissionais de saúde, sobre os cuidados que os idosos devem ter para se prevenir de doenças e obter uma melhor qualidade de vida por meio da tecnologia.

Os principais objetivos do software desenvolvido dentro do projeto CuidaIdoso são: orientar, de forma confiável, cuidadores e idosos à respeito da pandemia e cuidados com a saúde em geral; promover interação voluntária entre quem precisa de ajuda e quem pode ajudar e entreter os usuários das plataformas existentes.

> Neste cenário nasceu o projeto CuidaIdoso que tem o objetivo de disseminar informações confiáveis e verificadas por profissionais de saúde, sobre os cuidados que os idosos devem ter para se prevenir de doenças e obter uma melhor qualidade de vida por meio da tecnologia (ERSE, 2021, p. 1).

Para alcançar esses objetivos, o autor desenvolveu o *back-end* para um software onde é possível realizar os objetivos descritos anteriormente.

> Sua ideia central é promover a interação entre voluntários e idosos através da realização de atividades gravadas em forma de vídeos. Estes vídeos são fornecidos por voluntários cadastrados no aplicativo e possuem a finalidade de entreter os usuários da plataforma.(ERSE, 2021, p. 5).

O autor propõe uma solução para uma problemática — fake news sobre informações sobre saúde e bem-estar — percebida durante o período da pandemia, do começo de 2020 até 2023 quando foi decretado o seu fim pela *Organização Mundial de Saúde* (OMS) (FEDERAL, 2023).

Por fim, o autor desenvolve um segundo *back-end* com o objetivo de comparação e análise de métricas, buscando identificar as diferenças entre as duas ferramentas (*frameworks*) e determinar qual delas é mais eficiente para a resolução do mesmo problema. Ao contrário de Erse (2021), que emprega as linguagens de programação JavaScript e Go em conjunto com os *frameworks* Express.js (Javascript) e Gorrila Max (Go), este trabalho utiliza a linguagem de programação Python e o *framework* DRF.

4.4 Dyfocus

Cordeiro (2014, p. 10) apresenta seu trabalho desenvolvido na empresa Cheesecake Labs, que foi "criada com o preceito de desenvolver softwares inovadores, [tendo] seu foco principal em aplicações móveis, principalmente para as plataformas iOS (Apple) e Android (Google)" (CORDEIRO, 2014, p. 10).

Cordeiro (2014, p. 12) apresenta os objetivos como sendo o desenvolvimento de um:

> Aplicativo para iPhone, integrado a rede sociais (Facebook), no estilo Instagram, que seja capaz de tirar fotos com múltiplos pontos focais e filtros. A interface deve permitir que o usuário escolha quais esses pontos serão e, no formato final, seja apresentado um conjunto de imagens no estilo “GIF” que alterne as fotos, dando a impressão de ser apenas uma imagem dinâmica – daí a origem do nome, Dyfocus.

Este trabalho difere dos anteriores porque não propõe uma solução para um problema do mundo real, isto é, o trabalho realizado não tem por objetivo solucionar — ou melhorar um atual estado de coisas — um problema dentro de um contexto (social, acadêmico, pessoal), mas sim um software com objetivo conseguir um máximo de público e usuários possíveis e, talvez, obter um lucro com o software. Cordeiro (2014) utiliza a linguagem de programação Python e o *framework* Django, enquanto neste trabalho foi utilizado o *framework* DRF, uma extensão do Django voltada para o desenvolvimento de APIs REST.

4.5 Comparativo entre os trabalhos

Na Figura 8, está representados as diferenças e similaridades entre os trabalhos relacionados apresentados e o presente trabalho.

Figura 8 – Comparativo entre os trabalhos relacionados.

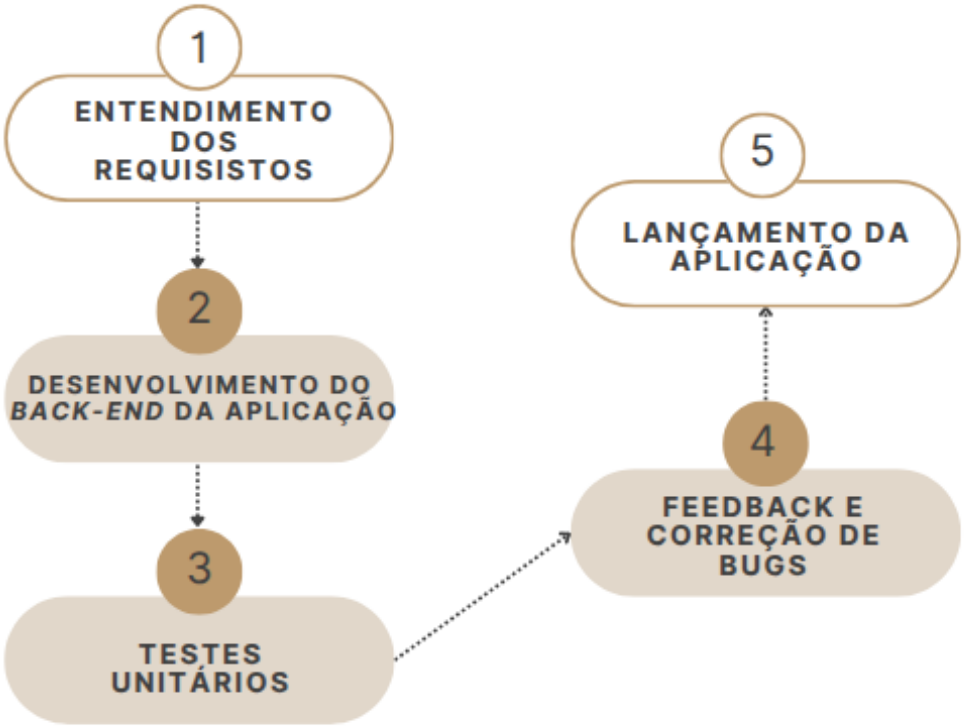
Referência	Objetivo da aplicação	Linguagem de programação	Framework	Plataforma
6Leva	Desenvolvimento de *back-end* para uma aplicação de fretes baseada em SaaS.	Node.js (TypeScript)	NestJS	Desktop, Mobile
OnTime	Sistema *open-source* de compra e venda de produtos alimentícios para a cantina da UFC campus Pici.	Java	Não especificado	Web, Mobile
Cuidaldoso	Disseminação de informações confiáveis sobre saúde para idosos e combate à proliferação de fake news.	JavaScript, Go	Express.js, Gorilla Max	Web
Dyfocus	Aplicativo para iPhone integrado a redes sociais, com múltiplos pontos focais e filtros para criar imagens dinâmicas	Python	Django	iOS
AMO	Desenvolvimento de API para um aplicativo que auxilia a monitoria acadêmica na UFC campus Russas	Python	DRF (Django Rest Framework)	Mobile

Fonte: Elaboração própria.

