**UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI**

**Faculdade de Ciências Exatas - Curso de Sistemas de Informação**

**Samuel José Rodrigues de Araújo Castro**

# Desenvolvimento da Smart Course Library, um repositório de objetos de aprendizagem baseado no modelo IEEE/LOM e ferramenta para criação de cursos personalizados

**Diamantina**

**2024**

**MINISTÉRIO DA EDUCAÇÃO**
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

# FOLHA DE APROVAÇÃO

**Samuel José Rodrigues de Araújo Castro**

## DESENVOLVIMENTO DA SMART COURSE LIBRARY, UM REPOSITÓRIO DE OBJETOS DE APRENDIZAGEM BASEADO NO MODELO IEEE/LOM E FERRAMENTA PARA CRIAÇÃO DE CURSOS PERSONALIZADOS

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisitos parcial para conclusão do curso.

Orientadora: Prof. Dr. Alessandro Vivas Andrade

Aprovada em 12 de abril de 2024

BANCA EXAMINADORA

Prof. Dr. Alessandro Vivas Andrade
Faculdade de Ciências Exatas - DECOM - UFVJM

Profª. Drª. Claudia Beatriz Berti
Faculdade de Ciências Exatas - DECOM - UFVJM

Profª. Drª. Luciana Pereira de Assis
Faculdade de Ciências Exatas - DECOM - UFVJM

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor (a)**, em 07/06/2024, às 08:04, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Claudia Beatriz Berti**, **Servidor (a)**, em 07/06/2024, às 10:39, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Luciana Pereira de Assis**, **Servidor (a)**, em 07/06/2024, às 21:19, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1440822** e o código CRC **25B8E890**.

**Referência:** Processo nº 23086.006867/2024-57 SEI nº 1440822

## RESUMO

Este trabalho apresenta o desenvolvimento da Smart Course Library, um repositório de objetos de aprendizagem baseado no modelo IEEE/LOM e ferramenta para criação de cursos personalizados, empregando linguagens e frameworks modernos como Python, Django, TypeScript e Vue.js. O sistema desenvolvido não apenas facilita a busca e organização de objetos de aprendizagem, mas também integra funcionalidades de CRUD, permitindo a gestão eficiente desses objetos. Além disso, o sistema proporciona a criação de cursos personalizados, promovendo uma experiência de aprendizagem adaptativa e engajadora. A implementação deste repositório visa contribuir para o campo da tecnologia educacional, melhorando a acessibilidade, organização e personalização dos recursos educacionais, e atendendo às necessidades variadas de educadores e alunos.



**Palavras-chave: Smart Course Library, Objetos de aprendizagem, IEEE/LOM, Tecnologia Educacional, Python, Django, TypeScript, JavaScript, Vue.js, CRUD, Experiência de aprendizagem adaptativa, Plataforma de busca**

## ABSTRACT

This work presents the development of the Smart Course Library, a repository of learning objects based on the IEEE/LOM model and a tool for creating personalized courses, using modern languages and frameworks such as Python, Django, TypeScript and Vue.js. The developed system not only facilitates the search and organization of learning objects, but also integrates CRUD functionalities, allowing the efficient management of these objects. Furthermore, the system provides the creation of personalized courses, promoting an adaptive and engaging learning experience. The implementation of this repository aims to contribute to the field of educational technology by improving the accessibility, organization and customization of educational resources, and meeting the varied needs of educators and students.



**Keywords: Learning Objects, IEEE/LOM, Educational Technology, Python, Django, TypeScript, JavaScript, Vue.js, CRUD, Adaptive Learning Experience, Search Platform**

## LISTA DE ILUSTRAÇÕES

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| API | Application Programming Interface |
| CRUD | Create, Read, Update, Delete |
| CSS | Cascading Style Sheets |
| DOM | Document Object Model |
| ECDSA | Elliptic Curve Digital Signature Algorithm |
| ECMA | European Computer Manufacturers Association |
| HMAC | Keyed-Hash Message Authentication Code |
| HTML | HyperText Markup Language |
| HTTP | HyperText Transfer Protocol |
| IEEE | Institute of Electrical and Electronics Engineers |
| JS | JavaScript |
| JSON | JavaScript Object Notation |
| JWT | JSON Web Token |
| LOM | Learning Object Metadata |
| MVC | Model View Controller |
| POO | Programação Orientada a Objetos |
| REST | Representational State Transfer |
| RFC | Request for Comments |
| RSA | Rivest-Shamir-Adleman |
| SEO | Search Engine Optimization |
| SFC | Single File Component |
| *SQL* | Structured Query Language |
| TCC | Trabalho de Conclusão de Curso |
| XML | eXtensible Markup Language |

**SUMÁRIO**

# 1. INTRODUÇÃO

À medida que a tecnologia educacional avança, cresce a demanda por soluções que permitam a educadores e alunos localizar, utilizar e gerenciar objetos de aprendizagem de forma eficaz. O conceito de objetos de aprendizagem, definido pelo padrão IEEE/LOM (Learning Object Metadata), oferece uma estrutura para organizar e recuperar conteúdos educacionais, possibilitando experiências de aprendizagem mais ricas e personalizadas. No entanto, apesar de seu potencial, muitos sistemas educacionais ainda não conseguem explorar plenamente as capacidades oferecidas por esses padrões, deixando uma lacuna significativa no mercado.

Em primeiro lugar, o IEEE Learning Object Metadata (LOM) é um padrão internacionalmente reconhecido para a descrição de "objetos de aprendizagem", publicado pela IEEE Standards Association. Os objetos de aprendizagem são definidos como "qualquer entidade, digital ou não digital, que pode ser usada para aprendizagem, educação ou treinamento" (BARKER, 2005). O IEEE 1484.12.1 é a primeira parte de um padrão multipartido e descreve o modelo de dados do LOM, especificando quais aspectos de um objeto de aprendizagem devem ser descritos e quais vocabulários podem ser usados para essas descrições. Ele também define como esse modelo de dados pode ser alterado por adições ou restrições.

Sendo assim, o LOM é projetado para ajudar na criação de descrições bem estruturadas de recursos de aprendizagem, facilitando sua descoberta, localização, avaliação e aquisição por estudantes, professores ou processos automatizados. Além disso, permite a troca de descrições de recursos entre sistemas de descoberta de recursos, a redução dos custos de serviços baseados em descrições de recursos de alta qualidade, e a adaptação das descrições para atender às necessidades especializadas de uma comunidade. O LOM pode ser usado juntamente com outras especificações para "marcar" recursos de aprendizagem com uma descrição que pode ser associada ao recurso, fornecendo informações em um formato padrão semelhante ao encontrado na capa de um livro (BARKER, 2005).

Neste contexto, o presente trabalho propõe o desenvolvimento da "Smart Course Library", um repositório de objetos de aprendizagem baseado no modelo IEEE/LOM, que também serve como uma ferramenta para a criação de cursos personalizados. Para isso, foram utilizadas linguagens e frameworks modernos como Python, Django, TypeScript e Vue.js, criando uma plataforma educacional que integra funcionalidades de busca avançada e gestão de objetos de aprendizagem.

O sistema proposto não apenas facilita a busca e organização de objetos de aprendizagem, mas também oferece funcionalidades de CRUD (Create, Read, Update, Delete) para gerenciar esses objetos, atendendo às necessidades de diferentes tipos de usuários. Além disso, permite a criação de cursos personalizados a partir dos objetos de aprendizagem selecionados, promovendo uma experiência educacional mais adaptativa.

O diferencial da Smart Course Library reside na integração de funcionalidades avançadas que atendem às necessidades específicas de educadores e alunos. Ao permitir a criação de cursos personalizados, a plataforma promove uma experiência de aprendizagem adaptativa e engajadora, ajustando-se às preferências e estilos de aprendizagem individuais. Estudos indicam que a personalização do ensino pode aumentar significativamente a motivação e a eficácia do aprendizado, adaptando os recursos educacionais aos estilos e capacidades dos alunos (SILVA, 2011).

Logo, a implementação deste projeto visa contribuir para o campo da tecnologia educacional, oferecendo uma ferramenta que melhora a acessibilidade, organização e personalização dos recursos educacionais. Além da possibilidade de impactar positivamente professores, estudantes e instituições educacionais, ao proporcionar uma alternativa eficiente para o gerenciamento de conteúdo educacional e a criação de cursos adaptativos.

## 2. JUSTIFICATIVA

A justificativa deste projeto, parte da problemática que a evolução tecnológica na educação traz. Há uma necessidade crescente de soluções que permitam experiências de ensino mais personalizadas e adaptativas. Sendo assim, a personalização do ensino é fundamental para atender às necessidades individuais dos alunos, melhorando seu engajamento e desempenho acadêmico. Estudos indicam que a personalização do aprendizado pode aumentar significativamente a motivação e a eficácia do ensino, adaptando os recursos educacionais aos estilos de aprendizagem e capacidades individuais dos estudantes (NAZEMPOUR; DARABI, 2023; WEI, Xin et al., 2018)..

A importância dos objetos de aprendizagem (OAs) no contexto educacional é vastamente reconhecida. Esses recursos didáticos digitais são fundamentais para promover a autonomia dos estudantes, acessibilidade, cooperação e interatividade no processo de ensino-aprendizagem. Os OAs permitem que educadores integrem diferentes sistemas e módulos de ensino, enriquecendo a experiência educacional e garantindo que os conteúdos sejam apresentados de maneira dinâmica e engajante. Deste modo, os objetivos de aprendizagem são projetados para serem reutilizáveis e adaptáveis a diversos contextos educativos. A reutilização e a adaptabilidade dos mesmos possibilitam que eles sejam utilizados em diferentes disciplinas e formatos, aumentando sua eficácia e eficiência na disseminação do conhecimento (ALEXANDRE; BARROS, 2020).

Ademais, o desenvolvimento de um repositório de objetos de aprendizagem é crucial para atender às demandas educacionais contemporâneas, além disso a implementação desse tipo de sistema oferece diversas vantagens, como a otimização do processo de ensino, maior engajamento dos alunos e a possibilidade de acompanhamento contínuo do progresso dos cursos criados. A análise dos comportamentos dos estudantes em ambientes virtuais permite ajustar os recursos educacionais de forma a maximizar o desempenho acadêmico (VIEIRA et al., 2005).

Portanto, o desenvolvimento deste projeto tem como propósito não apenas melhorar a acessibilidade e a organização dos recursos educacionais, mas também enriquecer o processo de ensino-aprendizagem. Sua relevância se destaca pelo potencial de colaborar positivamente professores, estudantes e instituições educacionais, oferecendo uma alternativa robusta e eficiente para o gerenciamento de conteúdos educacionais e a criação de cursos adaptativos.

# 3. OBJETIVOS

Neste capítulo, serão apresentados os objetivos que guiaram o desenvolvimento da Smart Course Library. Serão detalhadas as metas principais e específicas do projeto.

## 3.1 GERAL

O objetivo geral deste trabalho é desenvolver um repositório de objetos de aprendizagem baseado no modelo IEEE/LOM que também seja uma ferramenta de criação de cursos personalizados. Este projeto, denominado Smart Course Library, visa facilitar a busca, organização e gestão de objetos de aprendizagem, além de proporcionar a criação de cursos personalizados. A aplicação será desenvolvida utilizando linguagens e frameworks modernos como Python, Django, TypeScript e Vue.js.

## 3.2 ESPECÍFICOS

Os objetivos específicos deste trabalho incluem facilitar a busca e a organização de objetos de aprendizagem por meio de uma plataforma de busca, que permita aos usuários localizar esses objetos de maneira eficiente. Além disso, visa fornecer funcionalidades de CRUD (Create, Read, Update, Delete) para a gestão dos objetos de aprendizagem, atendendo às necessidades de diferentes tipos de usuários. Outro objetivo é permitir a criação de cursos personalizados, implementando uma ferramenta que possibilite aos usuários criar cursos personalizados a partir dos objetos de aprendizagem disponíveis, promovendo uma experiência de aprendizagem adaptativa e engajadora.

Adicionalmente, o trabalho busca garantir a eficiência e a escalabilidade do sistema, utilizando linguagens e frameworks modernos para assegurar que o sistema seja eficiente, escalável e fácil de manter. Por fim, pretende contribuir para o campo da tecnologia educacional, melhorando a acessibilidade, organização e personalização dos recursos educacionais, atendendo às variadas necessidades de educadores e alunos, e contribuindo para o avanço da tecnologia educacional.

# 4. METODOLOGIA

O presente trabalho tem como foco o desenvolvimento de uma solução prática para um problema identificado no campo da tecnologia educacional. O estudo envolve a criação e implementação de um sistema de software, caracterizando-se, portanto, como uma pesquisa de desenvolvimento tecnológico.

A primeira etapa envolveu o levantamento de requisitos funcionais e não funcionais do sistema. Para isso, foram realizadas reuniões com o meu orientador, Prof. Dr. Alessandro Vivas Andrade, visando identificar as necessidades e expectativas em relação ao repositório e à ferramenta de criação de cursos personalizados. O problema que justificou a criação deste repositório é a falta de soluções no mercado que integrem de forma eficiente funcionalidades de busca, organização e personalização de cursos, resultando em uma lacuna significativa na maneira como objetos de aprendizagem são gerenciados e utilizados.

O desenvolvimento do sistema utilizou linguagens e frameworks modernos. No backend, foram escolhidos Python e Django devido à sua robustez, segurança e facilidade de integração com outras tecnologias. Para o frontend, foram utilizados TypeScript e Vue.js, escolhidos por sua flexibilidade e capacidade de criar interfaces de usuário dinâmicas e responsivas. O banco de dados utilizado foi o PostgreSQL, selecionado por sua escalabilidade e suporte a operações complexas.

A implementação foi realizada em etapas distintas. Dito isso, a primeira etapa foi a configuração do ambiente de desenvolvimento, que envolveu a instalação e configuração das ferramentas e dependências necessárias. Em seguida, foi desenvolvido o backend com a implementação das APIs RESTful, utilizando Django Rest Framework (DRF) e a criação dos modelos de dados baseados no padrão IEEE/LOM. A terceira etapa foi o desenvolvimento do frontend, que incluiu a criação da interface do usuário utilizando Vue.js e a implementação das funcionalidades de busca, visualização, criação e gerenciamento de objetos de aprendizagem. Por fim, foi realizada a integração das partes frontend e backend, garantindo a comunicação e funcionamento harmonioso entre ambas.

## 4.1 TRABALHOS RELACIONADOS

Foram realizadas buscas para avaliar os repositórios de objetos de aprendizagem já existentes, a fim de identificar soluções similares e entender suas funcionalidades e limitações. Dentre os repositórios analisados, destacam-se o Banco Internacional de Objetos

Educacionais (BIOE), o MERLOT (Multimedia Educational Resource for Learning and Online Teaching), o Eduteka e o OER Commons.

1. O Laboratório Didático Virtual (LABVIRT), da USP, oferece acesso a objetos de aprendizagem de química que envolvem situações cotidianas. Ele disponibiliza uma série de experimentos virtuais e materiais de apoio para o ensino da química.
2. O Laboratório Virtual de Matemática da UNIJUÍ disponibiliza objetos de aprendizagem na área de matemática, organizados por assunto. Este repositório inclui software educacional desenvolvido para os níveis fundamental, médio e superior, com recursos que variam de jogos educativos a aplicações interativas.
3. O PHET, desenvolvido pela University of Colorado Boulder, apresenta um conjunto de simuladores interativos organizados por áreas como matemática, física, química e ciências da terra. Esses simuladores são projetados para melhorar a compreensão dos conceitos científicos por meio de experimentos virtuais.
4. O Repositório de Objetos de Aprendizagem Univates (ROAU) oferece diversos materiais educativos que cobrem uma ampla gama de tópicos, com ênfase em fornecer recursos acessíveis e de alta qualidade para diferentes áreas do conhecimento.
5. O BBC Learning English disponibiliza conteúdos para o estudo da Língua Inglesa, oferecendo uma variedade de materiais, incluindo vídeos, áudios e textos, que visam aprimorar as habilidades de comunicação em inglês dos usuários.
6. O MERLOT, disponibilizado pela California State University, apresenta coleções de materiais didático-pedagógicos em diversas áreas do conhecimento e múltiplos idiomas, incluindo materiais em língua portuguesa. Este repositório é amplamente utilizado por educadores para compartilhar e avaliar recursos educacionais.

