UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DOUGLAS VINICIUS CALDAS BONIN

# IMPLEMENTAÇÃO DE UMA API PARA VISUALIZAÇÃO E INTERAÇÃO COM OBJETOS DE APRENDIZAGEM PARA FERRAMENTA DE AUTORIA FARMA

GUARAPUAVA

2023

**DOUGLAS VINICIUS CALDAS BONIN**

# IMPLEMENTAÇÃO DE UMA API PARA VISUALIZAÇÃO E INTERAÇÃO COM OBJETOS DE APRENDIZAGEM PARA FERRAMENTA DE AUTORIA FARMA

# API Implementation to Visualize and Interact with Learning Objects for the FARMA Authoring Tool

Projeto de Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 1, do Curso Superior de Tecnologia em Sistemas para Internet da Universidade Tecnológica Federal do Paraná, Campus Guarapuava, como requisito parcial para obtenção do título de Tecnólogo em Tecnologia em Sistemas para Internet.

Orientador: Prof. Dr. Diego Marczal

Coorientador: Profª. Dra. Renata Luiza Stange Carneiro Gomes

**GUARAPUAVA**

**2023**

4.0 Internacional

Esta licença permite compartilhamento, remixe, adaptação e criação a partir do trabalho, mesmo para fins comerciais, desde que sejam atribuídos créditos ao(s) autor(es). Conteúdos elaborados por terceiros, citados e referenciados nesta obra não são cobertos pela licença.

## RESUMO

A Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem (FARMA) foi concebida com o propósito de oferecer funcionalidades para a criação de Objetos de Aprendizagem no ensino da matemática. Entre 2015 e 2018 foram propostas diversas melhorias para a ferramenta, tais como obter uma melhor experiência do usuário com o desenvolvimento de novas interfaces, a remodelagem do banco de dados para incluir um sistema de controle de versão para objetos de aprendizagem, dentre outras. Essas melhorias foram agrupadas em um projeto de implementação de uma nova versão da ferramenta. Dentre essas melhorias está a funcionalidade de criação de objetos de aprendizagem. Esta funcionalidade esta dividia em duas principais atividades: 1) Implementação do *front-end* e *back-end* para a criação dos objetos de aprendizagem; 2) Implementação do *front-end* e *back-end* para a visualização e interação dos objetos de aprendizagem; Com base nisso, o desenvolvimento deste trabalho tem como objetivo a implementação de recursos *back-end* que forneça os dados e receba as requisições necessárias para a visualização e interação de objetos de aprendizagem. Para isso, será utilizado o framework Ruby On Rails para implementação da API, a especificação OpenAPI para a documentação da API, o GitLab Flow para controle do fluxo de trabalho objetivando a Integração/Entrega Contínua. Com a finalização deste projeto espera-se que parte da funcionalidade visualização e interação dos objetos de aprendizagem seja incorporada a nova versão da FARMA. Esta funcionalidade inclui uma pré-visualização de um objeto de aprendizagem, por parte do seu criador, a interação por parte de um estudante e a interação por parte de um usuário visitante. O desenvolvimento desses recursos dará sequencia ao desenvolvimento da nova versão da FARMA, permitindo a implementação do *front-end* em uma próxima contribuição.



**Palavras-chave:** objeto; aprendizagem; ferramenta; sistema.

**ABSTRACT**

The Authoring Tool for Error Remediation with Mobility in Learning (FARMA) was conceived with the aim of offering functionalities for the creation of Learning Objects in mathematics teaching. Between 2015 and 2018, several improvements were proposed for the tool, such as obtaining a better user experience with the development of new interfaces, remodeling the database to include a version control system for learning objects, among others. These improvements were grouped together in a project to implement a new version of the tool. Among these improvements is the functionality for creating learning objects. This functionality was divided into two main activities: 1) Implementation of the front-end and back-end for the creation of learning objects; 2) Implementation of the front-end and back-end for the visualization and interaction of learning objects; Based on this, the development of this work aims to implement back-end resources that provide the data and receive the requests necessary for the visualization and interaction of learning objects. To this end, the Ruby On Rails framework will be used to implement the API, the OpenAPI specification will be used to document the API, and GitLab Flow will be used to control the workflow with the aim of Continuous Integration/Delivery. With the completion of this project, it is hoped that part of the functionality for visualization and interacting with learning objects will be incorporated into the new version of FARMA. This functionality includes a preview of a learning object by its creator, interaction by a student and interaction by a visiting user. The development of these features will follow on from the development of the new version of FARMA, allowing the front-end to be implemented in a future contribution.