5 METODOLOGIA

A metodologia adotada para o desenvolvimento deste trabalho é composta por cinco etapas, conforme ilustrado na Figura 9.

Figura 9 – Representação da metodologia utilizada no desenvolvimento da API.



Fonte: Elaboração própria.

5.1 Entendimento dos requisitos

Para entender as funcionalidades do AMO, foi necessário analisar o documento de requisitos, que faz parte da documentação do aplicativo realizada pelo PAE. Os requisitos foram elicitados pela equipe do projeto em 2020 com a supervisão da coordenadora do PAE, a professora Dra. Patrícia Vasconcelos. Este documento descreve com detalhes o que o sistema desenvolvido deve possuir quanto às funcionalidades — *features*. Essas funcionalidades representam todas as ações que um usuário do sistema poderá realizar dentro da aplicação. Todo requisito possui uma prioridade inerente:

– Requisitos essenciais: são os fundamentais para o funcionamento do sistema. Sem eles, o sistema não pode operar corretamente. Exemplo: em um sistema bancário, a funcionalidade

de autenticação do usuário é essencial.

- Requisitos importantes: são dispensáveis, mas são muito necessários para o sucesso do sistema. O sistema pode funcionar sem eles, mas com impactos negativos na experiência do usuário ou eficiência. Exemplo: em um sistema de venda on-line, uma funcionalidade de recomendação de produtos pode ser importante para melhorar vendas, mas não impede a operação principal.
- Requisitos desejáveis: são opcionais, geralmente agregam valor adicional, mas não comprometem o sistema se não forem implementados. São os primeiros a serem descartados caso haja restrições de tempo ou orçamento. Exemplo: um modo escuro em um aplicativo pode ser desejável para alguns usuários, mas sua ausência não impede a usabilidade.

Abaixo serão apresentados os requisitos que foram desenvolvidos para compor o produto viável mínimo e estão divididos em três módulos: cadastro de usuários; fórum e agendamentos.

5.1.1 Cadastro de usuários

Por meio da funcionalidade cadastro de usuários é possível realizar o registro de novos usuários dentro da plataforma, permitindo que façam uso das demais funcionalidades do sistema. Na Figura 10, apresentada a seguir, o requisito é descrito como consta no documento de requisitos.

Figura 10 – Requsito de cadastro de aluno descrito no documento de requisitos

[RF007] Cadastro de aluno
Descrição: O usuário poderá se cadastrar no sistema informando sua matrícula, nome e e-mail.
Prioridade: ■ Essencial ☐ Importante ☐ Desejável
Entradas e pré-condições: Usuário autenticado.
Saídas e pós-condição: o usuário é cadastrado no sistema.

Fonte: Elaboração própria.

5.1.2 Fórum

No aplicativo, alunos, monitores e professores poderão interagir no fórum, trocando conhecimentos e compartilhando aprendizados. Dentro do fórum um aluno com dúvida poderá

publicá-la para que os demais usuários o ajudem, possibilitando um ambiente de debates e aprendizados. Abaixo serão apresentados alguns requisitos relacionados ao fórum.

5.1.2.1 Manter dúvida e reponder dúvida

As postagens dentro do fórum consistirão em dúvidas e respostas dos usuários. Portanto, a funcionalidade associada ao requisito de manter dúvidas desempenham um papel importante. Além de possibilitar ao usuário criar dúvidas dentro do fórum, é necessário também que usuários possam responder essas dúvidas, dessa forma gerando uma discussão sobre o assunto. Estas funcionalidades são apresentadas a seguir nas Figuras 11 e 12.

Figura 11 – Requsito de manter dúvidas descrito no documento de requisitos

[RF009] Manter dúvida
Descrição: O aluno, monitor e professor podem cadastrar, consultar, atualizar e excluir uma dúvida no fórum. No cadastro, deve ser informado o título, a tag e a descrição.
Prioridade: ■ Essencial ☐ Importante ☐ Desejável
Entradas e pré-condições: Usuário esteja autenticado;
Saídas e pós-condição: Uma dúvida no fórum deve ser inserido no sistema.

Fonte: Elaboração própria.

Figura 12 – Requsito de responder dúvidas descrito no documento de requisitos

[RF010] Responder dúvida
Descrição: O aluno, monitor e professor podem responder uma dúvida no fórum.
Prioridade: ■ Essencial ☐ Importante ☐ Desejável
Entradas e pré-condições: Usuário esteja autenticado; O tópico esteja cadastrado no fórum;
Saídas e pós-condição: Uma resposta para a dúvida no fórum deve ser inserida no sistema.

Fonte: Elaboração própria.

5.1.2.2 Filtrar dúvidas

Dentro do fórum, existirão inúmeras dúvidas, tendo isso em mente, é necessário haver uma forma de organizar e filtrá-las seguindo algum critério. Para tanto, foi desenvolvido a

funcionalidade de filtrar dúvidas, tendo como critério, por exemplo, dúvidas mais antigas. Esta funcionalidade está descrita na Figura 13.

Figura 13 – Requsito de filtrar dúvidas descrito no documento de requisitos

[RF012] Filtrar dúvidas
Descrição: O sistema deve fornecer opções de consultas rápidas (filtros) de dúvidas, como: respondidos (possui uma resposta correta), abertos, datas (crescente e decrescente), seguindo, respostas (com e sem), meus (os que o usuário criou), mais curtidos.
Prioridade: ☐ Essencial ■ Importante ☐ Desejável
Entradas e pré-condições: Usuários esteja autenticado;
Saídas e pós-condição: O sistema deve mostrar dúvidas de acordo com o filtro especificado;

Fonte: Elaboração própria.