Por fim, o MIT OpenCourseWare, do Massachusetts Institute of Technology, é um repositório gratuito que disponibiliza materiais em vídeo, textos e aulas organizadas por área de conhecimento, com alguns materiais traduzidos para diversos idiomas. Este repositório é reconhecido pela qualidade e abrangência dos seus conteúdos educacionais.

A análise desses repositórios permitiu identificar características essenciais e boas práticas que foram consideradas no desenvolvimento do sistema proposto. A integração da

funcionalidade de personalização de cursos foi um diferencial observado, justificando a necessidade da criação de uma nova solução que preencha essa lacuna.

# 5. LINGUAGENS UTILIZADAS E SEUS FRAMEWORKS

Este capítulo visa descrever as linguagens de programação e seus respectivos frameworks utilizados no desenvolvimento deste projeto. A seleção das tecnologias é crucial para o desempenho, escalabilidade e manutenção da aplicação. Serão discutidas as principais linguagens empregadas, destacando suas características, vantagens e desvantagens, bem como os frameworks que complementam essas tecnologias, facilitando e agilizando o processo de desenvolvimento.

## 5.1 Tecnologias utilizadas para o Backend

O backend é uma parte fundamental de qualquer aplicação, sendo responsável pela lógica de negócios, manipulação de dados e integração com outras partes do sistema. Este tópico abordará as tecnologias selecionadas para o backend, começando pela linguagem de programação principal e seus frameworks associados, justificando as escolhas feitas com base em suas características e benefícios.

### *5.1.1 Linguagem de programação Python*

Python é uma linguagem de programação que foi criada no final dos anos 80, por Guido van Rossum, na época, um pesquisador do Centrum Wiskunde & Informatica (Centro de Matemática e Informática) na Holanda. Foi criado a partir de uma linguagem da época, chamada ABC, que tinha como foco inicial o uso para físicos e engenheiros. Desde então, ela tem se tornado cada vez mais popular e é atualmente uma das linguagens de programação mais utilizadas no mundo. De acordo com uma pesquisa realizada sobre o ecossistema dos desenvolvedores, a linguagem é a segunda mais utilizada, onde 55% dos desenvolvedores utilizam-na e 7% pretendem adotá-la; ficando atrás somente do Javascript com 65% de utilização (Jetbrains, 2022).

Por ser uma linguagem de código aberto, interpretada e sua sintaxe facilita a leitura e a escrita de código. Yuill e Halpin (2006) descrevem a linguagem de programação sendo poderosa, elegante, fácil de ler e entender, além de possuir uma comunidade de programadores ampla ao redor do mundo. Segundo Borges (2014), o Python é uma linguagem produtiva, devido ao fato de ter uma sintaxe clara e de fácil leitura, e ser composta por várias estruturas de alto nível junto com bibliotecas e frameworks de terceiros que podem ser

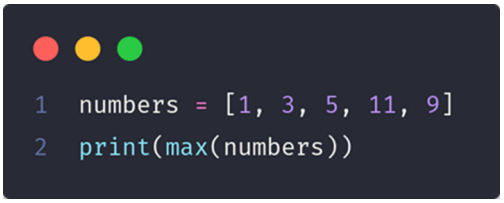
facilmente integrados aos projetos. Sobre seu paradigma e o modo de interpretação, pode-se afirmar que:

> Multiparadigma, a linguagem suporta programação modular e funcional, além da orientação a objetos. Mesmo os tipos básicos no Python são objetos. A linguagem é interpretada através de bytecode pela máquina virtual Python, tornando o código portável. Com isso é possível compilar aplicações em uma plataforma e rodar em outros sistemas ou executar direto do código-fonte (Borges, 2014, p. 14).

Como classificam Downey, Elkner e Meyers (2012, p. 1), Python é uma linguagem de alto nível, ou seja, é mais próxima da linguagem humana do que da linguagem de máquina. Contudo, eles inferem que a linguagem seja do mesmo nível que C, essa última, surgiu no início da década de 70 e foi usada inicialmente no desenvolvimento de sistemas operacionais. Com toda certeza C se difere da linguagem de máquina, mas pode-se observar os seguintes exemplos de códigos que retornam o maior valor de um conjunto de números:

Código em Python:

**Figura 1 - Obter número máximo em Python**

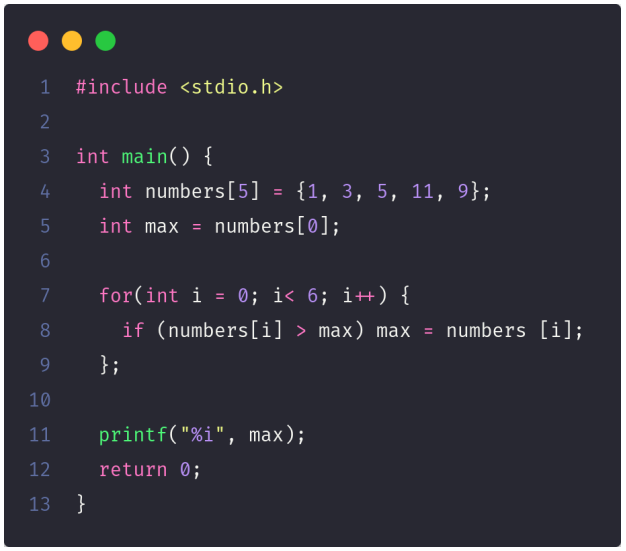


```
1  numbers = [1, 3, 5, 11, 9]
2  print(max(numbers))
```

Fonte: Elaborado pelo autor (2024).

Código em C:

**Figura 2 - Obter número máximo em C**

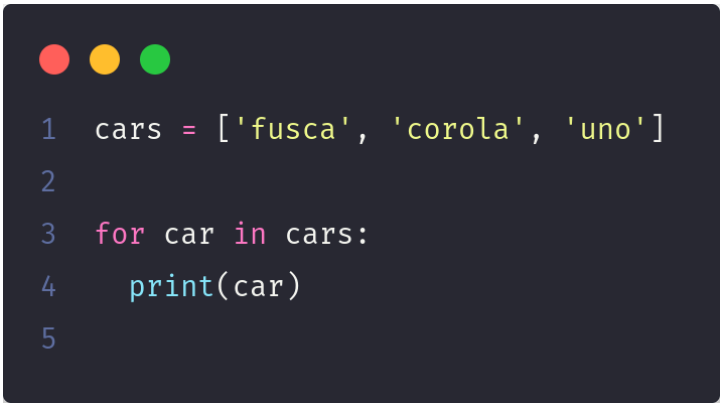


```
1  #include <stdio.h>
2
3  int main() {
4    int numbers[5] = {1, 3, 5, 11, 9};
5    int max = numbers[0];
6
7    for(int i = 0; i< 6; i++) {
8      if (numbers[i] > max) max = numbers [i];
9    };
10
11   printf("%i", max);
12   return 0;
13 }
```

Fonte: Elaborado pelo autor (2024).

É nítido que o trecho de código escrito em Python, na figura 1, é mais simples e fácil de entender, visto que foi utilizada uma função nativa para achar o maior valor na lista e o valor é mostrado no console. Já o trecho escrito em C, na figura 2, atribuiu uma variável para armazenar o valor máximo, e utilizando um loop para percorrer o array, comparando se o valor atual é maior que o anterior e se for atribui-se o valor maior à variável que armazena o valor máximo. Observou-se, a seguir, nas figuras 3 e 4, estruturas de repetição de ambas as linguagens.

**Figura 3 - Estrutura de repetição em Python**




Fonte: Elaborado pelo autor (2024).

**Figura 4 - Estrutura de repetição em C**


Fonte: Elaborado pelo autor (2024).

Visualizou-se que na figura 3, que está escrito em Python, a estrutura é possível ser lida de forma mais natural ou “humana”. Conseguiu-se ler “Para carro em carros” (*for car in*

*cars*), é explícito que “carro” é um valor de “carros”. Agora na figura 4, que está escrito em C, há um loop mais robusto, onde são passados três parâmetros. O primeiro é a variável que será iterada; o segundo, uma condição de funcionamento do loop, que enquanto for verdadeira o loop será executado; e por último, passa-se um incremento para a variável de iteração. Com esses exemplos, é correto afirmar que Python se aproxima mais da linguagem humana, ao ser comparada ao C.

Contudo, é importante ressaltar que apesar do Python possuir uma sintaxe mais clara e recursos nativos que aumentam a produtividade durante o desenvolvimento, ela é uma linguagem mais lenta, devido ao fato de rodar em uma máquina virtual e ser de mais alto nível. Com C ganha-se performance, porém é necessário preocupar-se com funcionalidades que já vêm prontas em outras linguagens.

A linguagem é multiparadigma, contudo, foi realizada uma explicação para o paradigma de orientação a objetos. De maneira resumida, a orientação a objetos pode ser utilizada para abstrair coisas do mundo real, como pessoas, ou até mesmo padrões ou protocolos. No exemplo a seguir, foi produzida a abstração de uma pessoa utilizando uma classe:

**Figura 5 - Classe em Python**

```python
class People:
  def __init__(self, name, age):
    self.name = name
    self.age = age

  def walk():
    print('wallking')

  def talk():
    print('talking')
```

Fonte: Elaborado pelo autor (2024),

Como mostrado na figura 5, a classe “People” representou uma pessoa e contém os atributos “name” e “age”, que são informações sobre a mesma; também contém os métodos “walk” e “talk” que são de maneira abstraída, possíveis ações a serem tomadas por um

humano. Agora, para representar uma pessoa no sistema, pode ser criada uma instância da classe e assim acessar seus métodos e propriedades, como mostrado na figura 6.

**Figura 6 - Instância da classe “People”**


Fonte: Elaborado pelo autor (2024).

Para justificar o uso do Python neste projeto, foi possível analisar novamente alguns dados da pesquisa “Ecossistema de desenvolvedores” realizada pela Jetbrains em 2022, com foco nos frameworks para desenvolvimento Web e uso da linguagem.

Segundo a pesquisa, os três maiores usos do Python são: empatados em primeiro lugar com 43%, análise de dados e desenvolvimento Web, e em segundo lugar, machine learning com 39%. Neste trabalho, foi realizado o uso do Python para desenvolvimento Web. No entanto, o levantamento também apontou informações sobre os frameworks da linguagem - conjuntos de pacotes que direcionam o desenvolvimento de aplicações. Foi considerada uma queda na tendência do uso do Python para o desenvolvimento Web, devido a uma queda na popularidade do Flask e Django, que caíram em 6%. Contudo, no mesmo período de tempo, a popularidade do FastAPI aumentou em 6%. Enquanto Flask e Django foram de 46% para 40% e 45% para 39%, respectivamente, a FastAPI foi de 14% para 20%. Esse acontecimento pode ser justificado pelo seguinte argumento:

> A FastAPI é uma framework Web moderna construída para alto desempenho e ergonomia do desenvolvedor. Ela usa recursos modernos do Python, como dica de tipos, possui suporte assíncrono integrado, foi projetada para criar APIs com Python e muito mais (Jetbrains, 2022).

Foi possível concluir que a queda de popularidade do Django e Flask aconteceu devido ao fato de serem frameworks Full Stack, enquanto a FastAPI é focada na construção de APIs. Neste projeto, foi utilizado o Django para desenvolver o back-end da aplicação.

Em síntese, Python é uma linguagem de programação versátil e fácil de aprender, que possui uma ampla gama de bibliotecas e ferramentas disponíveis. No entanto, como qualquer tecnologia, ela apresenta suas desvantagens. Uma delas é sua relativa lentidão em comparação com linguagens compiladas, como C ou Java. Outra desvantagem é a possibilidade de erros de tipo em tempo de execução, devido a operações incompatíveis com tipos de dados, uma característica comum em linguagens dinamicamente tipadas como Python. Esses erros ocorrem porque o tipo de uma variável é determinado apenas durante a execução, o que pode levar a bugs que são detectados apenas quando o código é executado. Apesar desses desafios, a popularidade do Python tem crescido consistentemente, não apenas na indústria, mas também no campo educacional, tornando-se uma escolha popular para ensinar ciência da computação.

### *5.1.2 Framework Django*

Após a análise dos dados sobre os frameworks utilizados para o desenvolvimento Web com Python no capítulo anterior, optou-se por utilizar o Django no desenvolvimento do Backend, já que o mesmo é um framework amplamente utilizado e consolidado no mercado. A sua documentação oficial, introduz o Django da seguinte forma:

> Um impulsionador no processo de criação de aplicativos de forma ágil, ao mesmo tempo em que oferece um design refinado e pragmático. Concebido por talentosos desenvolvedores, ele alivia muitas das complexidades envolvidas no desenvolvimento Web, permitindo que você se concentre exclusivamente na criação de seu aplicativo, sem a necessidade de começar do zero. Além disso, o Django é uma solução gratuita e de código aberto, disponível para todos que desejam utilizá-lo (Django Project, 2023).

Sua documentação (2023), também levantou as seguintes vantagens:

1. “Ridiculamente rápido”, foi desenvolvido para os desenvolvedores finalizarem suas aplicações o mais rápido possível.
2. “Totalmente carregado”, o framework contém dezenas de extras para lidar com tarefas comuns no desenvolvimento Web.

3. “Tranquilamente seguro”, o Django leva segurança a sério, e ajuda os desenvolvedores a evitar erros de segurança comuns como: *SQL injection*, *cross-site scripting*, *cross-site request forgery* e *clickjacking.* Além de prover segurança para administrar usuários e suas senhas.
4. “Extremamente escalável”, eles citam que alguns dos sites mais movimentados do planeta usam a capacidade do Django de escalar com rapidez e flexibilidade para atender às demandas de tráfego mais pesadas.
5. “Incrivelmente versátil”, eles afirmam que empresas, organizações e governos usaram o Django para construir todo tipo de coisas, de sistemas de gerenciamento de conteúdo à redes sociais e plataformas de computação científica.

Comumente, viu-se frameworks com a arquitetura MVC (Model View Controller), ou seja, possuem 3 camadas: Modelo, Visão e Controlador.

> Modelo deve lidar com questões relativas à lógica do negócio. A Visão deve se ocupar somente da interface com o usuário. Por fim, o Controlador deve realizar a tarefa intermediária de receber as requisições do usuário através da Visão e distribuir a realização dessa requisição entre os vários componentes da camada de Modelo, enviando o resultado à camada de Visão, e novamente ao usuário (Lopes, 2005, p. 34).

Porém, o Django possui uma pequena variação dessa arquitetura, o MVT (Model View Template), possuindo as camadas: Modelo, Visão e Template. A documentação oficial do framework explicou as camadas da seguinte forma: a camada Modelo é uma camada de abstração, utilizada para estruturar e manipular os dados da sua aplicação com seu banco de dados. A Visão é a responsável por encapsular a lógica responsável por processar uma solicitação de um usuário e devolver uma resposta. A camada Template fornece uma sintaxe amigável ao designer para renderizar as informações a serem apresentadas ao usuário, que utiliza a Linguagem de Template Django. Essa linguagem permite que você faça condicionais, loops e acesse variáveis utilizando uma sintaxe semelhante ao Python em meio ao HTML.

Apesar do Django oferecer recursos robustos de templates, os mesmos não foram utilizados inicialmente nesta aplicação. Contudo, é viável utilizar esses templates futuramente para aprimorar e personalizar o painel de administrador conforme as necessidades do projeto. O painel de administrador é uma ferramenta essencial no Django, destinada aos administradores do sistema para gerenciar usuários e suas permissões de forma eficiente dentro da aplicação.

A documentação oficial do Django destacou que a interface de administrador é eficaz porque lê informações dos modelos para criar uma interface focada nos dados. Isso torna mais fácil para os usuários autorizados gerenciarem o conteúdo do site. Embora poderosa, a recomendação é usar o painel de administrador primordialmente para tarefas de gerenciamento interno, não como uma solução completa para o front-end da aplicação.

O Django tem o papel de comunicar-se com o Frontend, e para alcançar tal objetivo, foi feito o uso de APIs REST. Pode-se definir API como:

> Uma API (Application Programming Interface) é um conjunto de características e regras existentes em uma aplicação que possibilitam interações com essa através de um software - ao contrário de uma interface de usuário humana. A API pode ser entendida como um simples contrato entre a aplicação que a fornece e outros itens, como outros componentes do software, ou software de terceiros (API, 2024).