**Keywords:** object; learning; tool; system.

## LISTA DE FIGURAS


**Figura 1 – Página principal da Farma Reborn . . . . . 11**

**Figura 2 – Introdução do objeto de aprendizagem . . . . . 12**

**Figura 3 – Enunciado do objeto de aprendizagem . . . . . 12**

**Figura 4 – Passos de solução no exercício do objeto de aprendizagem . . . . . 13**

**Figura 5 – Orientações e dicas do objeto de aprendizagem . . . . . 14**

**Figura 6 – Histórico no objeto de aprendizagem . . . . . 14**

**Figura 7 – Percentuais no objeto de aprendizagem . . . . . 15**

**Figura 8 – Visualização do objeto de aprendizagem na aba turma - Aluno . . . . . 15**

**Figura 9 – Objeto de aprendizagem - Aluno . . . . . 16**

**Figura 10 – Gitlab flow . . . . . 22**

**Figura 11 – Esteira CI/CD . . . . . 23**

**Figura 12 – Modelo do banco de dados para visualização e interação de objetos de aprendizagem na FARMA . . . . . 27**

## LISTA DE ABREVIATURAS E SIGLAS

### Siglas

| | |
|---|---|
| API | Interface de Programação de Aplicativos, do inglês *Application Programming Interface* |
| CD | Entrega Contínua, do inglês *Continuous Delivery* |
| CI | Integração Contínua, do inglês *Continuous Integration* |
| FARMA | Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem |
| GRAPHQL | Linguagem de Consulta de Dados em Grafos, do inglês *Graph Query Language* |
| gRPC | Chamada de Procedimento Remoto do Google, do inglês *Google Remote Procedure Call* |
| HTTP | Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol* |
| JSON | Notação de Objetos Javascript, do inglês *Javascript Object Notation* |
| PR | Pedido de Mesclagem, do inglês *Pull Request* |
| REST | Transferência de Estado Representacional, do inglês *Representational State Transfer* |
| SSH | Shell Seguro, do inglês *Secure Shell* |
| URL | Localizador Uniforme de Recursos, do inglês *Uniform Resource Locator* |
| VPS | Servidor Virtual Privado, do inglês *Virtual Private Server* |