5.1.3 Agendamentos

Dentro do AMO, um aluno pode solicitar um agendamento ao monitor, que ocorrerá de forma remota ou presencial. Este agendamento vai permitir que o aluno sane suas dúvidas sobre algum assunto da disciplina. Esta funcionalidade está descrita na figura 14.

Figura 14 – Requsito de solicitação de agendamentos descrito no documento de requisitos

[RF020] Solicitar um agendamento
Descrição: O aluno pode solicitar um agendamento presencialmente/virtualmente com o monitor, informando o assunto, descrição, horário, data e tipo (presencial ou virtual), especificar o monitor desejado (opcional).
Prioridade: ☐ Essencial ■ Importante ☐ Desejável
Entradas e pré-condições: Usuário esteja autenticado; O aluno deve está cadastrado em uma monitoria.
Saídas e pós-condição: Deve ser cadastrado um pedido de agendamento para os monitores ou para o monitor especificado caso houver.

Fonte: Elaboração própria.

O sistema AMO deve permitir que usuários possam visualizar os agendamentos feitos com um monitor. Esta funcionalidade está descrita na Figura 15.

O usuário deve ter a opção de filtrar todos o agendamentos, preferindo visualizar

Figura 15 – Requsito de visualizar agendamentos descrito no documento de requisitos

[RF021] Visualizar agendamentos

Descrição: O aluno pode visualizar todos os agendamentos (presencial ou virtual) dele ou todos os demais alunos da mesma monitoria.

Prioridade: ☐ Essencial ■ Importante ☐ Desejável

Entradas e pré-condições: Usuário esteja autenticado;

Saídas e pós-condição: Nenhuma ou visualização dos pedidos já realizados.

Fonte: Elaboração própria.

Figura 16 – Requsito de filtrar agendamentos descrito no documento de requisitos

[RF022] Filtrar agendamentos

Descrição: O sistema deve filtrar os agendamentos dos alunos com o monitor por data, nome do aluno, agendamentos confirmados e tags (conteúdos).

Prioridade: ☐ Essencial ■ Importante ☐ Desejável

Entradas e pré-condições: Usuário esteja autenticado;

Saídas e pós-condição: Nenhuma ou visualização dos pedidos já realizados.

Fonte: Elaboração própria.

apenas os presenciais ou remotos, os realizados ou pendentes. Esta funcionalidade está descrita na Figura 16.

5.2 Desenvolvimento do *back-end* da aplicação

Neste capítulo, serão descritas e documentadas as tecnologias e ferramentas utilizadas no desenvolvimento da aplicação, bem como sua contribuição para a obtenção dos objetivos deste trabalho. O sistema desenvolvido consiste em uma API REST estruturada com a arquitetura baseada no padrão MVC. que disponibiliza *endpoints* para permitir transações, manipulação e troca de dados pela aplicação *mobile*. Dentre os três módulos do sistema apresentados anteriormente, apenas o fórum não é uma entidade no contexto de banco de dados. Nesse caso, as dúvidas e respostas que o compõem são consideradas entidades do sistema e, portanto, serão representadas no banco de dados.

No Django, essas entidades são tratadas como *models*, que possuem atributos essenciais para sua definição. No módulo de agendamentos, por exemplo, foi necessário criar um *model* correspondente, contendo atributos como horário, tipo (remoto ou presencial) e monitor

responsável. O processo de desenvolvimento envolveu atividades e ferramentas aplicadas aos três módulos, mesmo que apresentem diferenças entre si. As principais ferramentas utilizadas foram o Visual Studio Code, para a edição do código, o Insomnia, para testes de *API*, e o Render, para hospedagem e *deploy* da aplicação. O trabalho foi realizado remotamente em um ambiente de desenvolvimento configurado em uma máquina com processador Intel i5 e 16 GB de memória *Random Access Memory* (RAM).

A abordagem adotada seguiu etapas fundamentais, iniciando pelo entendimento dos requisitos, seguido da implementação, realização de testes unitários e, por fim, o *deploy* da aplicação na plataforma Render. O documento de requisitos do sistema serviu como a principal referência, contendo todas as funcionalidades necessárias para garantir o funcionamento adequado do software.

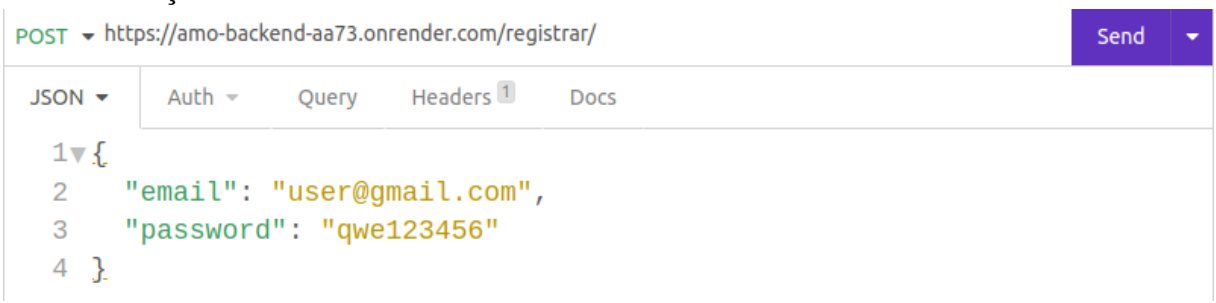
O DRF utiliza a biblioteca *models* do Django para a definição das entidades do banco de dados. Cada *model* foi projetado com atributos que o caracterizam, conforme especificado no documento de requisitos. Além disso, o DRF fornece um mecanismo próprio para serialização e desserialização, independente do Django. Esses processos são fundamentais para converter objetos para diferentes formatos, permitindo seu armazenamento e transmissão, além de facilitar a recuperação e reutilização dos dados. No sistema desenvolvido, os dados são armazenados em um banco de dados e precisam ser recuperados para serem transmitidos via *endpoints*, sendo representados em formatos como JSON ou XML. Esse processo permite que um usuário solicite informações de forma acessível. A serialização ocorre no momento em que um usuário envia dados por meio de uma *endpoint* da API, e a aplicação os processa para armazenamento no banco de dados.