Segundo Barry (2023), REST (Representational State Transfer) é um estilo de arquitetura baseado em um conjunto de princípios que descrevem como os recursos em rede são definidos e endereçados. Um ponto de possível o ao construir as APIs, e citado pelo autor, é que os recursos são endereçáveis usando um conjunto mínimo e uniforme de comandos, normalmente usando comandos HTTP de GET, POST, PUT ou DELETE pela Internet; ou seja, uma mesma API pode ser acessada utilizando esses diferentes verbos, e cada verbo acessa um recurso específico.

Para construir uma aplicação REST, foi utilizado o Django Rest Framework, descrito em sua documentação como um kit de ferramentas para desenvolver Web APIs. Esse framework facilita a serialização de *Models*, utilizando de *Serializers* ou Serializadores convertendo dados complexos, como querysets e instâncias de modelo, em formatos nativos do Python, como JSON ou XML. Querysets são conjuntos de registros do banco de dados, representando cada linha como uma instância de modelo em Python, seguindo os princípios da Programação Orientada a Objetos (POO).

### *5.1.3 Conclusão sobre as Tecnologias para o Backend*

Combinando o Python com o Django, é possível criar aplicativos Web altamente eficientes e escaláveis, com grande facilidade de desenvolvimento e manutenção. Para a construção do repositório de objetos de aprendizagem, a utilização do Python e do Django pode permitir o desenvolvimento de um sistema Web altamente escalável e seguro, com grande capacidade de processamento de dados e facilidade de integração com outras

ferramentas. Além disso, a linguagem Python possui uma vasta variedade de bibliotecas científicas, que podem ser utilizadas para a análise de dados e o processamento de informações, o que é essencial em um projeto de pesquisa.

## 5.2 Tecnologias utilizadas para o Frontend

Neste projeto, foram escolhidas tecnologias adequadas e eficientes para o desenvolvimento do frontend, buscando oferecer uma experiência de usuário funcional e agradável. Serão apresentadas as principais tecnologias utilizadas, explicando suas escolhas e destacando suas características e benefícios. A combinação de linguagens e frameworks foi selecionada para facilitar o desenvolvimento e garantir uma interface de usuário clara e intuitiva.

### *5.2.1 Linguagem de Programação JavaScript e Superset TypeScript*

O desenvolvimento de aplicações Web tem sido uma área em constante evolução, com linguagens de programação como JavaScript desempenhando um papel crucial nesse cenário. JavaScript foi criado para ser uma linguagem de programação utilizada principalmente para scripts dinâmicos do lado do cliente em páginas Web. Com o tempo, também ganhou aplicabilidade no lado do servidor através de interpretadores como o Node.js (JavaScript, 2024). Ou seja, a linguagem ganhou um ambiente de execução fora do navegador.

A linguagem foi criada em setembro de 1995 para o Netscape Navigator 2.0, sendo introduzido pelo Internet Explorer 3.0 em agosto de 1996. Em novembro do mesmo ano, a Netscape, juntamente com a ECMA International (ECMA é a sigla para European Computer Manufacturers Association) começaram um trabalho para transformar a linguagem de programação um padrão para a Web, cujo o JavaScript padronizado possui o nome de ECMAScript. Sua edição mais recente é a décima primeira versão, o ES2020, lançado em junho de 2020 (JavaScript, 2024).

Diferentemente de linguagens com tipagem estática, o JavaScript, assim como o Python, não exige a declaração explícita de tipos de dados. Isso oferece uma maior liberdade para os desenvolvedores, permitindo a criação de variáveis, parâmetros e tipos de retorno de função sem tipagem explícita (JavaScript, 2024). Porém, foi apresentado a frente, que isso também pode ocasionar problemas indesejados no desenvolvimento de aplicações.

Dito isto, o JavaScript utiliza um modelo de objeto baseado em protótipo, chamado de *prototype chain* ou cadeia de protótipos*,* que difere do modelo baseado em classes encontrado em outras linguagens de programação. Este modelo oferece herança dinâmica, permitindo que

o que é herdado possa variar para objetos individuais, o que torna o JavaScript uma linguagem altamente flexível e adaptável (JavaScript, 2024).

Utilizando o terminal de um navegador, foi demonstrado essa cadeia. Foi declarado um array que contém nomes de esportes, e adicionou-se o esporte “basquete” ao array através do método push, como mostrado na figura 7.

**Figura 7 - Código em JavaScript**

```
> const sports = ["futebol"];

  sports.push("basquete");
```

Fonte: Elaborado pelo autor (2024).

No JavaScript, o array declarado é um tipo primitivo, e tipos primitivos são apenas valores e não possuem métodos ou propriedades. Então foi plausível fazer a seguinte pergunta: como foi possível utilizar o método “push”? Inspecionou-se o array na figura 8.

**Figura 8 - Inspeção do array no terminal do navegador**

```
> sports
  ▼ (2) ['futebol', 'basquete'] i
      0: "futebol"
      1: "basquete"
      length: 2
    ▶ [[Prototype]]: Array(0)
```

Fonte: Elaborado pelo autor (2024).

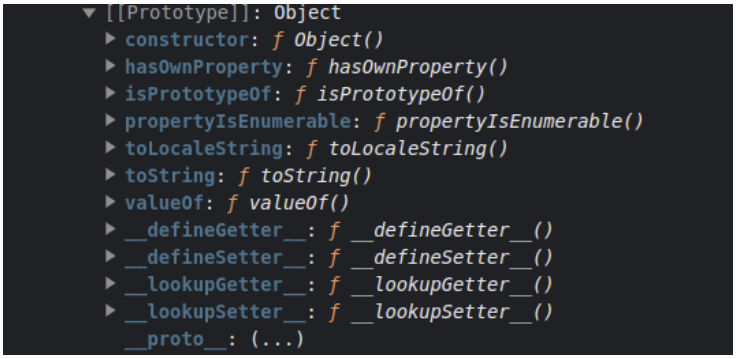
Foi visto que o array possui 2 valores, “futebol” na posição 0 e “basquete” na posição 1. O navegador também mostrou o comprimento do array com a propriedade length, porém, essa propriedade também não está associada ao tipo primitivo. O browser executou isto para facilitar a visualização do array “sports”. Então, como foi viável ter utilizado essa propriedade se ela não existe no tipo primitivo? Por último, foi visualizado o “[[Prototype]]”, ou seja, o protótipo. Foi visto a propriedade “length” e utilizado o método “push” pois, o tipo primitivo estava ligado ao objeto global “Array” pelo protótipo. Ao ser inspecionado pelo terminal, foram vistos alguns métodos pertencentes a ele. Como foi revelado na figura 9:

**Figura 9 - Inspeção dos métodos do array no terminal do navegador**


Fonte: Elaborado pelo autor (2024).

Para justificar o nome cadeia de protótipo, notou-se que o objeto “Array” também é ligado a outro objeto global, no caso “Object”, como foi visualizado na figura 10:

**Figura 10 - Amostragem da propriedade __proto__**




Fonte: Elaborado pelo autor (2024).

Todos os objetos em JavaScript estão ligados diretamente ou indiretamente à “Object”, e seu *prototype* é “null”, ou seja, ele não possui protótipo e assim, termina a *prototype chain*. Logo, foi válido afirmar que, se uma propriedade ou método não existir em determinado objeto, ela é procurada em seu Protótipo, formando assim a cadeia de protótipos. Na comunidade de desenvolvedores, há uma famosa frase que exemplifica toda essa cadeia: “tudo em JavaScript é objeto”.

Contudo, é importante frisar que é possível utilizar classes em JavaScript, porém elas são consideradas *syntatic sugar* ou açúcar sintático. Pois por baixo dos panos, elas são convertidas para a herança baseada em protótipos.

Sabendo que o JS possui tipagem dinâmica, chegou-se ao TypeScript, que foi desenvolvido como uma extensão do JavaScript para adicionar suporte a tipos estáticos. Ou seja, ele torna o JavaScript uma linguagem fortemente tipada. A principal vantagem do TypeScript é que ele oferece uma integração mais estreita com editores de código, permitindo

a detecção precoce de erros (TypeScript, 2023). Além disso, o código TypeScript é convertido em JavaScript, garantindo compatibilidade com todos os ambientes que suportam JavaScript (TypeScript, 2023).

Na figura 11, utilizou-se Typescript para declarar o tipo “GenderType”, o qual pode receber três valores definidos; uma interface “AddressProps” que define suas propriedades e que tipos ela pode receber; e outra interface “UserProps” que também define os tipos de suas propriedades e diz que a propriedade “gender” precisa ser do tipo “GenderType” e “address” precisa ser um objeto do tipo “AddressProps”.

**Figura 11 - Tipos no TypeScript**

```typescript
type GenderType = "Male" | "Female" | "Other";

interface AddressProps {
  street: string;
  number: number;
  neighborhood: string;
  city: string;
  state: string;
}

interface UserProps {
  name: string;
  age: number;
  gender: GenderType;
  address: AddressProps;
}
```

Fonte: Elaborado pelo autor (2024).

Agora, na figura 12, visualizou-se a declaração do objeto “user” e a definição que ele é do tipo “UserProps”. Este objeto recebe todas as propriedades exigidas pelo seu tipo, caso não faça ou tente adicionar uma propriedade inexistente, o Typescript chama a atenção com um erro. Evitando assim, futuras inconsistências e até mesmo erros mais severos na aplicação.

**Figura 12 - Criando um objeto com base no tipo**

```
const user: UserProps = {
  name: 'Fulano',
  age: 32,
  gender: 'Other',
  address: {
    number: 87,
    street: 'Logo ali',
    city: 'São Paulo',
    neighborhood: 'Liberdade',
    state: 'São Paulo',
  }
}
```

Fonte: Elaborado pelo autor (2024).

Ademais, um estudo realizado por Zheng Gao, C. Bird e Earl T. Barr em 2017 examinou a eficácia dos sistemas de tipos estáticos, como o Flow do Facebook e o TypeScript da Microsoft, na detecção de bugs em código JavaScript. O estudo revelou que esses sistemas de tipos estáticos foram capazes de detectar com sucesso 15% dos bugs (Gao; Bird; Barr, 2017).

Para concluir este capítulo e destacar a relevância das tecnologias escolhidas no panorama atual, foram examinados alguns dados relevantes da pesquisa "Ecossistema de desenvolvedores". Essa pesquisa aponta que o JavaScript é a linguagem de programação mais adotada no momento, com 65% dos entrevistados afirmando seu uso. Notavelmente, em 40% desses casos, o TypeScript é utilizado em conjunto, evidenciando sua importância crescente no desenvolvimento. Quanto aos frameworks, o React.js se destaca como o mais empregado por 55% dos desenvolvedores, enquanto o Vue.js segue na preferência de 35%, ocupando a segunda posição. Esses dados não apenas sublinham a popularidade dessas tecnologias, mas também reforçam sua relevância no contexto atual do desenvolvimento de software.

### *5.2.2 Framework Vue.js*

Para o desenvolvimento dessa aplicação, foi utilizado o Vue.js. Segundo a documentação oficial, Vue, um projeto independente e orientado pela comunidade, foi criado por Evan You em 2014 como um projeto pessoal e evoluiu rapidamente com o apoio da

comunidade de desenvolvedores (Vue.js, 2023). Além disso, o projeto é mantido por uma equipe de membros em tempo integral e voluntários de todo o mundo, com Evan You atuando como líder do projeto.

O Vue 3 é descrito como a versão atual e mais recente do Vue, contendo novos recursos que não estão presentes no Vue 2, como Teleport, Suspense e múltiplos elementos raiz por template. Apesar das diferenças entre as versões, a maioria das APIs do Vue são compartilhadas entre as duas versões principais (Vue.js, 2023).

O Vue.js é retratado na documentação oficial como um framework maduro e testado em batalha (Vue.js, 2023). Ele é utilizado por mais de 1,5 milhão de usuários em todo o mundo e tem sido adotado por organizações renomadas em várias capacidades, incluindo Wikimedia Foundation, NASA, Apple, Google, Microsoft, GitLab, Zoom, Tencent, Weibo, Bilibili e Kuaishou (Vue.js, 2023).

Em relação à performance, a sua documentação afirma que o Vue 3 é um dos frameworks *FrontEnd* de melhor desempenho (Vue.js, 2023). Ele supera React e Angular em cenários de teste de estresse e compete de forma igualitária contra alguns dos frameworks não-Virtual-DOM mais rápidos em benchmarks. No entanto, foi importante notar que benchmarks sintéticos como os acima focam no desempenho bruto de renderização com otimizações dedicadas e podem não ser totalmente representativos dos resultados de desempenho do mundo real (Vue.js, 2023).

O React foi pioneiro no padrão Virtual DOM e segundo a documentação deste framework, "O virtual DOM (VDOM) é um conceito de programação onde uma representação ideal, ou 'virtual', da interface do usuário é mantida em memória e sincronizada com o DOM "real" (React, 2023). Bibliotecas não-Virtual-DOM são bibliotecas que não aplicam esse padrão, e fazem alterações direto no DOM, como o JQuery por exemplo.

O Vue.js é também caracterizado como um framework leve. Um aplicativo Vue *"hello world*" que utiliza apenas as APIs mínimas necessárias, tem um tamanho de pacote inicial de apenas 16 kb quando minificado e comprimido. A documentação também destacou que muitas das APIs do Vue são "tree-shakable", o que significa que recursos não utilizados não são incluídos no pacote final (Vue.js, 2023).

Um ponto relevante do Vue.js, é que o mesmo utiliza de componentização para construir as páginas da Web. Ou seja, partes menores são divididas e colocadas em componentes, como um botão por exemplo, tornando assim esse componente reutilizável e usando o mesmo em várias partes do site. Em projetos Vue, é comum o uso de um formato de arquivo especial conhecido como Componente de Arquivo Único (Single-File Component, ou

SFC). Este formato permite que desenvolvedores encapsulem a lógica do componente (JavaScript), o template (HTML) e os estilos (CSS) em um único arquivo com extensão *.vue (Vue.js, 2023).

### *5.2.3 Conclusão sobre as Tecnologias para o Frontend*

Em síntese, o JavaScript é uma linguagem de programação formidável e flexível, amplamente utilizada no desenvolvimento Web. A linguagem continua a evoluir, oferecendo cada vez mais recursos e funcionalidades que a mantêm na vanguarda das tecnologias Web. Além de que, a introdução de sistemas de tipos estáticos como TypeScript trouxe benefícios significativos para o desenvolvimento de aplicações Web. Os sistemas de tipos estáticos oferecem vantagens como melhor integração com editores de código e documentação mais eficaz. No contexto de frameworks JavaScript, o Vue.js se destaca não apenas por sua flexibilidade e facilidade de integração, mas também por sua performance otimizada. E com essas tecnologias, torna-se possível a construção do frontend da nossa aplicação.

# 6. ARQUITETURA DA APLICAÇÃO

Neste capítulo, será apresentada a arquitetura geral da aplicação, detalhando a organização e a estrutura das diferentes partes do sistema. Serão abordados tanto o backend quanto o frontend, explicando a lógica por trás da escolha das tecnologias e a forma como foram implementadas para alcançar os objetivos do projeto.

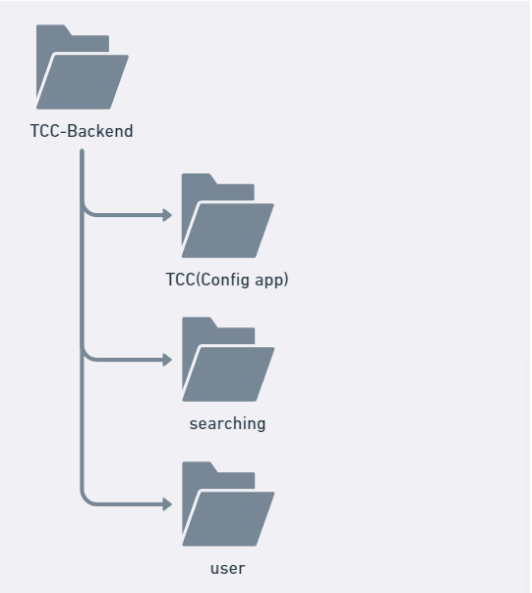
## 6.1 Aplicação Backend

Esta seção foi focada na organização e estrutura do backend da aplicação. Serão discutidos os principais componentes, como a estrutura dos diretórios, a organização dos aplicativos dentro do Django, e as práticas adotadas para gerenciamento e autenticação de usuários.