**SUMÁRIO**


**1 INTRODUÇÃO . . . . . 7**

**1.1 Contextualização . . . . . 7**

**1.2 Objetivos . . . . . 9**

1.2.1 Objetivo geral . . . . . 9

1.2.2 Objetivos específicos . . . . . 9

**1.3 Justificativa . . . . . 10**

**1.4 Organização . . . . . 10**

**2 FARMA . . . . . 11**

**2.1 Visualização do objeto de aprendizagem . . . . . 12**

**3 API - INTERFACE DE PROGRAMAÇÃO DE APLICAÇÕES . . . . . 17**

**3.1 Padrões arquiteturais . . . . . 17**

3.1.1 REST E RESTFULL . . . . . 17

3.1.2 GRAPHQL . . . . . 18

3.1.3 gRPC . . . . . 18

**4 MATERIAIS E MÉTODOS . . . . . 20**

**4.1 Materiais . . . . . 20**

**4.2 Métodos . . . . . 20**

4.2.1 Levantamento e especificação de requisitos . . . . . 21

4.2.2 Projeto e implementação de software . . . . . 21

4.2.3 Verificação e validação do software . . . . . 22

4.2.4 Entrega e Implantação . . . . . 23

**5 RESULTADOS PARCIAIS . . . . . 25**

**5.1 Requisitos . . . . . 25**

5.1.1 Requisitos funcionais . . . . . 25

5.1.2 Requisitos não funcionais . . . . . 26

**5.2 Banco de dados . . . . . 26**

**5.3 Rotas da API . . . . . 28**

5.3.1 Rota de detalhe do objeto de aprendizagem . . . . . 28

5.3.2 Rota de detalhe de página de um objeto de aprendizagem específico . . . . 29

5.3.3 Rota de solicitação de dicas . . . . . 29

5.3.4 Rota de confirmação de visualização de passo de solução . . . . . 30
5.3.5 Rota de envio de resposta para passo de solução . . . . . 30
**5.4 Backlog . . . . . 30**
5.4.1 Configuração de modelos necessários . . . . . 30
5.4.2 Criar rota de detalhe de objeto de aprendizagem para aluno e professor . . . 31
5.4.3 Criar rota de detalhe de página de um objeto de aprendizagem específico . . 31
5.4.4 Criar rota de solicitação de dicas . . . . . 32
5.4.5 Criar rota de confirmação de visualização . . . . . 32
5.4.6 Criar a rota de envio de resposta para passo de solução . . . . . 32
5.4.7 Criar rotas públicas para usuário visitante . . . . . 32
**6 CONSIDERAÇÕES FINAIS . . . . . 33**
**REFERÊNCIAS . . . . . 34**

# 1 INTRODUÇÃO

Neste capítulo será apresentado uma visão geral do projeto, mostrando o cenário onde ele se aplica, bem como a motivação e justificativas do seu desenvolvimento. Também, os objetivos que devem ser cumpridos para que o trabalho seja concluído com êxito.

## 1.1 Contextualização

A educação é um procedimento no qual ocorre a transferência de conhecimento, valores, aptidões e elementos culturais de indivíduos para indivíduos. Isso acontece em uma variedade de cenários, que incluem o ambiente familiar, instituições formais e até mesmo por meio de aprendizado autodirigido. Com o avanço tecnológico, a maneira como a educação é concebida e entregue tem passado por uma transformação significativa. A integração da tecnologia no campo educacional tem proporcionado oportunidades únicas para aprimorar as estratégias pedagógicas e envolver os alunos de maneira mais proveitosa (SOUZA, 2018).

No ensino, os professores frequentemente empregam uma diversidade de recursos, com o intuito de melhorar a experiência de aprendizado dos alunos. Esses recursos podem incluir elementos como demonstrações práticas, materiais visuais, interações em grupo e o uso da tecnologia educacional, além de ferramentas como quadros interativos e aplicações didáticas. Esse cenário demonstra como os educadores fazem uso de diferentes abordagens e recursos, adaptando-se às demandas e preferências de aprendizado dos estudantes. Ao fazer isso, eles conseguem criar ambientes educacionais mais dinâmicos e cativantes, oferecendo aos alunos oportunidades adicionais para compreender e aplicar o conhecimento de maneira eficaz (LIMA; ARAúJO, 2021).

Nesse cenário, um importante recurso utilizado para melhorar as estratégias de ensino são os objetos de aprendizado, ou aprendizagem. Os objetos de aprendizagem são quaisquer recursos (digitais ou não) que podem contribuir no processo de aprendizagem para instruir uma variedade de tópicos e auxiliar na revisão de conceitos. A metodologia de utilização desses objetos de aprendizagem desempenha um papel indispensável na promoção do pensamento crítico dos alunos. Esse recurso se destaca pela flexibilidade e pela capacidade de reutilização, o que facilita a disseminação do conhecimento e a sua atualização constante (AGUIAR; FLÔRES, 2014).

De acordo com Aguiar e Flôres (2014) os objetos de aprendizagem podem se apresentar em diferentes formatos, desde simples apresentações de slides até simulações complexas, muitas vezes organizados em módulos reutilizáveis para atender às necessidades de diferentes contextos. Para isso, os professores podem contar com o auxílio de sistemas educacionais para explorar diversas formas de recursos digitais. Por exemplo, para uso do ensino da matemática, o Mathway[1] oferece recursos para resolver problemas e equações em áreas como álgebra,

[1] https://www.mathway.com

trigonometria, cálculo e estatística. Nesta ferramenta, os objetos de aprendizagem são representados na forma de exercícios interativos. Já na área de ciências, o PhET[2] pode ser utilizado por educadores e estudantes para entender e experimentar conceitos científicos relacionadas à física, química, biologia, matemática e outras disciplinas científicas de maneira interativa. Tanto no Mathway quanto no PhET as simulações podem ser acessadas por diferentes dispositivos, tais como dispositivos móveis e computadores.