5.3 Testes unitários

Os testes unitários constituem um método de verificação de software que analisa individualmente pequenas partes do código, como funções, métodos ou classes, garantindo que cada componente funcione corretamente de forma isolada. Eles ajudam a detectar erros precocemente, facilitam a manutenção do código e melhoram sua confiabilidade. Neste trabalho, após a implementação de uma *feature*, são obtidos os *endpoints* com os quais é possível realizar requisições ao servidor com métodos HTTP. Através dos *endpoints* são realizados testes unitários. Os testes foram realizados pela equipe de desenvolvimento *back-end*. Para tanto, foi utilizado o Insomnia como ferramenta principal, onde foi possível simular um usuário solicitando e enviando

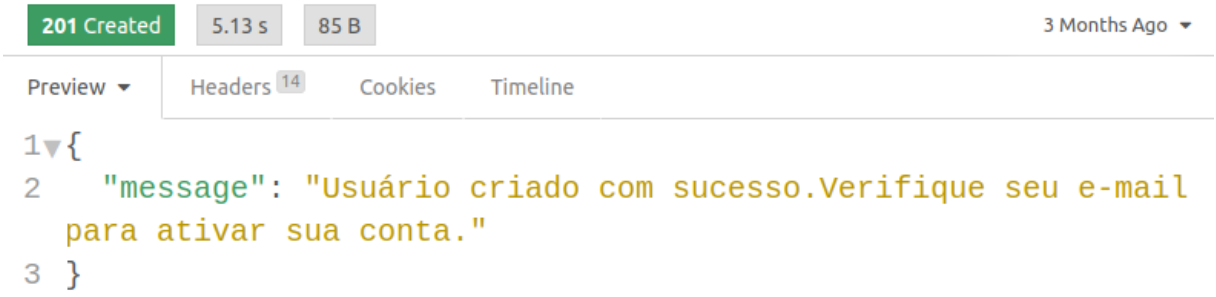
dados para a API e consequentemente recebendo dados de volta do servidor. Para a realização de testes, é necessário criar uma nova requisição do tipo *GET*, *POST*, *PUT* ou *DELETE*, dependendo do *endpoint* que deseja testar. É necessário também inserir a *Uniform Resource Locator* (URL) do *endpoint* da API que será testada. Dessa forma é possível verificar se os dados retornados são os esperados e se o *endpoint* em questão está acessível. A Figura 17 mostra a interface do Insomnia exemplificando uma requisição HTTP *POST* sendo enviada para um *endpoint* e a Figura 18 exibe a resposta HTTP, indicando que a requisição foi processada com sucesso.

Figura 17 – Utilizando o método HTTP POST no Insomnia para testar o *endpoint* de criação de um novo usuário.



Fonte: Elaboração própria.

Figura 18 – Resposta HTTP indicando que o usuário foi criado com sucesso.



Fonte: Elaboração própria.

5.4 Feedback e correção de bugs

Após a realização dos testes unitários, informações sobre os bugs encontrados pela equipe, são registradas em um relatório de testes. Essas informações são cruciais para a equipe entender o que está errado e priorizar correções.

Além de problemas técnicos, o relatório fornece um feedback sobre a usabilidade e dificuldades encontradas na interação com o software. A equipe analisa o relatório de bugs para determinar a causa dos problemas. Correções são feitas e o software é testado novamente

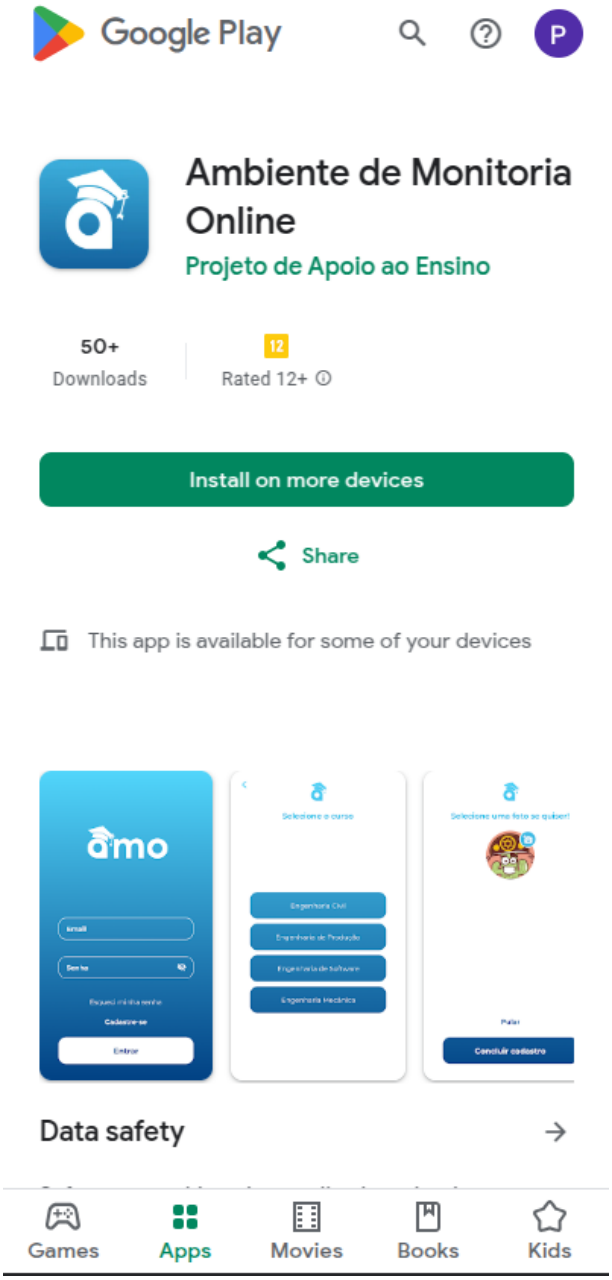
para garantir que os problemas foram resolvidos e que as correções não introduziram novos bugs. Foram utilizadas as ferramentas Visual Studio Code — para edição do código-fonte — e o Insomnia — para a realização de testes através dos *endpoints* da API.