### *6.1.1 Organização de Apps no Django*

O Django permite que os desenvolvedores criem os chamados “apps” dentro de seus projetos. Apps são pacotes Python que provêm um conjunto de recursos, esses podem ser reutilizados em outros projetos (Django Project, 2023). O projeto “TCC-Backend” possui três diretórios: “TCC”, “searching” e “user”, como mostrado na figura 13.

**Figura 13 - Diretórios do backend**




Fonte: Elaborado pelo autor (2024).

O diretório “TCC” é o principal. Utilizado para as configurações do nosso projeto, nele há um arquivo chamado “settings.py” que contém, desde registros de apps criados, informações para acessar o banco de dados, configuração de fuso horário até configurações de bibliotecas externas.

Por último, tanto o diretório “searching”, quanto “user” são apps desenvolvidos com recursos específicos. Como diz o nome, searching significa pesquisando, contendo assim os modelos e rotas das APIs necessárias para que um usuário possa realizar uma pesquisa em nossa aplicação e criar um curso com objetos de aprendizagem dispostos para busca. Esse app está fortemente conectado com o app “user”. Esse último, contém os modelos de usuários, além de rotas e APIs para autenticação, criação de conta e recursos de autorização utilizados pelo app de busca e pelo frontend.

### *6.1.2 Utilização do Django Admin*

Django oferece uma interface administrativa automática chamada de “admin”, que usa metadados dos modelos para criar uma interface centrada no modelo. Esta interface é projetada principalmente como uma ferramenta interna para organizações e não deve ser a base para toda a interface do usuário (Django Project, 2023).

Essa interface será utilizada principalmente para o gerenciamento dos usuários por administradores,  usuários que podem ser tanto outros administradores, quanto funcionários ou usuários comuns. Ele também pode ser utilizado para alterar dados de algum modelo, porém foi desenvolvido uma interface própria para tal propósito no Frontend.

### *6.1.3 Extensão do AbstractUser e BaseUserManager*

Além do site de administrador, o framework já provém um sistema de autenticação e de usuários; realizou-se o uso desse sistema parcialmente e para isso, estendeu-se o usuário padrão do Django. Esse usuário foi aproveitado tanto para nosso Frontend, quanto para o site de administração gerado, removeu-se algumas propriedades que são herdadas da classe AbstractUser e adicionou-se outras para que o modelo conseguisse se adequar às necessidades da aplicação, como mostra a figura 14.

Foi criado a classe ‘BaseUserManager’ que sobrescreve os métodos ‘create_user’ e ‘create_superuser’, como mostra a figura 15, para que se adequem ao nosso usuário. Isso

permitirá que, ao iniciar a aplicação, a criação de um usuário administrador seja feita pelo terminal do servidor e assim, possibilitar o acesso ao painel administrador.

Com acesso ao painel, o administrador é capaz de dar as devidas permissões para outros usuários. Contudo, para a autenticação e autorização de usuários na aplicação Frontend, não foi feita a utilização das estratégias de autenticação fornecidas pelo Django e utilizadas no Django Admin, em vez disso utilizou-se de JWT tokens, que será visto a seguir.

**Figura 14 - Estendendo modelo User**

```python
class User(authModels.AbstractUser):
  CATEGORY_CHOICES = [
    ('GRADUATE', 'Graduate Student'),
    ('MASTERING', 'Mastering Student'),
    ('PHD', 'PHD Student'),
    ('PROFESSOR', 'Professor'),
  ]

  id = models.BigAutoField(primary_key=True)
  name = models.CharField(max_length=100)
  email = models.EmailField(max_length=100, unique=True)
  password = models.CharField(max_length=100)
  lattes = models.URLField(max_length=100, null=True, blank=True)
  googleScholar = models.URLField(max_length=100, null=True, blank=True)
  researchGate = models.URLField(max_length=100, null=True, blank=True)
  orcid = models.URLField(max_length=100, null=True, blank=True)
  github = models.URLField(max_length=100, null=True, blank=True)
  course = models.CharField(max_length=100, null=True, blank=True)
  category = models.CharField(choices=CATEGORY_CHOICES, default='GRADUATE', max_length=50, blank=True)
  oia = models.BooleanField(default=False, blank=True)

  first_name = None
  last_name = None
  username = None

  USERNAME_FIELD = "email"
  REQUIRED_FIELDS = ['name']

  objects = UserManeger()
```

Fonte: Elaborado pelo autor (2024).

**Figura 15 - Sobrescrevendo create_user e create_superuser**

```python
class UserManeger(authModels.BaseUserManager):
    def create_user(
        self,
        name: str,
        email: str,
        password: str = None,
        is_staff=False,
        is_superuser=False
    ) -> "User":
        if not email:
            raise ValueError("User must have an email")
        if not name:
            raise ValueError("User must have an name")

        user = self.model(email=self.normalize_email(email))
        user.name = name
        if password:
            user.set_password(password)
        user.is_Active = True
        user.is_staff = is_staff
        user.is_superuser = is_superuser
        user.save()

        return user

    def create_superuser(self, name: str, email: str, password: str = None) -> "User":
        user = self.create_user(
            name=name,
            email=email,
            password=password,
            is_staff=True,
            is_superuser=True
        )
        user.save()

        return user
```

Fonte: Elaborado pelo autor (2024).

### *6.1.4 Autorização por meio de JSON Web Tokens*

Nesta aplicação, a autorização é gerenciada através de JSON Web Tokens (JWTs). Após o processo de autenticação, no qual o usuário comprova sua identidade utilizando email e senha, um token JWT será gerado. Este token serve como uma chave de acesso, permitindo ao usuário autorizar-se e, consequentemente, acessar e utilizar os recursos disponíveis na aplicação.

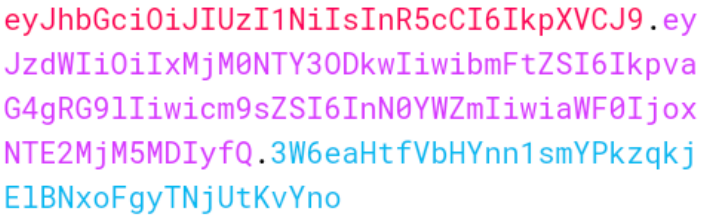
O JSON Web Token (JWT) é um padrão aberto, definido pela RFC 7519, que oferece um método compacto e autossuficiente para a comunicação segura de informações através de um objeto JSON entre entidades comunicantes (JWT.io, 2023).

O JWT é digitalmente assinado, isso significa que a informação contida nele pode ser verificada e confiável. Dependendo da natureza da chave utilizada para assinar o token, seja ela secreta com o algoritmo HMAC ou um par de chaves público/privado usando RSA ou ECDSA, é possível assegurar a origem e a integridade da informação (JWT.io, 2023).

Um dos principais usos do JWT é na autorização. Após a autenticação bem-sucedida de um usuário em um sistema, um JWT é gerado e retornado ao usuário. Este token, quando incluído em solicitações subsequentes, permite que o usuário acesse rotas, serviços e recursos específicos que são permitidos com esse token Além da autorização, o JWT também se mostra eficaz na troca segura de informações entre as partes. A capacidade de assinar o token garante que os remetentes são quem afirmam ser. Mais crucialmente, a assinatura garante que o conteúdo do JWT não foi adulterado durante a transmissão (JWT.io, 2023).

A estrutura do JWT é composta por três partes distintas: o cabeçalho (header), a carga útil (payload) e a assinatura (signature). Todas as partes são separadas por ponto. Na figura 15, vê-se um JWT e na figura 16 as partes decodificadas do mesmo. O cabeçalho identifica o tipo do token e o algoritmo de assinatura utilizado; na figura 15 ele é codificado em base 64 e está em vermelho. A carga útil contém as reivindicações ou declarações sobre uma entidade, geralmente o usuário; na figura 16 está em roxo e também codificado em base 64. A assinatura é gerada pelo algoritmo de criptografia, no caso o HS256, e valida a origem e a integridade do token; na figura 15 ela está em azul, (JWT.io, 2023).

**Figura 16 - JSON Web Token**




Fonte: Criado em https://jwt.io/.

Como vê-se na figura 17, a assinatura é gerada pela combinação do header e payload em base64, mais a chave secreta. Logo, se houver adulteração do payload, por exemplo, o token é invalidado na decodificação do mesmo.

Neste projeto, foi criada uma função utilitária que cria um JWT a partir de um usuário. Veja a figura 18.

**Figura 17 - Partes de um JWT**


Fonte: Criado em https://jwt.io/.

**Figura 18 - Função utilitária para criação do token**

```python
def createToken(user):
    payload = dict(
        id=user.id,
        exp=time.mktime((datetime.datetime.now() + datetime.timedelta(hours=24)).timetuple()) * 1000,
        iat=datetime.datetime.now(),
        isStaff=user.is_staff
    )
    token = jwt.encode(payload, settings.JWT_SECRET, algorithm="HS256")

    return token
```

Fonte: Elaborado pelo autor (2024).

Essa função utilitária, utiliza da biblioteca pyjwt. Utiliza-se do método encode para criar o jwt, para isso é preciso fornecer um payload com: id do usuário (para que seja possível

identificar o usuário pelo token), exp e iat são respectivamente as datas de expiração e criação do token, e um valor booleano "isStaff", que é utilizado pelo frontend para dizer se o usuário tem permissão para acessar a área reservada aos funcionários e administradores. Além disso, é fornecido uma chave secreta e especificado o algoritmo a ser utilizado para a criptografia.

E para verificar se o token é válido, também foi criado uma utilitária com o intuito de validar o token, veja a figura 19. A função decodifica o token com nossa chave secreta e verifica se o token está expirado.

**Figura 19 - Função utilitária para validação do token**

```python
def isTokenValid(token):
    try:
      tokenDecoded = jwt.decode(token, settings.JWT_SECRET,  algorithms="HS256")
    except:
       return False

    expTime = tokenDecoded['exp']
    expDate = datetime.datetime.fromtimestamp(expTime/1000.0)
    currentDate = datetime.datetime.now()

    if expDate < currentDate:
        return True
    return False
```

Fonte: Elaborado pelo autor (2024).

Com ambas funções, de criação e validação, tem-se uma base para realizar a autorização em nosso sistema através de um JSON Web Token.

### *6.1.5 APIs do app user*

O app "user" contém as APIs necessárias para autorização e autenticação dos usuários em nosso sistema. Começando com a api de criação de conta: após o recebimento de uma requisição POST com todos os dados do usuário, é feita uma verificação da existência do mesmo. Se o usuário não existir, é utilizado o serializer para validar os dados e se for válido, cria-se um novo usuário. Como mostra a imagem 20.

**Figura 20 - Api de criação de usuário**

```python
class RegisterUserApi(APIView):
  def post(self, request):
    email = request.data['email']
    user = User.objects.filter(email=email).first()

    if user:
      raise exceptions.AuthenticationFailed("Email em uso")

    serializser = UserSerializer(data=request.data)

    if serializser.is_valid():
      user = User.objects.create_user(
        name=serializser.data['name'],
        email=serializser.data['email'],
        password=serializser.data['password']
      )
      user.lattes = serializser.data['lattes']
      user.googleScholar = serializser.data['googleScholar']
      user.researchGate = serializser.data['researchGate']
      user.orcid = serializser.data['orcid']
      user.github = serializser.data['github']
      user.category = serializser.data['category']
      user.oia = bool(serializser.data['oia'])

      user.save()
    else:
      raise exceptions.APIException('Dados inválidos')

    return Response(status=status.HTTP_201_CREATED)
```

Fonte: Elaborado pelo autor (2024).

Seguindo para a api de login, ela recebe email e senha de um usuário. Utilizando o método authenticate do próprio django, ele recupera o usuário tentando logar ou levanta uma exceção de credenciais incorretas. Se for possível recuperar o usuário, é retornado também para o frontend, o token de acesso "jwtToken" e o token de revalidação "refreshToken". Como mostra a imagem 21.

Essa aplicação também oferece a opção do usuário realizar o login pelo google. E para tal, foi utilizado o próprio sdk do google, recebendo um "idToken" que é enviado pelo frontend. Após fazer a verificação desse token, é retornado algumas informações, incluindo o email do usuário. A partir do email é feito uma verificação, se já existir um usuário com esse email, os tokens de acesso e revalidação são gerados normalmente. Porém, se o usuário não

existir, é criado um novo usuário e retornadas suas credenciais de acesso. Como mostra a imagem 22.

**Figura 21 - Api de login**

```python
class LoginApi(APIView):
  def post(self, request):
    email = request.data['email']
    password = request.data['password']
    user = authenticate(request, email=email, password=password)

    if user is None:
      raise exceptions.AuthenticationFailed("Credenciais Incorretas")

    jwtToken = createToken(user=user)
    refreshToken = createRefresh(user=user)
    login(request, user)

    return Response(status=status.HTTP_200_OK, data={'access_token': jwtToken, 'refresh_token': refreshToken, 'is_staff': user.is_staff})
```

Fonte: Elaborado pelo autor (2024).

**Figura 22 - Api de login com Google**

```python
class GoogleSignIn(APIView):
  def post(self, request):
    try:
      token = request.data['idToken']
      idInfo = id_token.verify_oauth2_token(token, requests.Request(), settings.CLIENT_ID)
      user = User.objects.filter(email=idInfo['email']).first()
      if not user:
        try:
          user = User.objects.create_user(name=idInfo['name'], email=idInfo['email'])
        except:
          raise exceptions.AuthenticationFailed("Email em uso")

      jwtToken = createToken(user=user)
      refreshToken = createRefresh(user=user)

      return Response(status=status.HTTP_200_OK, data={'access_token': jwtToken, 'refresh_token': refreshToken, 'is_staff': user.is_staff})
    except ValueError:
      raise exceptions.AuthenticationFailed("Token inválido")
```

Fonte: Elaborado pelo autor (2024).

Para facilitar a recuperação dos dados do usuário logado, foi criado um endpoint que recupera o usuário através do token de autenticação e retorna seus dados. Como mostra a figura 23.

E por último, o endpoint de revalidação do usuário. Quando o token de acesso expirar, basta enviar o refresh token e gerar um novo token de acesso. Como mostra a figura 24.

**Figura 23 - Api de recuperação dos dados do usuário**

```python
class RetrieveUserApi(APIView):
  def get(self, request):
    token = request.META['HTTP_AUTHORIZATION']
    if isTokenExpired(token):
      return Response({'detail': 'Token inválido'}, status=status.HTTP_401_UNAUTHORIZED)

    user = getUserFromToken(token)

    if not user:
      raise exceptions.AuthenticationFailed("Usuário não autenticado")

    serializer = RetrieveUserSerializer(user)

    return Response(serializer.data, status=status.HTTP_200_OK)
```

Fonte: Elaborado pelo autor (2024).

**Figura 24 - Api de revalidação de token**

```python
class RefresUserApi(APIView):
  def post(self, request):
    token = request.data['refresh_token']
    user = getUserFromRefresh(token)
    if user:
      token = createToken(user)
      return Response(status=status.HTTP_200_OK, data={'access_token': token})
    raise exceptions.AuthenticationFailed("Não autorizado")
```

Fonte: Elaborado pelo autor (2024).

Agora, é possível observar todas as APIs citadas acima e suas rotas na imagem 25. As mesmas estão registradas no app de configuração com o prefixo "api/user/". Logo, para acessar alguma rota exposta nesse app, utiliza-se por exemplo: "api/user/me/".