5.5 Lançamento da aplicação

Após o desenvolvimento das funcionalidades, a realização dos testes e a correção dos bugs, a aplicação foi oficialmente lançada em 5 de fevereiro de 2025 no campus de Russas da UFC e divulgada no site do campus (UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS RUSSAS, 2025). O aplicativo *mobile* está hospedo no Google PlayStore (Figura 19) e pode ser baixado por todos usuários [1].

[1] Disponível em: https://play.google.com/store/apps/details?id=com.pae.ambientedemonitoriaonline. Acesso em: 26 fev. 2025.

Figura 19 – Aplicativo AMO no Google PlayStore.



Fonte: Elaboração própria.

6 RESULTADOS

Este capítulo apresenta os resultados do desenvolvimento realizado no projeto até o lançamento da aplicação. Depois que os três módulos principais foram implementados e testados para uso, a aplicação *mobile* foi lançada em 05 de fevereiro de 2025. A API da aplicativo foi hospedada on-line[1], onde pode ser acessada. Os módulos da aplicação podem ser visualizados na plataforma GitHub, onde o código-fonte está hospedado online. Os links para os módulos são: o módulo de cadastro de usuários[2], o módulo do fórum[3] e o módulo de agendamentos[4].

O objetivo geral estabelecido na seção 2.1 foi alcançado, com o desenvolvimento de um *back-end* responsável por fornecer as funcionalidades essenciais para o funcionamento do sistema AMO, como apresentado no capítulo 5

Os objetivos específicos 1 e 2 apresentados na seção 2.2 foram contemplados e apresentados nas seções 5.1, 5.2, 5.3 e 5.4, onde está explícito o desenvolvimento da API REST para aplicação *mobile* e implementação das funcionalidades essenciais do AMO. O alcance do objetivo específico 3 pode ser visto no capítulo 5, onde mostra o processo de documentação das atividades realizadas durante o desenvolvimento.

A API REST foi desenvolvida com sucesso, permitindo a comunicação eficiente entre a API e o aplicativo *mobile*. Além disso, todas as funcionalidades necessárias para a aplicação AMO foram implementadas conforme planejado. Por fim, o processo de desenvolvimento do *back-end* no PAE foi cuidadosamente documentado, detalhando cada etapa, desde a concepção da arquitetura da API até a implementação e testes. Esse registro técnico não apenas reforça a transparência e a rastreabilidade do trabalho realizado, mas também serve como base para futuras melhorias e manutenções do sistema.

---

[1] A API pode ser acessada por meio deste link: https://amo-backend-aa73.onrender.com/Acesso em: 11 de fevereiro de 2025.

[2] Disponível em: https://github.com/PAE-UFC-Russas/amo-backend/tree/3fe350042a7d14c16bdd3c53783e932b85fd6fdc/accounts. Acesso em: 11 fev. 2025.

[3] Disponível em: https://github.com/PAE-UFC-Russas/amo-backend/tree/3fe350042a7d14c16bdd3c53783e932b85fd6fdc/forum_amo. Acesso em: 11 fev. 2025.

[4] Disponível em: https://github.com/PAE-UFC-Russas/amo-backend/tree/3fe350042a7d14c16bdd3c53783e932b85fd6fdc/core. Acesso em: 11 fev. 2025.

7 CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento do sistema AMO atingiu plenamente os objetivos propostos, garantindo a implementação de uma API REST funcional, capaz de viabilizar a comunicação eficiente entre a API e o aplicativo *mobile*. Além disso, todas as funcionalidades essenciais para o correto funcionamento da aplicação foram implementadas, assegurando estabilidade, segurança e desempenho adequado para os usuários. O processo de desenvolvimento do *back-end* foi documentado, proporcionando um registro detalhado das decisões técnicas e arquitetura adotada.

Embora o sistema tenha sido concluído com êxito, abre-se espaço para estudos futuros voltados à avaliação da aceitação, qualidade e usabilidade da aplicação. Além disso, a aplicação de métricas objetivas de qualidade de software, como eficiência, escalabilidade e manutenibilidade, pode fornecer uma visão mais aprofundada sobre o impacto do sistema e sua viabilidade a longo prazo.

Por fim, futuras pesquisas podem explorar a evolução da API para suportar novas funcionalidades, como integração com serviços externos, maior automação de processos e a adoção de arquiteturas mais avançadas, como microsserviços, para garantir maior flexibilidade e escalabilidade. Dessa forma, o sistema AMO poderá continuar evoluindo e atendendo às necessidades dos usuários de modo cada vez mais eficiente.

REFERÊNCIAS

APIDOG. **What is the Difference between Django and Django REST Framework?** s.d. Disponível em: https://apidog.com/articles/difference-between-django-and-django-rest-framework/. Acesso em: 10 set. 2024.

AVARI. **Python Sintaxe: Aprenda a Programar de Forma Simples e Eficiente**. 2023. Disponível em: https://awari.com.br/python-sintaxe-aprenda-a-programar-de-forma-simples-e-eficiente/. Acesso em: 10 set. 2024.

AWS. **O que é uma framework em programação e engenharia?** 2024. Disponível em: https://aws.amazon.com/pt/what-is/framework/. Acesso em: 10 set. 2024.

CORDEIRO, A. C. **Dyfocus: Desenvolvimento do Back-End de um Aplicativo Mobile para Smartphone**. Projeto de Fim de Curso (PFC) – Universidade Federal de Santa Catarina, Florianópolis, SC, February 2014. Apresentado como requisito para a disciplina DAS 5511: Projeto de Fim de Curso.

COSENTINO, V.; IZQUIERDO, J. L. C.; CABOT, J. A systematic mapping study of software development with github. **Journal of Software: Evolution and Process**, v. 29, n. 6, p. e1920, 2017. Disponível em: https://onlinelibrary.wiley.com/doi/10.1002/smr.1920.

DOCS, M. W. **Métodos de requisição HTTP**. 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Methods. Acesso em: 10 set. 2024.

ELLIS, D. R. **Git vs GitHub: What's the Difference?** 2022. Disponível em: https://blog.hubspot.com/website/git-vs-github#:~:text=Git%20is%20a%20version%20control,cannot%20use%20GitHub%20without%20Git. Acesso em: 06 set. 2024.

ERSE, A. V. **Desenvolvimento e análise do backend do projeto CuidaIdoso**. Monografia – Universidade Federal de Ouro Preto, Ouro Preto, MG, August 2021. Apresentada como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

FEDERAL, S. **Decretado fim da emergência sanitária global de Covid-19**. 2023. Disponível em: https://www12.senado.leg.br/radio/1/noticia/2023/05/08/decretado-fim-da-emergencia-sanitaria-global-de-covid-19#:~:text=O%20DIRETOR%2DGERAL%20DA%20OMS,REP%C3%93RTER%20BIANCA%20MINGOTE. Acesso em: 10 set. 2024.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. Design patterns: Elements of reusable object-oriented software. In: **Software Engineering**. [*S. l.*]: Addison-Wesley, 1995.

INSOMNIA. **Get started with Insomnia**. n.d. https://docs.insomnia.rest/insomnia/get-started. Acesso em: 21 jan. 2025.

JONES, C. **The Technical and Social History of Software Engineering**. 1st. ed. Boston: Addison-Wesley Professional, 2013. ISBN 978-0321903426.

MASSÉ, M. **REST API Design Rulebook**. Sebastopol, CA: O'Reilly Media, 2011.

MELO, R. G. de A. Trabalho de Conclusão de Curso, **Desenvolvimento Back-End para um Modelo de Aplicação de Fretes**. Campina Grande, Brasil: [*S. n.*], 2023. Disponível na biblioteca digital da UEPB.

PINHEIRO, R. M. Trabalho de Conclusão de Curso (Graduação), **Desenvolvimento do Back-End de Compra e Venda de Alimentos em Cantinas Universitárias**. Fortaleza: [*S. n.*], 2021.

QUEIRÓS, R.; PORTELA, F. **Desenvolvimento Avançado Para A Web: Do Front-end ao Back-end**. São Paulo: Capa comum, 2020.

SILVERMAN, R. E. **Git Pocket Guide: A Working Introduction**. 1st. ed. Sebastopol, CA: O'Reilly Media, 2013. Capa comum – Ilustrado. ISBN 9781449325862.

SOUZA, H. **A História do Python**. 2023. Disponível em: https://www.dio.me/articles/a-historia-do-python-JCO7UB. Acesso em: 27 jun. 2024.

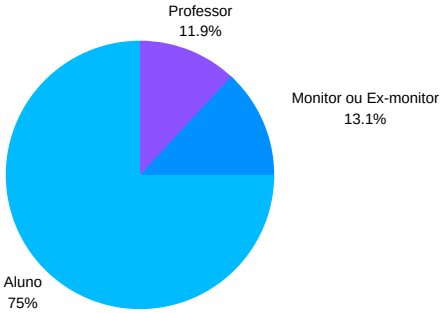
UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS RUSSAS. **Projeto de extensão PAE do campus de Russas lança aplicativo de monitoria online para alunos**. 2025. Disponível em: http://www.campusrussas.ufc.br/noticia.php?v=2275&t=Projeto-de-extens%C3%A3o-PAE-do-campus-de-Russas-lan%C3%A7a-aplicativo-de-monitoria-online-para-alunos. Acesso em: 22 fev. 2025.

VIT, I. C. S. **An Insight Into Visual Studio Code**. 2023. Disponível em: https://medium.com/@IEEE_Computer_Society_VIT/an-insight-into-visual-studio-code-b9e385c8deb7. Acesso em: 10 set. 2024.