**Figura 25 - Rotas do app user**

```python
urlpatterns = [
  path('create/', views.RegisterUserApi.as_view(), name='register'),
  path('login/', views.LoginApi.as_view(), name='login'),
  path('login/google/', views.GoogleSignIn.as_view(), name='google-login'),
  path('me/', views.RetrieveUserApi.as_view(), name='me'),
  path('refresh/', views.RefresUserApi.as_view(), name='me'),
  path('logout/', views.LogoutApi.as_view(), name='logout')
]
```

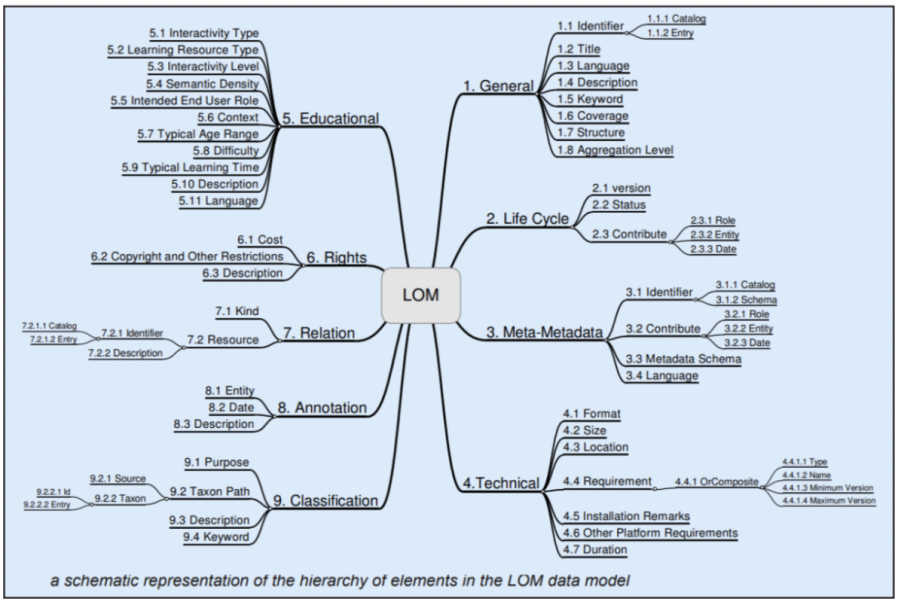
Fonte: Elaborado pelo autor (2024).

### *6.1.6 Modelos de negócio*

Chega-se agora ao app "searching", este contém toda lógica de negócio da nossa aplicação. Ou seja, é aqui que residem os modelos e APIs para a busca de objetos de aprendizagem e criação de cursos pelo usuário.

Começando pelos modelos, existem somente três. O primeiro é "LearningObject", que está abstraindo o padrão de metadados IEEE/LOM, o qual pode ser visto na figura 26. Utilizou-se apenas os seguintes atributos baseados no padrão real: "id", "title", "description", "keywords", "duration_min", "language" e adicionou-se o atributo "link" veja na figura 27.

**Figura 26 - Metadados IEEE/LOM**




Fonte: (Barker, 2005).

**Figura 27 - Modelo "LearningObject"**

```python
class LearningObject(models.Model):
  id = models.BigAutoField(primary_key=True)
  title = models.CharField(max_length=30)
  description = models.CharField(max_length=2000)
  keywords = models.CharField(max_length=100)
  link = models.CharField(max_length=500)
  duration_min = models.IntegerField(null=True, default=None)
  language = models.CharField(choices=LanguageTypes.choices)
  isPublished = models.BooleanField(default=False)
  createdBy = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, related_name='createdBy')
  editedBy = models.ForeignKey(User, on_delete=models.SET_NULL, null=True, related_name='editedBy')

  def __str__(self) -> str:
    return self.title

  class Meta:
    ordering = ['name']
```

Fonte: Elaborado pelo autor (2024).

O segundo modelo é "Course", representa um curso criado por um usuário e está relacionado ao mesmo pelo campo "user". E por terceiro, foi criado uma tabela de relação, "CourseLearningObjects" que relaciona um curso a um objeto de aprendizagem. Nesse caso observa-se uma relação de **n** para **n** (muitos para muitos), já que um mesmo objeto pode estar presentes em vários cursos e um curso pode conter múltiplos objetos. Pode-se observar esses modelos na figura 28.

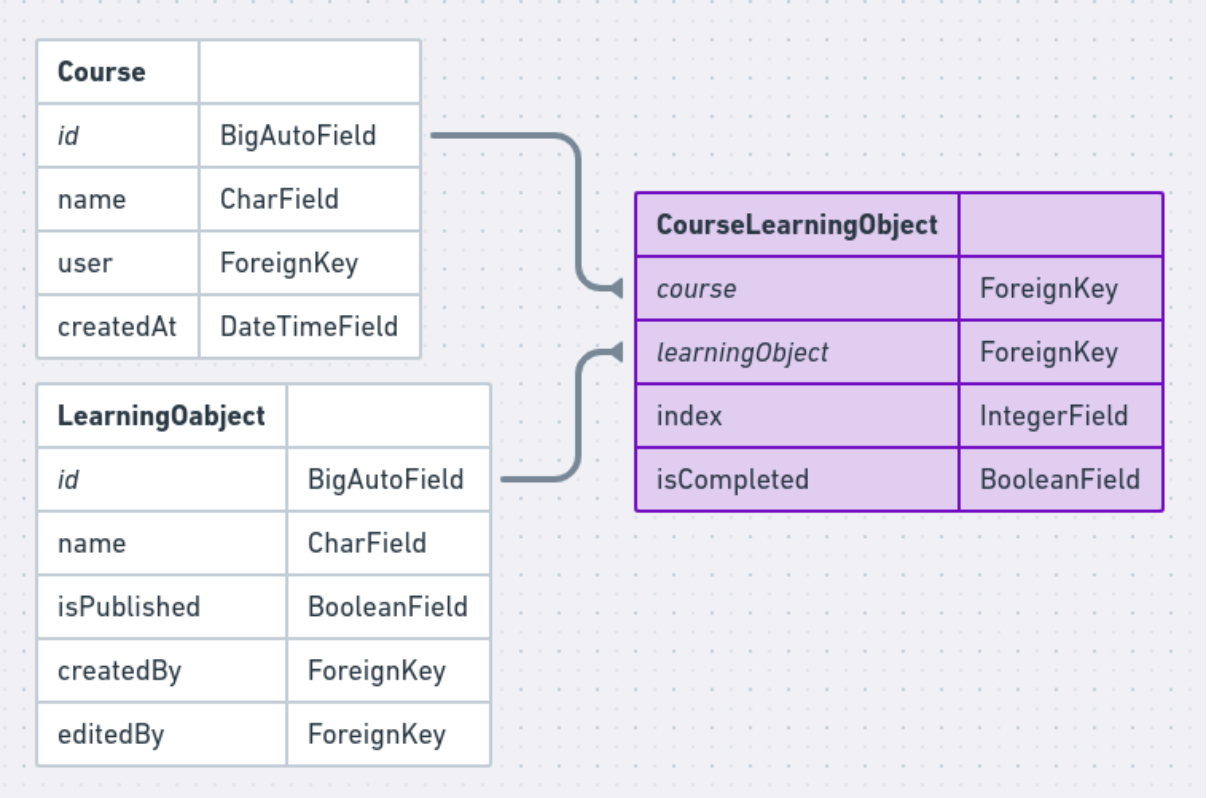
**Figura 28 - Modelos "Course" e "CourseLearningObject"**

```python
class Course(models.Model):
  id = models.BigAutoField(primary_key=True)
  name = models.CharField(max_length=30)
  user = models.ForeignKey(User, on_delete=models.CASCADE, related_name='courseUser')
  createdAt = models.DateTimeField(auto_now_add=True)

  class Meta:
    ordering = ['createdAt']

  def __str__(self):
    return self.name

class CourseLearningObject(models.Model):
  course = models.ForeignKey(Course, on_delete=models.CASCADE)
  learningObject = models.ForeignKey(LearningObject, on_delete=models.CASCADE)
  index = models.IntegerField()
  isCompleted = models.BooleanField(default=False)

  class Meta:
    ordering = ['index']

  def __str__(self):
    return self.course.name + '-' + self.learningObject.name
```

Fonte: Elaborado pelo autor (2024).

Agora na figura 29, consegue-se ver melhor essas relações. É preciso destacar na tabela "Course Learning Object" e há dois campos importantes para dados; o primeiro é o campo "index" que indica a posição do objeto no curso (se é o primeiro, segundo, etc.); e o segundo é o campo "isCompleted", que indica se o usuário já visualizou ou completou o objeto.

**Figura 29 - Diagrama de relacionamentos entre "Course" e "LearningObject"**




Fonte: Elaborado pelo autor (2024).

### *6.1.7 Níveis de acesso dos usuários*

Pode-se dividir os níveis de acesso do sistema em três: 'USER', 'STAFF' e 'ADMIN'. Os usuários normais, ou que pertencem ao grupo 'USER', podem pesquisar objetos de aprendizagem e montar cursos. Usuários com a função 'STAFF', que além de realizar as ações tomadas pelo usuário convencional, podem adicionar os objetos de aprendizagem no sistema. E por último, o usuário com a função de 'ADMIN', que além das ações tomadas pelos outros dois tipos de usuários, podem também realizar operações de edição, exclusão e publicação de objetos.

### *6.1.8 APIs do app searching*

O app "searching" contém as APIs necessárias para a realização das operações nos objetos de aprendizagem, a busca desses mesmos objetos e a criação de cursos por um usuário. Começando com a API de Objetos de aprendizagem, que utiliza dos métodos do padrão REST: 'get', 'post', 'put' e 'delete'. A partir deles, realiza-se todas nossas operações nos objetos de aprendizagem a partir de uma única rota.

O método 'get', possibilita que usuários 'STAFF' recebam uma lista paginada de objetos de aprendizagem. Essa lista é paginada e passível de aplicação de filtros de busca e publicação de objetos. Utiliza-se o método 'getLearningObjects' para aplicação dos filtros e a função 'paginate' para paginar nossos dados com um serializer, que serão úteis para a implementação da paginação no frontend. Pode-se visualizar o método 'get' na figura 30.

Já o método 'post' da API, realiza o cadastro de um objeto no nosso banco de dados. Ele também permite usuário 'STAFF' e utiliza de um serializer para criar salvar o objeto, a partir dos dados enviados para nosso endpoint. Pode-se observar o método 'post'' na figura 31.

Com o método 'put', possibilita a atualização de um objeto para usuários 'ADMIN'. Recebe-se o id através de um parâmetro da rota, e é recuperado o objeto para aquele id e atualizado do serializer para validar e salvar os dados enviados na requisição. Veja o método na figura 32.

E por fim, o método 'delete', também é de acesso para administradores, e faz a exclusão de um objeto a partir do id obtido pelo parâmetro da rota. Observe a figura 33.

**Figura 30 - Método 'get' da api de objetos de aprendizagem**

```python
def get(self, request):
    token = request.META['HTTP_AUTHORIZATION']
    validationResponse = validateStaffUser(token)

    if validationResponse:
      return validationResponse

    learningObjects = self.getLearningObjects()
    paginatedResponse = paginate(learningObjects, request, LearningObjectSerializer)

    return paginatedResponse
```

Fonte: Elaborado pelo autor (2024).

**Figura 31 - Método 'post' da api de objetos de aprendizagem**

```python
def post(self, request):
    token = request.META['HTTP_AUTHORIZATION']
    validationResponse = validateStaffUser(token)

    if validationResponse:
      return validationResponse

    serializer = CreateLearningObjectSerializer(data=request.data)
    if serializer.is_valid():
      serializer.save(createdBy=getUserFromToken(token))

    return Response(status=status.HTTP_201_CREATED)
```

Fonte: Elaborado pelo autor (2024).

**Figura 32 - Método 'put' da api de objetos de aprendizagem**

```python
def put(self, request, id):
    token = request.META['HTTP_AUTHORIZATION']
    validationResponse = validateSuperUser(token)

    if validationResponse:
      return validationResponse

    learningObject = LearningObject.objects.get(id=int(id))
    serializer = LearningObjectSerializer(instance=learningObject, data=request.data)

    if serializer.is_valid():
        serializer.save(editedBy=getUserFromToken(token))

    return Response(status=status.HTTP_200_OK)
```

Fonte: Elaborado pelo autor (2024).

**Figura 33 - Método 'delete'' da api de objetos de aprendizagem**

```python
def delete(self, request, id):
    token = request.META['HTTP_AUTHORIZATION']
    validationResponse = validateSuperUser(token)

    if validationResponse:
      return validationResponse

    learningObject = LearningObject.objects.get(id=id)
    learningObject.delete()

    return Response(status=status.HTTP_200_OK)
```

Fonte: Elaborado pelo autor (2024).

Após a API de Objetos de aprendizagem, tem-se em seguida um 'endpoint' para usuários do tipo 'ADMIN' com a função de publicar e despublicar um objeto. Essa rota de API suporta o método 'put' e através do id do objeto, realiza uma operação booleana de negação na propriedade isPublished do objeto. Desta maneira, o valor de isPublished é invertido, e o objeto é publicado ou despublicado. Pode-se observar essa lógica na figura 34.

**Figura 34 - API de publicação de objetos de aprendizagem**

```python
class PublishLearningObjectAPI(APIView):
  def put(self, request, id):
    token = request.META['HTTP_AUTHORIZATION']
    validationResponse = validateSuperUser(token)

    if validationResponse:
      return validationResponse

    learningObject = LearningObject.objects.get(id=int(id))
    learningObject.isPublished = not learningObject.isPublished
    learningObject.save()

    return Response(status=status.HTTP_200_OK)
```

Fonte: Elaborado pelo autor (2024).

Para que o usuário comum possa buscar pelos objetos publicados e criar um curso, foi disponibilizado uma API de busca que retorna dados paginados dos objetos. Através de um método 'get', ela retorna as instâncias do modelo "LearningObject" que foram publicados por um administrador e com os atributos "title", "description" ou "keywords" correspondentes a busca do usuário. Pode-se observar essa API na figura 35.

**Figura 35 - API de busca de objetos de aprendizagem publicados**

```python
class SearchObjectsAPI(APIView):
  def get(self, request):
    search = request.GET.get('search')
    learningObjects = LearningObject.objects.filter((
        Q(title__icontains=search) |
        Q(description__icontains=search) |
        Q(keywords__icontains=search)
      ) & Q(isPublished=True))

    paginatedResponse = paginate(learningObjects, request, LearningObjectSerializer)

    return paginatedResponse
```

Fonte: Elaborado pelo autor (2024).

Com o intuito de possibilitar a criar um curso com os objetos fornecidos ao usuário, foi criada a API de cursos, e a mesma segue o mesmo padrão da api de objetos de aprendizagem. Nela, foram disponibilizados os métodos 'get', 'post', 'put' e 'delete'. A diferença para a api de objetos, além do modelo usado para as operações ser diferente, os métodos recuperam o usuário do token de acesso e verificam se o usuário está acessando o curso que foi criado por ele. Desta maneira, somente o usuário tem acesso às operações no seu curso. Contudo, o método 'post' cria o curso de maneira diferente; ele utiliza uma transação para criar todos os relacionamentos entre o curso e os objetos de aprendizagem. Caso alguma operação no banco dê errado, as operações que já foram feitas são canceladas. Essa estrutura de transação pode ser observada no bloco de código 'with' na figura 36, nela foi criado um curso e uma relação entre curso e objeto de aprendizagem para cada objeto enviado na requisição.

**Figura 36 - Transação no método ‘post’ da API de cursos**

```python
def post(self, request):
  token = request.META['HTTP_AUTHORIZATION']
  user = getUserFromToken(token).id
  with transaction.atomic():
    course = Course.objects.create(name=request.data['name'], user_id=user)
    index = 0
    for object in request.data['objects']:
      CourseLearningObject.objects.create(course_id=course.id, learningObject_id=object['id'], index=index)
      index += 1

    return Response(status=status.HTTP_200_OK)
```

Fonte: Elaborado pelo autor (2024).

Por último, há três APIs restantes nesse app. A Primeira recupera os objetos de aprendizagem relacionados a um curso. A segunda, recupera as relações entre o curso e os objetos, já que essas relações guardam informações sobre a posição do objeto e se ele já foi completado pelo usuário. E a última, segue a mesma lógica de publicação de objetos, porém ela completa ou descompleta o objeto relacionado ao curso do usuário.

Essas APIs foram registradas nas rotas com o prefixo 'api/', logo para acessá-las precisa-se incluir esse prefixo nas rotas mostradas na figura 37. Por exemplo: 'api/learning-objects/'.

**Figura 37 - Rotas do app searching**

```python
urlpatterns = [
  path('learning-objects/', views.LearningObjectAPI.as_view(), name='learning-objects'),
  path('learning-objects/<int:id>/', views.LearningObjectAPI.as_view(), name='learning-objects'),
  path('toggle-publish/<int:id>/', views.PublishLearningObjectAPI.as_view(), name='toggle-publish'),
  path('search-objects/', views.SearchObjectsAPI.as_view(), name='published-objects'),
  path('course/', views.CourseAPI.as_view(), name='couse'),
  path('course/<int:course>/', views.CourseAPI.as_view(), name='couse'),
  path('course-objects/<int:course>/', views.ObjectsFromCoursesAPI.as_view(), name='couse-objects'),
  path('course-objects-progression/<int:course>/', views.CouseLearningObjectsAPI.as_view(), name='couse-object-relation'),
  path('course-toogle-object/', views.ToogleObjectInCouseAPI.as_view(), name='couse-toogle-object'),
]
```

Fonte: Elaborado pelo autor (2024).

## 6.2 Aplicação Frontend

Esta seção abordará a organização e a estrutura do frontend da aplicação. Serão descritos os diretórios e arquivos principais, explicando a função de cada um e como eles contribuem para a construção da interface do usuário. Além disso, será detalhado o processo de integração com o backend e a implementação das funcionalidades essenciais para a experiência do usuário.

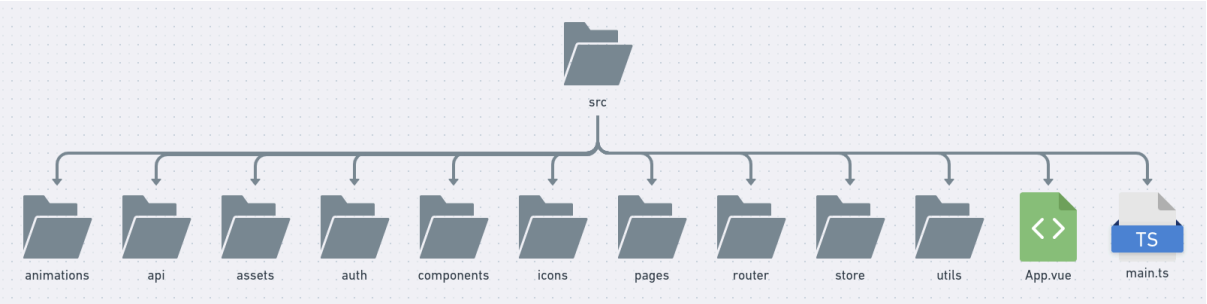
### *6.2.1 Organização dos diretórios no Vue*

Toda lógica do frontend do projeto está localizada dentro da pasta "src". Sendo assim, neste tópico, será discutido os seus subdiretórios. O primeiro é o diretório "animations", nele ficam localizados os efeitos de animação para nossa interface. O segundo, é o "api" e nele fica localizado o "fetcher", e todas as funções que fazem requisições à API. O terceiro, é o diretório "assets" e nele ficam localizados os arquivos css, imagens e se necessário, outros arquivos de mídia. Em quarto, tem-se a pasta "auth", e é nela que ficam localizadas todas as telas e lógicas para login e criação de conta do usuário. Em seguida, o diretório "components", e aqui ficam salvos todos os componentes reutilizáveis criados com o Vue. Depois tem-se o diretório "icons", nele encontra-se a configuração da biblioteca "fontawesome" de ícones; se for preciso utilizar algum ícone, basta registrá-lo nesse arquivo. Em sétimo, a pasta "pages", aqui localizam-se todas as páginas restantes da aplicação e suas lógicas. Por oitavo, o "router", e nesse diretório são configuradas as rotas de nossa aplicação, tanto rotas protegidas (rotas que requerem autenticação) quanto rotas públicas. Em penúltimo, tem-se o diretório "store". Aqui encontram-se algumas "stores" do projeto, feitas para

armazenarem determinados tipos de dados e utilizarem os mesmos em diferentes componentes, e foram construídas utilizando o "reactive" do próprio Vue. E por último o diretório utils, usado para armazenar funções utilitárias (essas funções implementam algum cálculo, ou lógica comum, que podem ser reutilizadas em várias partes do código).

Além desses diretórios, há dois arquivos importantes. O primeiro é o "App.vue", este arquivo renderiza o router da aplicação do projeto, e consequentemente as páginas que estão associadas ao router. O segundo é o "main.ts", este monta a aplicação e configura as bibliotecas que tem a necessidade de configuração. Pode-se observar o diretório src na figura 38.

**Figura 38 - Subdiretórios de “src”**

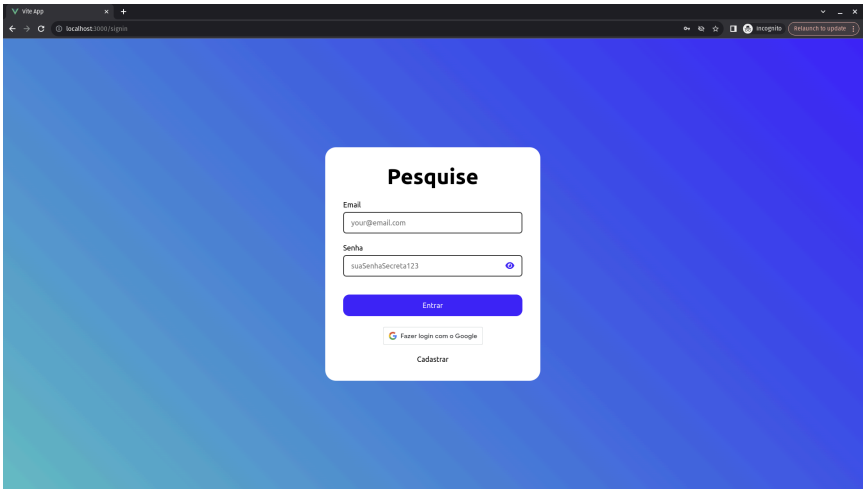



Fonte: Elaborado pelo autor (2024).

### *6.2.2 Páginas de autenticação e formulários*

Atualmente, o projeto possui 2 páginas de autenticação localizadas na pasta "auth": "SignIn" para a autenticação do usuário e "SignUp" para criação de conta. Na página de login, tem-se um campo para e-mail, outro para senha e dois botões, um para realizar o login através de nossa api e outro para realizar o login através da implementação com a api do Google. E por fim, um link para a página de cadastro. Pode-se observar a página descrita acima na figura 39.

**Figura 39 - Página de login**




Fonte: Elaborado pelo autor (2024).

A página de cadastro segue o mesmo layout da página de login, porém, com todos os campos necessários para a criação da conta, que são: nome, email, senha, confirmar senha, curso e categoria. E campos opcionais: Lattes, Google Scholar, Research Gate, Linkedin, Orcid e Github.

Todos os formulários criados no sistema seguem a mesma arquitetura de desenvolvimento. Como exemplo, a página de "SingIn" possui um formulário simples: primeiro é criada a estrutura, todos os elementos necessários para a criação de um formulário foram componentizados. Ao observar a figura 40, vê-se que o componente "<Form />" é importado da biblioteca "vee-validate", a qual é utilizada para validação de formulário no Vue e recebe uma propriedade chamada de "validation-schema", que serve para a validação dos campos. Logo após, vê-se o componente "<Input />" que representa no caso um input de texto simples, porém na aplicação do projeto, foram criados inputs de checkbox e select. Esse input, recebe a propriedade "name", usada para identificar o campo no schema de validação. Há o componente "<Button />", que envia o formulário e possui propriedades de carregamento e desativado, essas propriedades alteram o estilo do botão. E por fim, existem 2 componentes que não são necessários para a criação de um formulário; porém o primeiro "<GoogleLogin />, permite que faça-se login com o Google e o segundo componente é um link.

**Figura 40 - Formulário de login**

```vue
<Form
    class="form"
    @submit="onSubmit"
    :validation-schema="schema"
    v-slot="{ meta, isSubmitting }"
  >
    <Input
      name="email"
      type="email"
      label="Email"
      placeholder="your@email.com"
    />
    <Input
      name="password"
      type="password"
      label="Senha"
      placeholder="suaSenhaSecreta123"
    />
    <Button :disabled="!meta.valid || isSubmitting" :is-loading="isSubmitting"
      >Entrar</Button
    >
    <GoogleLogin :callback="googleCallback" />
    <router-link class="link" to="/signup">Cadastrar</router-link>
  </Form>
```

Fonte: Elaborado pelo autor (2024).

O schema é usado na validação do formulário e contém as regras de validação. Foi empregado da biblioteca "yup" para a criação do mesmo e pode-se observar na figura 41 como ele é feito. Ao importar o yup, cria-se o schema com o método object, que recebe um objeto cujas propriedades são também outros métodos do yup. Para esse formulário, foram criadas duas propriedades: "email" e "password", repare que elas são escritas da mesma forma que as propriedades "name" passadas no componente de input. Para a propriedade "email",

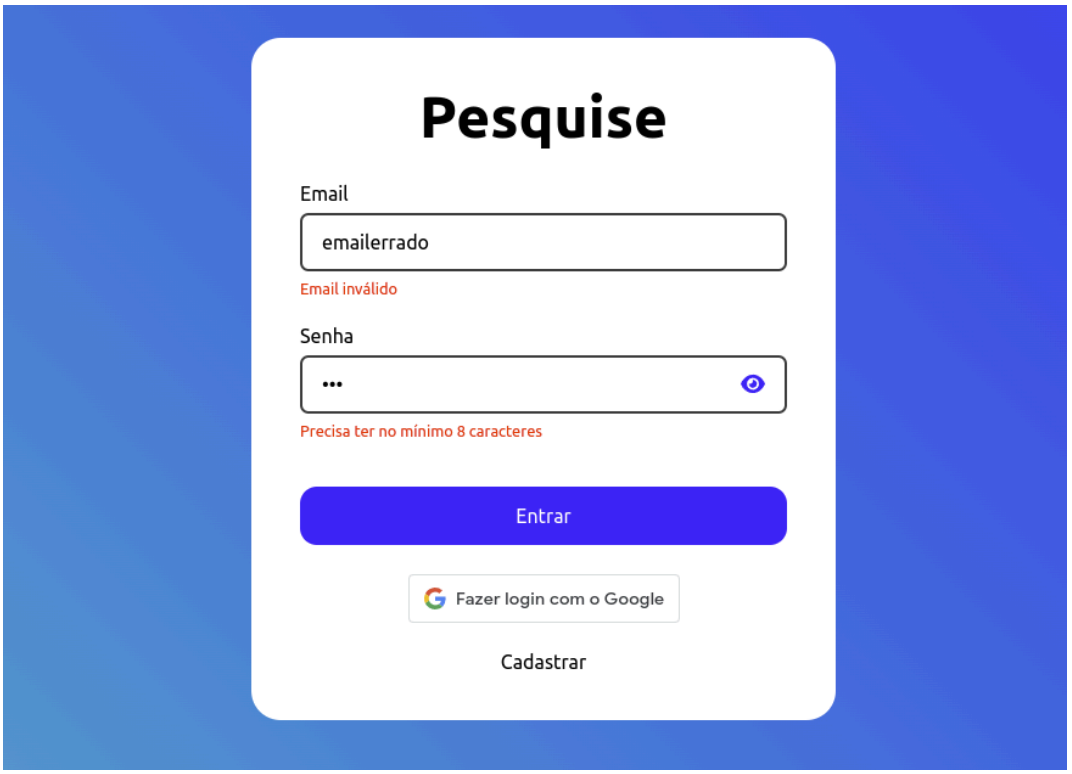
informa-se que ela é uma string obrigatória e utiliza-se um método de validação de email provido pela própria biblioteca, informa-se também as mensagens de erro. E para a propriedade "password", instrui-se também que ela é uma string obrigatória e que precisa ter tamanho mínimo de 8 caracteres. Veja na figura 42, a validação em ação na tentativa de enviar o formulário de login com dados incorretos. O formulário é invalidado, aparecendo as mensagens de erro embaixo dos inputs. Só é possível enviar a requisição para a api de login quando os dados inseridos no formulário estiverem no formato correto.

**Figura 41 - Schema de validação**

```
const schema = yup.object({
  email: yup.string().required("Email obrigatório").email("Email inválido"),
  password: yup
    .string()
    .required("Senha obrigatória")
    .min(8, "Precisa ter no mínimo 8 caracteres"),
});
```

Fonte: Elaborado pelo autor (2024).

**Figura 42 - Formulário de login com erros de validação**

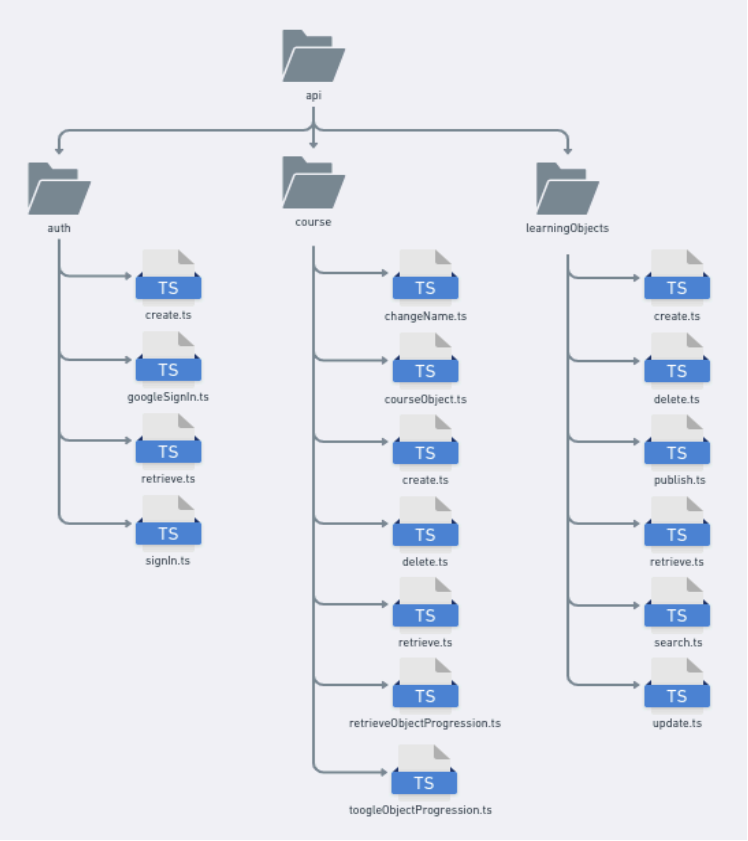



Fonte: Elaborado pelo autor (2024).

### *6.2.3 Arquitetura de requisições à API*

Como visto anteriormente, no diretório api ficam localizadas as funções que realizam as requisições para  a api na aplicação do projeto. Veja como é feita a organização dessas funções e como elas são construídas: na figura 43 é possível visualizar a organização. Separa-se em subdiretórios as funções de requisições que ficam relacionadas com suas determinadas entidades, propósitos ou APIs. Esses subdiretórios são: "auth", que armazena as requisições com o propósito de autenticação e autorização; "course", que reúne as requisições relacionadas às APIs que exercem alguma ação sobre os cursos do usuário; "learningObjects", que segue a mesma ideia do diretório "course", porém, tendo como principal entidade os objetos de aprendizagem. E cada arquivo dentro desses subdiretórios, possuem o nome de uma ação a ser exercida em suas entidades relacionadas.

**Figura 43 - Arquitetura do diretório "api"**




Fonte: Elaborado pelo autor (2024).

Observa-se na figura 44, o arquivo "signIn.ts" presente no subdiretório "auth". Primeiramente, é importado o "fetcher" (será falado sobre ele no capítulo seguinte) e em seguida é declarado uma interface chamada de "ResponseProps", que indica o tipo da resposta

da API do projeto, em caso de sucesso. Cria-se aqui uma função denominada "signIn", que recebe como parâmetro os dados necessários para realizar a requisição. Chamados então o fetcher, indica-se no caso que será utilizado o método "post" para realizar a requisição, passa-se a interface criada acima, define-se a rota a ser acessada em nosso backend e por último é fornecido o corpo da requisição como um objeto. Em caso de sucesso, o fetcher irá retornar um objeto de resposta contendo os dados esperados pela interface e o status da requisição. Em caso de erro, ele joga um erro que pode ser tratado em nossas páginas para uma melhor experiência de usuário.

Todas as requisições feitas para o backend partindo do frontend, são construídas utilizando a mesma lógica e respeitando a estrutura explicada neste capítulo, com exceção da requisição para revalidação de sessão, a qual será vista a seguir.

**Figura 44 - Função de singIn**

```
1  import fetcher from "..";
2
3  export interface ResponseProps {
4    access_token: string;
5    refresh_token: string;
6    is_staff: boolean;
7  }
8
9  const signIn = async (email: string, password: string) => {
10   const response = await fetcher("post")<ResponseProps>(`user/login/`, {
11     email,
12     password,
13   });
14
15   return response;
16 };
17
18 export default signIn;
19
```

Fonte: Elaborado pelo autor (2024).