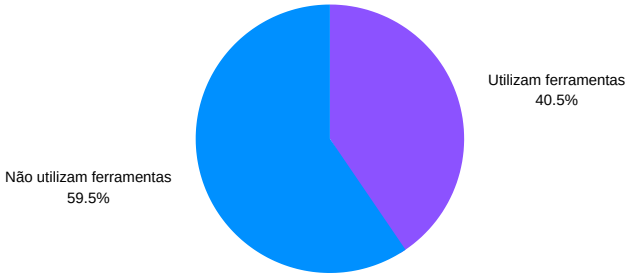
ANEXO A – RESUMO DAS RESPOSTAS OBTIDAS NO QUESTIONÁRIO APLICADO PELO PAE

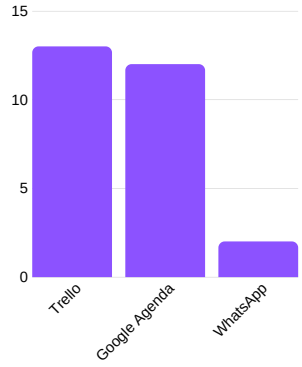
Dados Gerais

Categorias de usuários

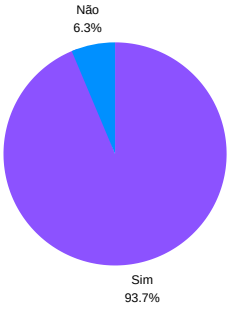


Ferramentas usadas pelos usuários



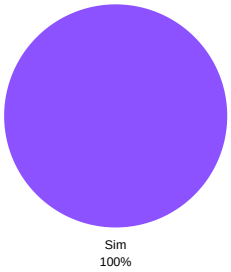


Sobre a utilização de um sistema

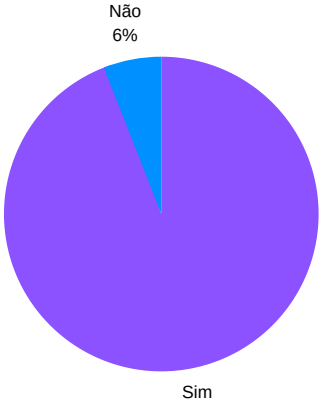


Alunos

Monitores



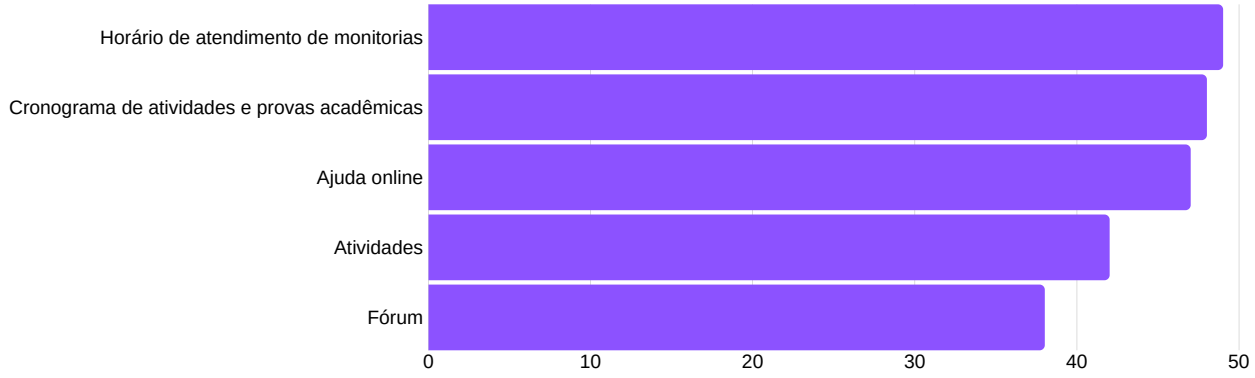
Professores



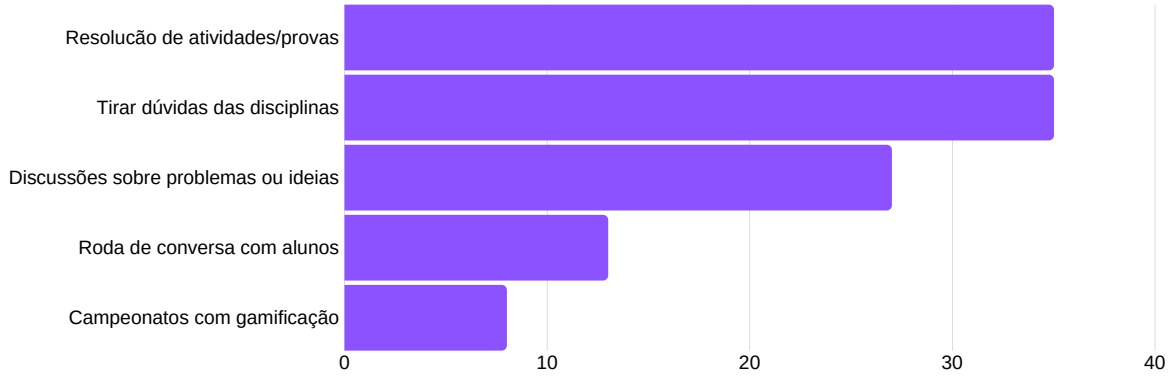
Geral

Alunos

Principais funcionalidades



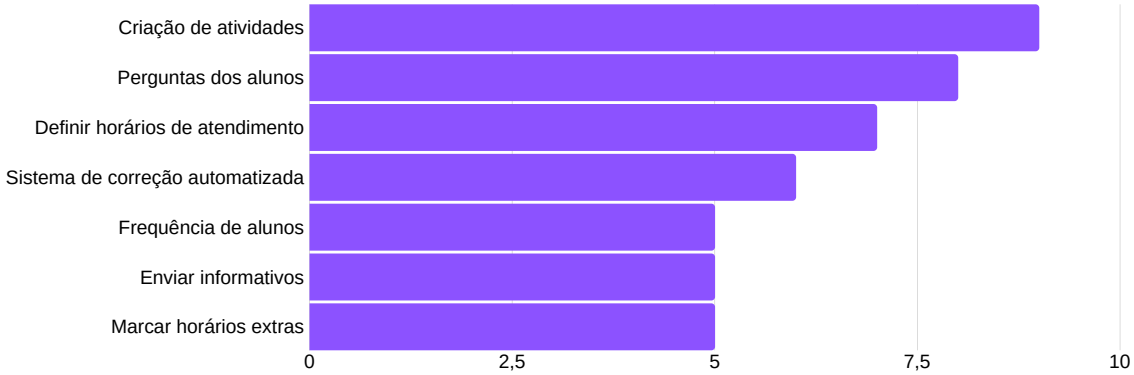
Atividades mais importantes na monitoria