### *6.2.4 Fetcher, axios, autorização e revalidação*

Para construir o fetcher do projeto, foi utilizado o "axios", um cliente http. Veja na figura 45 a criação da uma instância do axios para ser utilizada em nossa aplicação. A instância se chama "http" e recebe a url base da para acessar a api.

**Figura 45 - Instância do axios**

```
export const http = axios.create({ baseURL: import.meta.env.API_BASE_URL });
```

Fonte: Elaborado pelo autor (2024).

A partir dessa instância "http" foi criado o fetcher. Pode-se observar o mesmo na figura 46. O fetcher é uma função que, a partir do método http informado, chama a instância do axios com ou sem o corpo da requisição. No caso do projeto, não é feito o envio do corpo para as requisições do tipo "get" ou "delete". E por fim, essa função retorna o corpo da resposta, presente em "data" e o status da requisição, presente na propriedade "status".

**Figura 46 - Fetcher**

```
const fetcher =
  (method: Method) =>
  async <ResponseProps>(
    path: string,
    body?: Record<string, any>,
    config?: AxiosRequestConfig
  ) => {
    const { data, status } =
      method === "get" || method === "delete"
        ? await http[method]<ResponseProps>(path, config)
        : await http[method]<ResponseProps>(path, body, config);

    return { data, status };
  };
```

Fonte: Elaborado pelo autor (2024).

Com a biblioteca de requisições, é possível "interceptar requisições ou respostas" (axios, 2023), utilizando os "interceptors". Veja na figura 47, a utilização deste recurso para realizar o token de acesso no cabeçalho "Authorization" de todas as requisições se o mesmo token estiver presente na sessão.

**Figura 47 - Interceptor na requisição**

```
1  http.interceptors.request.use(
2    (config: AxiosRequestConfig) => {
3      const token = session.accessToken;
4      if (token)
5        return {
6          ...config,
7          headers: {
8            ...config.headers,
9            Authorization: session.accessToken,
10         },
11       };
12
13     return config;
14   },
15   (error) => {
16     return Promise.reject(error);
17   }
18 );
```

Fonte: Elaborado pelo autor (2024).

Contudo, em algum momento, o token de acesso irá expirar. Ao expirar, a API retornará "Unauthorized" com status 401. Utilizando de interceptors, é possível ver que na figura 48 intercepta-se a resposta de uma requisição e é realizada a seguinte verificação: se o status da resposta for 401 e houver o token de revalidação na sessão, então revalida-se o usuário e caso contrário, é continuado com o retorno do erro. Ao revalidar o usuário, utiliza-se a seguinte lógica: é enviado uma requisição para o endpoint de revalidação que nos retorna um novo token de acesso, depois é definido o novo token de acesso em nossa sessão e é refeito a requisição que retornou 401 na linha 17 da figura 42, reenviando os dados com o novo token. Se ocorrer algum erro durante a revalidação do usuário, é excluída a sessão do mesmo e recarregada a página. Desta maneira, quando ocorrer um erro de não autorizado em nosso sistema, o usuário não precisará exercer nenhuma ação, somente quando o token de revalidação expirar (no caso, ele precisará logar novamente). Mas caso o token de revalidação

não esteja expirado, esse erro passará despercebido sem incomodar ou atrapalhar na experiência de uso da plataforma.

**Figura 48 - Interceptor na resposta**

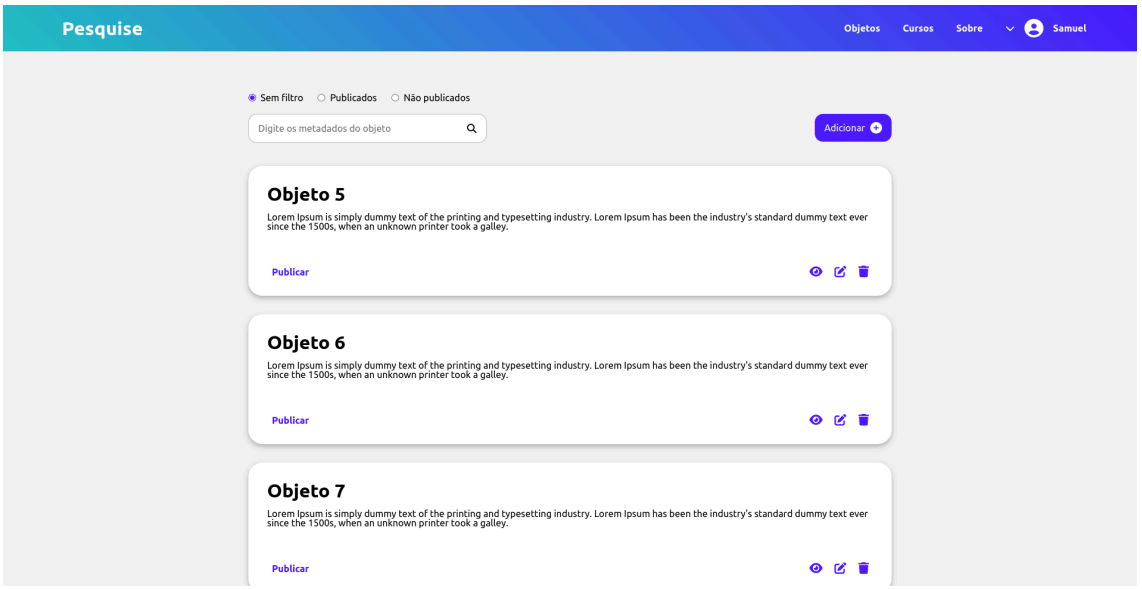
```
http.interceptors.response.use(
  (response) => response,
  async (error) => {
    if (
      (error as AxiosError).response?.status === 401 &&
      session.refreshToken
    ) {
      try {
        const { data } = await axios.post<{ access_token: string }>(
          `${import.meta.env.API_BASE_URL}user/refresh/`,
          {
            refresh_token: session.refreshToken,
          }
        );
        session.setAccessToken(data.access_token);
        error.config.headers["Authorization"] = session.accessToken;
        return axios.request(error.config);
      } catch {
        session.deleteSession();
        user.clearStore();
        window.location.reload();
      }
    }
    return Promise.reject(error);
  }
);
```

Fonte: Elaborado pelo autor (2024).

### *6.2.5 Páginas de CRUD, pesquisa e cursos*

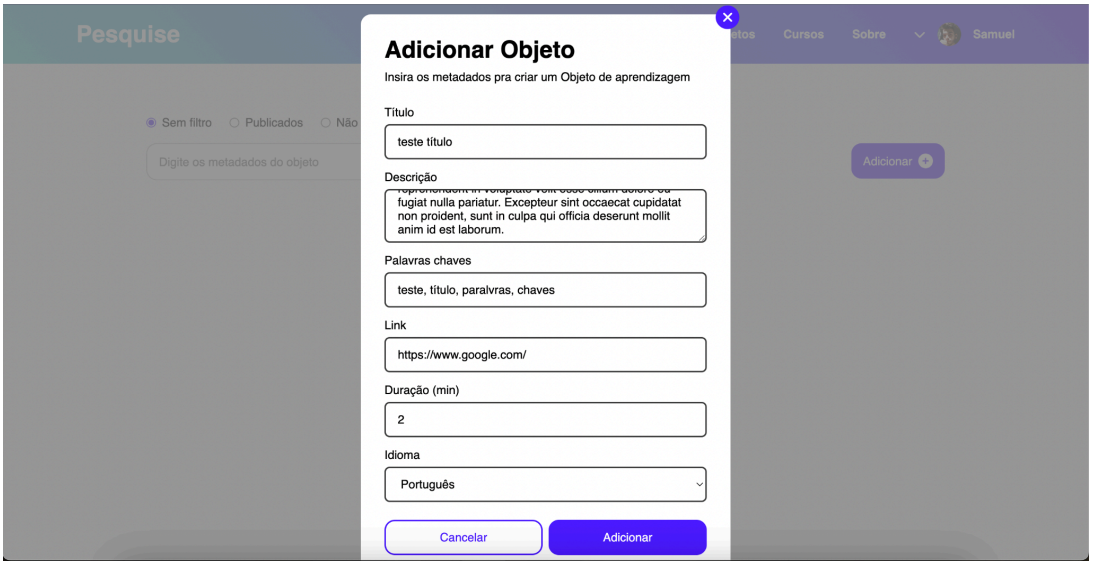
Na figura 49, pode-se observar a página responsável pela criação (create), leitura (read), edição (update) e exclusão (delete) de dos objetos de aprendizagem ou o nosso CRUD. Nessa rota, através dos modais nas figuras 50, 51 e 52 consegue-se realizar as operações listadas acima e publicar o objeto, tornando-o disponível para a busca de um usuário. É possível também filtrar os objetos por publicados e não publicados e realizar uma busca por algum metadado de um objeto.

**Figura 49 - Página do CRUD de objetos**



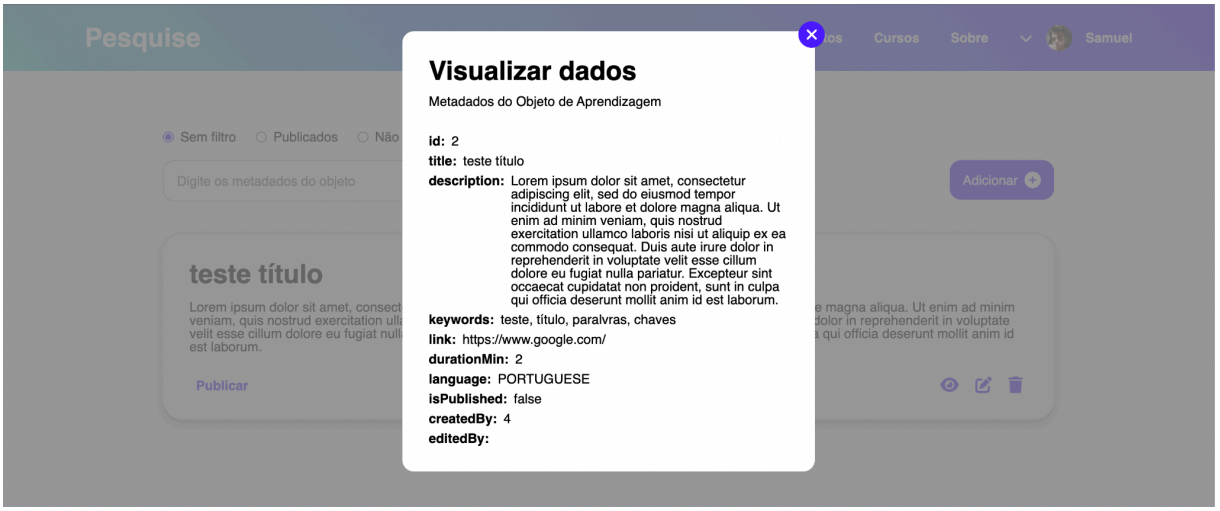

Fonte: Elaborado pelo autor (2024).

**Figura 50 - Modal de criação do objeto**



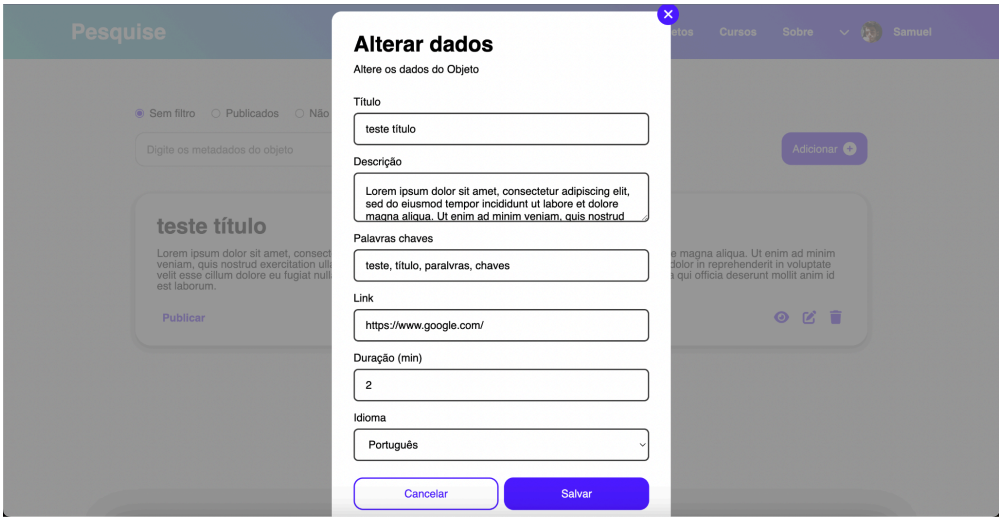

Fonte: Elaborado pelo autor (2024).

**Figura 51 - Modal de visualização de dados do objeto**



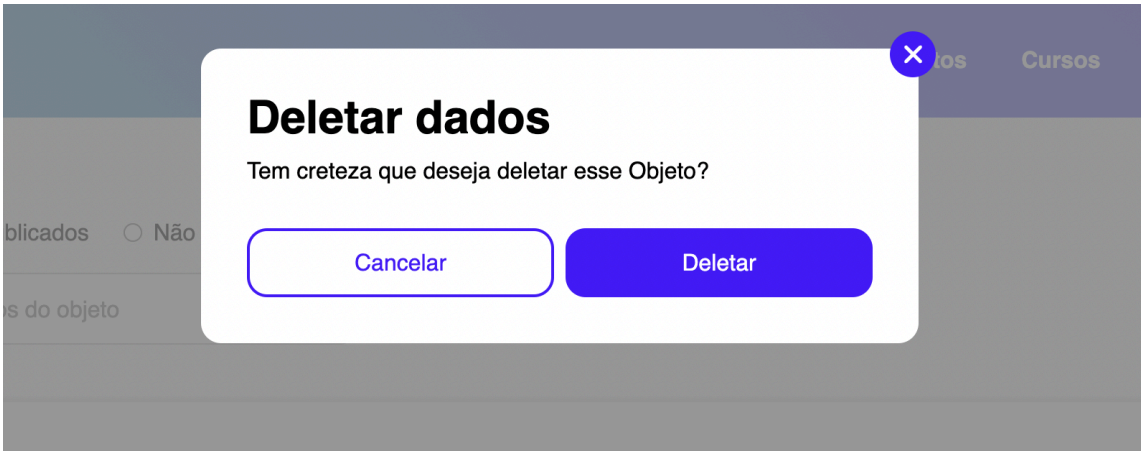

Fonte: Elaborado pelo autor (2024).

**Figura 52 - Modal de atualização do objeto**




Fonte: Elaborado pelo autor (2024).

**Figura 53 - Modal de exclusão do objeto**



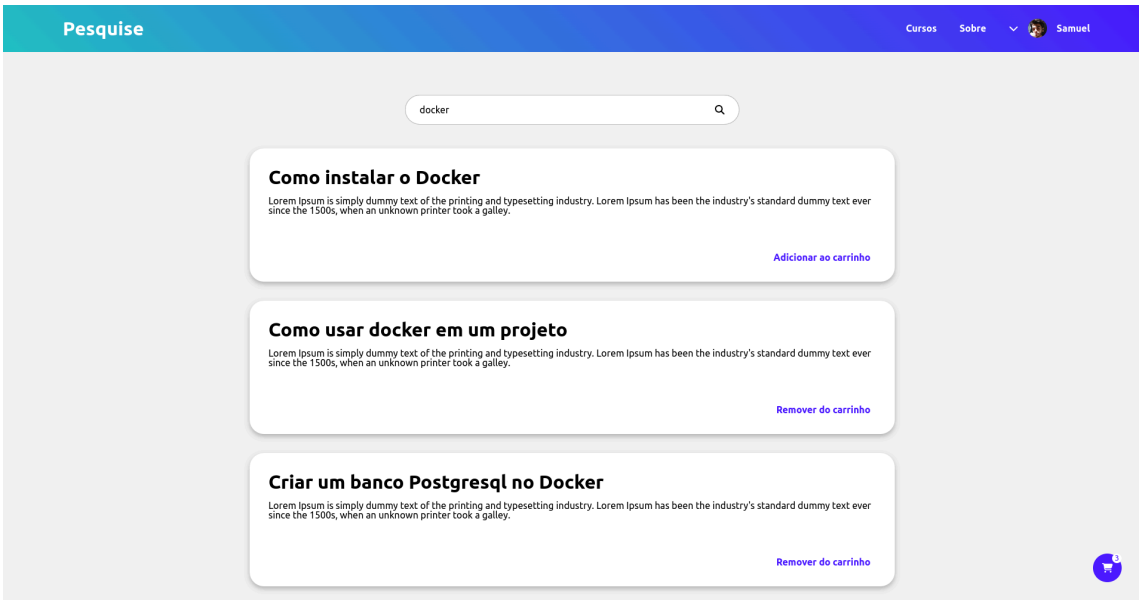

Fonte: Elaborado pelo autor (2024).

Na página inicial da aplicação, tem-se um input de busca, com algumas sugestões de temas que mudam dentro de alguns segundos. No caso da figura 54, o tema sugerido é “Banco de Dados”. E ao realizar uma busca, pode-se notar na figura 55, a maneira que o site se comporta. Neste momento, é listado os objetos que correspondem à busca. No caso da figura, foi buscado pelo tema “docker” e foram listados os objetos: “Como instalar o Docker”, “Como usar docker em um projeto” e “Criar um banco Postgresql no Docker”. Visualiza-se também ver um botão circular no canto inferior direito da tela, este botão é o carrinho.

**Figura 54 - Página inicial da aplicação**


Fonte: Elaborado pelo autor (2024).

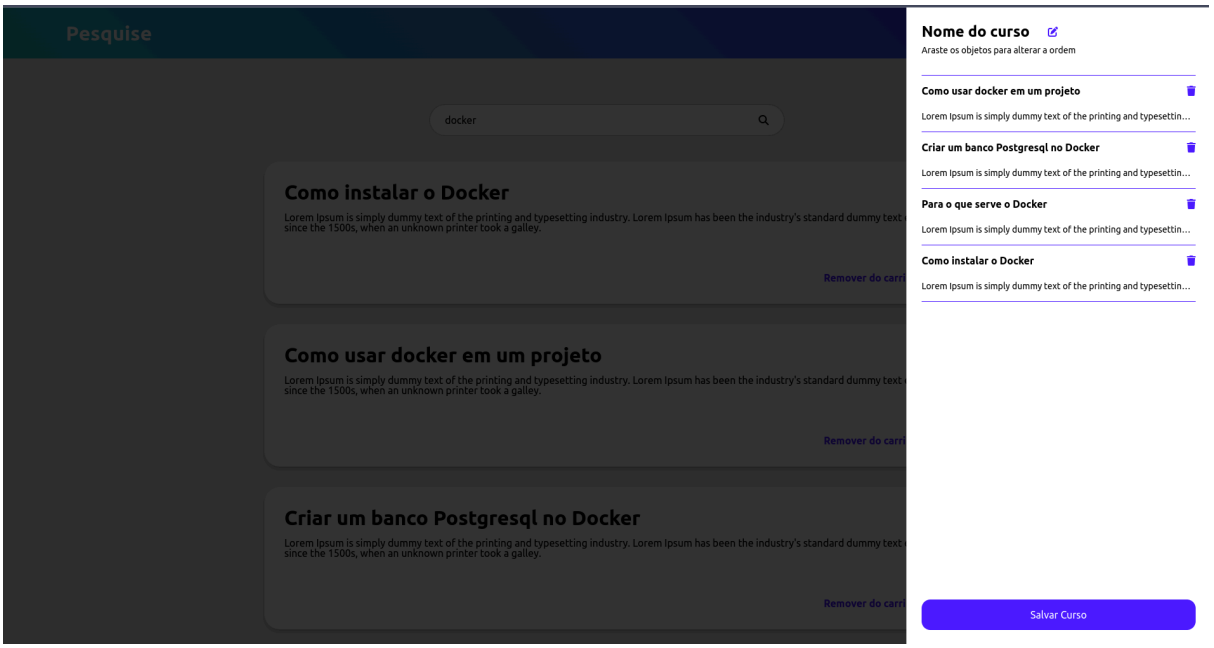
**Figura 55 - Busca principal da aplicação**




Fonte: Elaborado pelo autor (2024).

O carrinho possui algumas interações especiais, objetos podem ser adicionados ou removidos dele a partir da lista retornada por nossa busca. Além disso, é possível mover os objetos de posição dentro do carrinho, caso o usuário queira trocar a ordem dos mesmos. Também é possível definir o nome do curso, remover objetos e salvar o curso com os objetos selecionados na ordem selecionada. Pode-se visualizar o carrinho na figura 56.

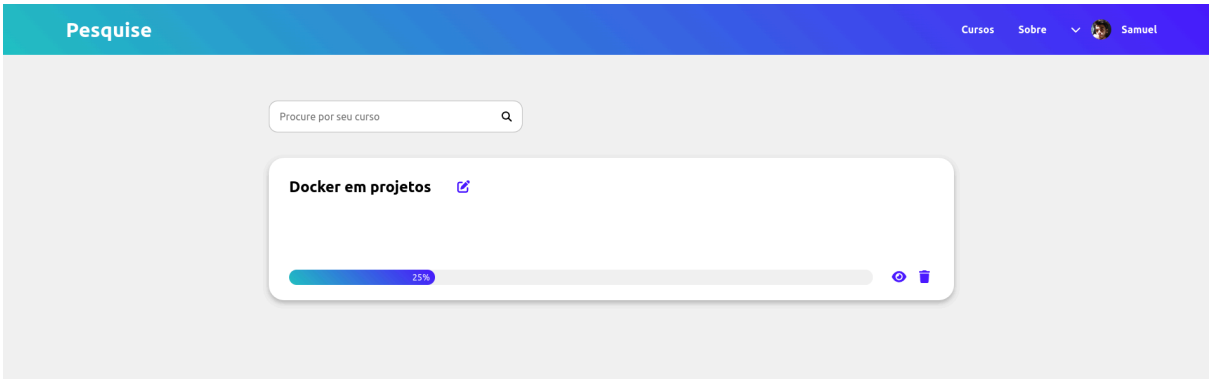
**Figura 56 - Carrinho para criação de cursos**




Fonte: Elaborado pelo autor (2024).

Após a criação de um curso, o usuário pode acessá-lo na página de cursos. Nesta página ficam listados os cursos criados pelo usuário, como é visto na figura 57. É possível visualizar o progresso do curso, editar o curso, deletar o curso e visualizar os objetos que foram relacionados a esse curso. Visualiza-se os objetos, através de um modal como mostrado na figura 58, pode-se aqui marcar e desmarcar um objeto como concluído, influenciando assim na progressão do curso do usuário. É possível clicar no nome do objeto listado pelo modal e acessar seu conteúdo.

**Figura 57 - Listagem de cursos do usuário**




Fonte: Elaborado pelo autor (2024).

**Figura 58 - Modal de visualização dos objetos dos cursos**


Fonte: Elaborado pelo autor (2024).

# 7 SUGESTÕES DE MELHORIAS E CONCLUSÕES

Neste trabalho, o objetivo principal é criar um repositório de materiais de aprendizagem baseado no modelo IEEE/LOM, utilizando tecnologias modernas como Python e Django para o backend e TypeScript e Vue.js para o backend. O sistema proposto não só facilita a busca e organização de materiais didáticos, mas também disponibiliza funcionalidades CRUD, além de permitir a criação e acompanhamento de cursos personalizados, melhorando assim a experiência educacional.

A importância deste projeto reside no seu contributo para a área da tecnologia educativa, disponibilizando uma ferramenta que melhora a acessibilidade, o planeamento e a gestão de conteúdos educativos. O impacto esperado inclui professores, alunos e instituições de ensino, proporcionando experiências de aprendizagem flexíveis e envolventes e melhorando a utilização de materiais de aprendizagem em vários contextos educativos.

É importante observar que, embora nosso sistema Vue.js forneça uma interface dinâmica e interativa, a natureza depende de JavaScript no cliente para buscar informações no servidor. Os motores de busca que possuem o JavaScript desativado durante suas buscas não conseguem indexar conteúdo dinâmico gerado por JavaScript, o que destaca a necessidade de uma estratégia de SEO específica para aplicações de página única. Embora não foram utilizadas técnicas de SEO neste projeto, é essencial que em trabalhos futuros, sejam consideradas técnicas de pré-renderização e renderização do lado do servidor. Esses métodos podem ajudar a garantir que o conteúdo criado seja acessível aos buscadores, melhorando a visibilidade e a indexação do site.

Além disso, a implementação de uma tela de perfil para os usuários é de suma importância. Atualmente o sistema permite que os dados do usuário sejam inseridos ao criar uma conta, mas não oferece a opção de atualizar essas informações. Permitir que os utilizadores editem os seus dados pessoais é importante para manter a precisão e a relevância das informações, refletindo as mudanças nas suas circunstâncias ou preferências ao longo do tempo. Adicionar essa funcionalidade pode melhorar muito a experiência do usuário no sistema, proporcionando uma experiência mais personalizada e capacitadora.

Em resumo, este projecto representa um passo no sentido de melhorar a infra-estrutura tecnológica educacional, fornecendo um sistema poderoso e flexível para a gestão de materiais de aprendizagem. A implementação e os resultados alcançados mostram o potencial do sistema em contribuir para o desenvolvimento do ensino e da aprendizagem, mostrando a

importância e capacidade de integração da tecnologia moderna no desenvolvimento de soluções educativas.

# REFERÊNCIAS


API. *In*: Glossário do MDN Web Docs: Definições de termos relacionado à Web. [Mountain View: Mozilla Foundation], 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Glossary/API. Acesso em: 04 fev. 2024.

BARRY, Douglas K. **Representational State Transfer (REST)**. Portland: Barry & Associates Incorporation, [20--]. Disponível em: https://www.service-architecture.com/articles/web-services/representational-state-transfer-rest.html. Acesso em: 23 fev. 2024.

BORGES, Luiz Eduardo. **Python para Desenvolvedores**. São Paulo: Novatec, 2014. E-book. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=&id=eZmtBAAAQBAJ&oi=fnd&pg=PA14&dq=python&ots=VETrnrHjer&sig=rBrcRk2OPFd_Kl71FjN12MSU8ow&redir_esc=y#v=onepage&q=python&f=false. Acesso em: 17 fev. 2024.

DJANGO PROJECT. **Django Documention**. [*S. l.],* Django Software Foundation, 2023. Disponível em: https://docs.djangoproject.com/en/5.0/. Acesso em: 02 mar. 2024.

DJANGO REST FRAMEWORK. **Django Rest Framework**. 2024. Disponível em: https://www.django-rest-framework.org. Acesso em: 28 jan. 2024.

DOWNEY, Allen; ELKNER, Jeffrey; MEYERS, Chris. **How to Think Like a Computer Scientist - Learning with Python**. Wellesley: Green Tea Press, 2022. E-book. Disponível em: https://www.greenteapress.com/thinkpython/thinkCSpy.pdf. Acesso em: 18 fev. 2024.

GAO, Honghao *et al*. Collaborative Learning-Based Industrial IoT API Recommendation for Software-Defined Devices: The Implicit Knowledge Discovery Perspective. **IEEE Transactions on Emerging Topics in Computational Intelligence**, Sydney, v. 6, n. 1, p. 66-76, fev./2020. DOI: 10.1109/TETCI.2020.3023155. Disponível em: https://opus.lib.uts.edu.au/bitstream/10453/143796/3/10.1109%20tetci.2020.3023155%20%20am%20Combined%20%20%20.pdf. Acesso em: 31 jan. 2024.

GAO, Zheng; BIRD, Christian; BARR, Earl T. To Type or Not to Type: Quantifying Detectable Bugs in JavaScript. *In*: IEEE/ACM 39th International Conference on Software Engineering (ICSE), 2017, Buenos Aires. **Proceedings** [...]. Buenos Aires: 2017. p. 1-12. DOI: 10.1109/ICSE.2017.75. Disponível em: https://earlbarr.com/publications/typestudy.pdf. Acesso em: 19 fev. 2024.

JAVASCRIPT. *In*: Glossário do MDN Web Docs: Definições de termos relacionado à Web. [Mountain View: Mozilla Foundation], 2024. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 24 fev. 2024.

JETBRAINS. **O Estado do Ecossistema de Desenvolvedores de 2022**. Praga: JetBrains, 2022. Disponível em: https://www.jetbrains.com/pt-br/lp/devecosystem-2022/#:~:text=As%20linguagens%20de%20programa%C3%A7%C3%A3o%20favoritas,Visual%20Basic%2C%20Delphi%20e%20C. Acesso em: 23 jan. 2024.

LOPES, Odnei Cuesta. **iRIS Sistema: Sistema servidor para recuperação de imagens por conteúdo**. 2005. Dissertação em Mestrado (Mestrado em Ciências da Computação e Matemática Computacional) - Universidade de São Paulo. São Carlos, 2005. Disponível em: https://teses.usp.br/teses/disponiveis/55/55134/tde-30102014-150509/publico/OdneiCuestaLopes.pdf. Acesso em: 14 fev. 2024.

TYPESCRIPT. **TypeScript is JavaScript with syntax for types**. Washington: Microsoft Incorporation, 2023. Disponível em: https://www.typescriptlang.org/. Acesso em: 04 mar. 2023.

VUE JS. **The Progressive JavaScript Framework**. 2023. Disponível em: https://vuejs.org/. Acesso em: 18 fev. 2024.

YUILL, Simon; HALPHIN, Harry. **Python**. State College: CiteSeerx, 2006. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1f2ee3831eebfc97bfafd514ca2abb7e2c5c86bb. Acesso em: 08 mar. 2024.

AXIOS. **Promise based HTTP client for the browser and node.js.** 2024. Disponível em: https://axios-http.com/. Acesso em: 14 jan. 2024.

JWT. J**SON Web Tokens.** 2024. Disponível em: https://jwt.io/. Acesso em: 11 jan. 2024.

BARKER, Phil. **What is IEEE Learning Object Metadata / IMS Learning Resource Metadata?** 2005. Disponível em: https://publications.cetis.org.uk/wp-content/uploads/2011/02/WhatIsIEEELOM.pdf. Acesso em: 12 jan. 2024

REACT. **Virtual DOM e Objetos Internos.** 2024. Disponível em: https://pt-br.legacy.reactjs.org/docs/getting-started.html. Acesso em: 10 jan. 2024

SILVA, Julia Marques Carvalho da. **Análise técnica e pedagógica de metadados para objetos de aprendizagem. 2011.** Disponível em: https://www.lume.ufrgs.br/handle/10183/40478. Acesso em: 11 mai. 2024

WEI, Xin et al. **Personalized online learning resource recommendation based on artificial intelligence and educational psychology.** Frontiers in psychology, v. 12, p. 767837, 2021. Disponível em: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2021.767837/full. Acesso em: 11 mai. 2024

NAZEMPOUR, Rezvan; DARABI, Houshang. **Personalized learning in virtual learning environments using students' behavior analysis**. Education Sciences, v. 13, n. 5, p. 457, 2023. Disponível em: https://www.mdpi.com/2227-7102/13/5/457. Acesso em: 11 mai. 2024

ALEXANDRE, M. dos R.; BARROS, D. M. V. **Objetos Digitais de Aprendizagem e os estilos de uso do virtual: estreitando relações e construindo diálogos**. Indagatio Didactica,

2020. Disponível em: https://doaj.org/article/2691dd87da7744abb7ccbbbf4181cad4. Acesso em: 11 mai. 2024

VIEIRA, N., et al. **A Internet nas Escolas do 1º Ciclo: Alguns Factores de Utilização Regional**. In: DIAS, P.; FREITAS, C. V. (Orgs.). Challenges 2005 – IV Conferência Internacional de Tecnologias de Informação e Comunicação na Educação. Braga: Universidade do Minho, 2005. Disponível em: https://www.nonio.uminho.pt/wp-content/uploads/2020/09/actas_challenges_2005.pdf. Acesso em: 11 mai. 2024

# AUTORIZAÇÃO

Autorizo a reprodução e/ou divulgação total ou parcial do presente trabalho, por qualquer meio convencional ou eletrônico, desde que citada a fonte.

---

Samuel José Rodrigues de Araújo Castro
samuel.castro@ufvjm.edu.br
Universidade Federal dos Vales do Jequitinhonha e Mucuri