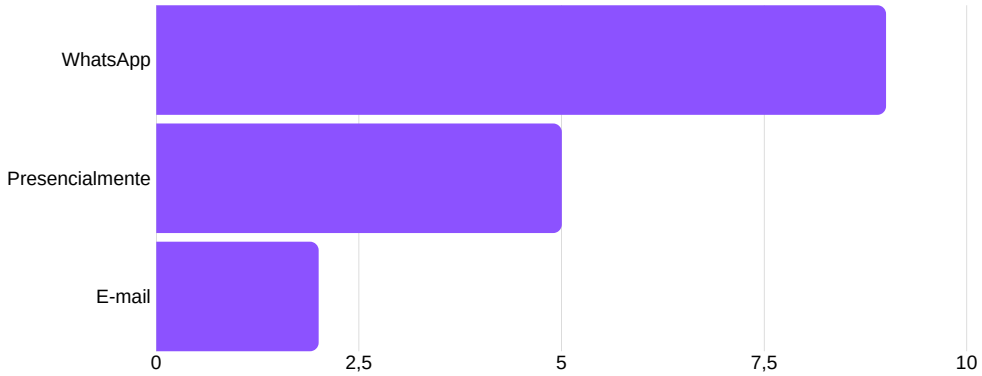
Palavras chaves

Monitores

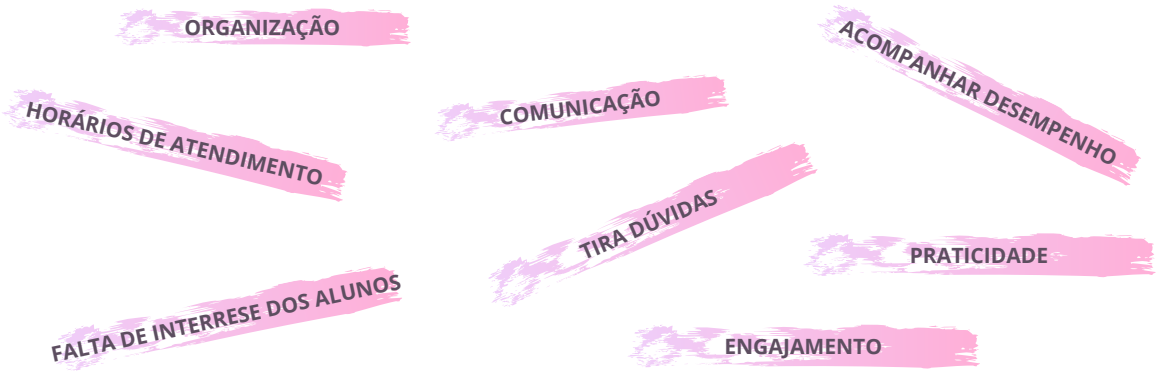
Principais funcionalidades



Comunicação

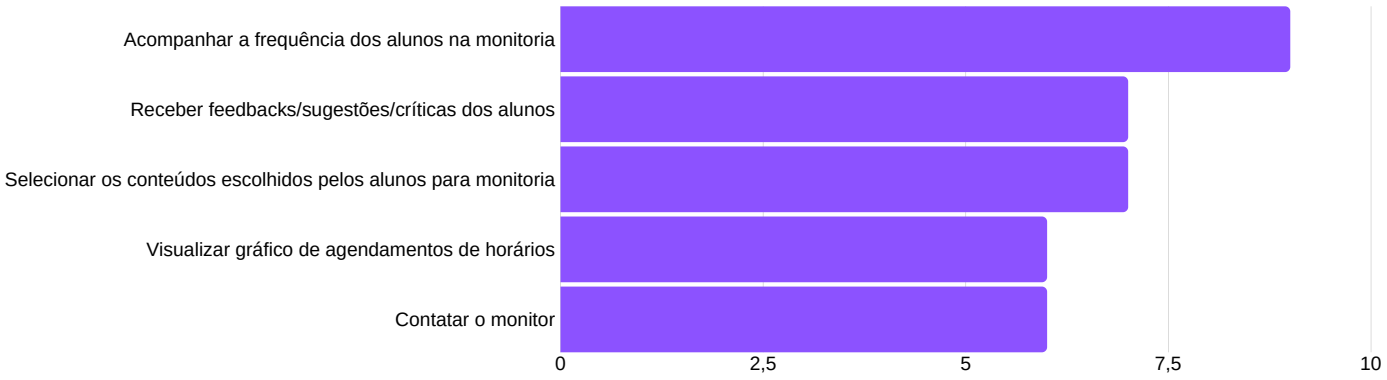


Palavras chaves

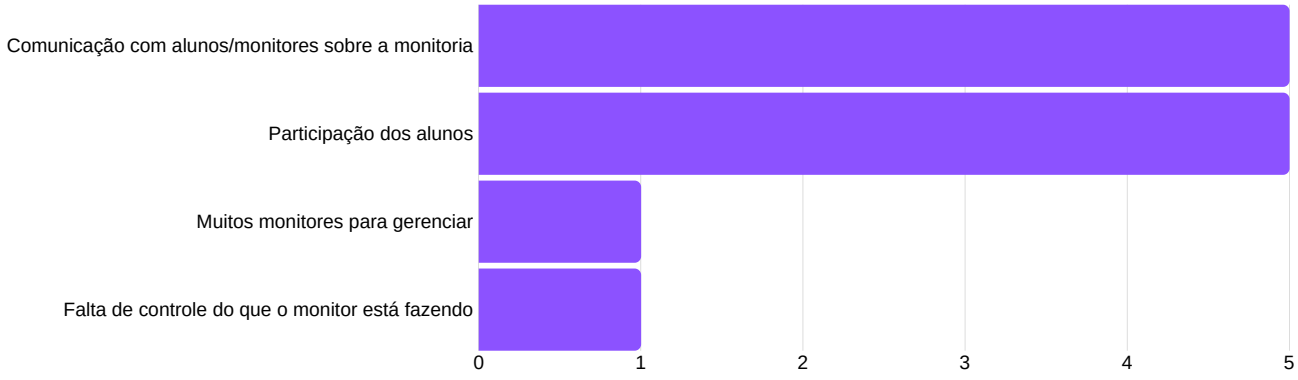


Professores

Principais funcionalidades



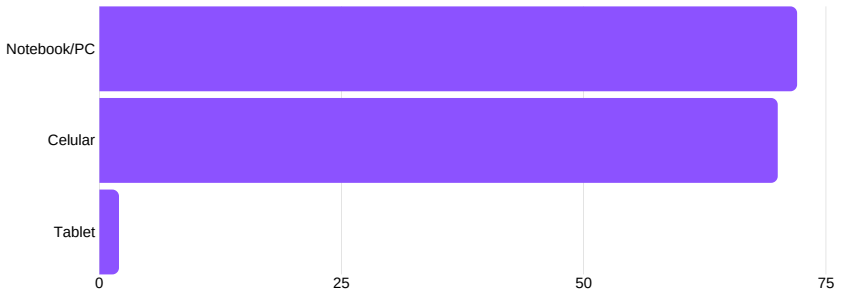
Dificuldades



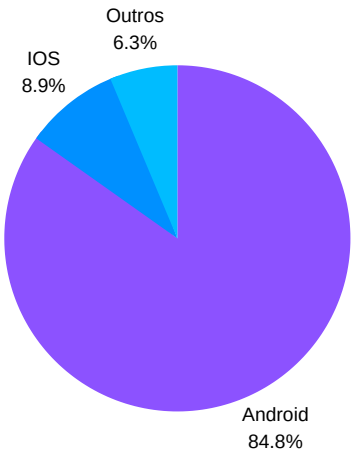
Palavras chaves

Plataforma

Dispositivos



S.O. do celular



Navegadores