Neste contexto, este trabalho enfatiza a Ferramenta de Autoria para Remediação de Erros com Mobilidade na Aprendizagem (FARMA). Esta ferramenta foi desenvolvida como parte do trabalho publicado em Marczal *et al.* (2014). O propósito geral da ferramenta é dispor de um ambiente, onde cada professor pode criar atividades (Autoria) para testar, avaliar e corrigir o processo de resolução adotado pelos alunos (Remediação de Erros). Trata-se de um sistema web, desta forma podendo ser acessado por diferentes dispositivos, desde que tenham acesso a Internet, conferindo tanto aos alunos quanto aos professores a flexibilidade de utilização (Mobilidade na Aprendizagem).

Na FARMA o objeto de aprendizagem é apresentado na forma de atividades que compreendem uma explicação do conteúdo, seguida por exercícios relacionados ao tema. Isso simplifica o processo de estruturação por parte do professor e proporciona ao aluno a oportunidade de avaliar sua compreensão do assunto abordado. Imediatamente após a conclusão dos exercícios, o aluno pode revisar seus resultados, identificando e corrigindo os erros cometidos com a assistência do sistema de auditoria. Além disso, tanto o aluno quanto o professor têm a capacidade de revisitar as submissões dos exercícios em seu caminho para a resposta final, o que facilita a compreensão das dificuldades enfrentadas durante o estudo do conteúdo (MARCZAL *et al.*, 2014).

Atualmente, existem duas versões da ferramenta, uma legada e outra em desenvolvimento. A concepção da nova versão foi impulsionada pela busca por uma melhor experiência do usuário, sendo esta uma versão aprimorada e intuitiva. Além disso, esta nova versão tem o objetivo de modernizar e otimizar a arquitetura da ferramenta, fazendo uso de tecnologias mais recentes, para facilitar futuras manutenções.

O desenvolvimento deste trabalho foi motivado para dar sequência ao desenvolvimento da nova versão da FARMA, na qual serão disponibilizados recursos para visualização de objetos de aprendizagem nas áreas do professor, aluno e visitante. Este projeto de trabalho de conclusão de curso descreve o desenvolvimento de recursos *back-end*[3] para visualização e interação desses objetos de aprendizagem. Esses recursos permitem que os dados relacionados aos objetos de aprendizagem possam ser integrados à aplicação *front-end*[4], desta forma

[2] https://phet.colorado.edu

[3] Back-end se relaciona com o que está por trás das aplicações desenvolvidas na programação (TOTVS, 2020).

[4] O front-end está relacionado com a interface gráfica do projeto. Ou seja, é onde se desenvolve a aplicação com a qual o usuário irá interagir diretamente, seja em softwares, sites, aplicativos, entre outros (TOTVS, 2021).

sendo possível observar os objetos de aprendizagem de forma interativa durante a resolução dos exercícios.

## 1.2 Objetivos

Nesta seção, é apresentado os objetivos gerais e específicos que orientam a execução deste projeto. Cada um desses objetivos foi definido para guiar a realização deste projeto prático, demonstrando como se aplica o conhecimento em situações reais.

### 1.2.1 Objetivo geral

Neste trabalho, é proposto o desenvolvimento de uma Interface de Programação de Aplicativos, do inglês *Application Programming Interface* (API) que forneça os dados e receba as requisições necessárias para a visualização e interação de objetos de aprendizagem para a FARMA. Os dados e requisições devem suprir a pré-visualização de um objeto de aprendizagem, por parte do seu criador, a interação por parte de um estudante e a interação por parte de um usuário visitante.

### 1.2.2 Objetivos específicos

A seguir, são descritos os objetivos específicos:

- Analisar e projetar recursos necessários para visualização do objeto de aprendizagem.
- Analisar e projetar recursos necessários para interação com um objeto de aprendizagem.
- Implementar recursos para pré-visualização e interação com objeto de aprendizagem para professor (criador de objetos de aprendizagem).
- Implementar recursos para visualização e interação com objeto de aprendizagem para aluno.
- Implementar recursos para visualização e interação com objeto de aprendizagem para usuário visitante.
- Implementar recursos para registrar a interação do aluno com o objeto de aprendizagem, a saber:
    a) acesso aos recursos do objeto de aprendizagem;
    b) acerto;

c) error;

d) registro do contexto de erro e acerto.

- Implementar recursos para o aluno poder acessar os detalhes de sua interação com o objeto de aprendizagem.

## 1.3 Justificativa

Considerando o desenvolvimento da nova versão da FARMA, compreendendo uma arquitetura dividida em *back-end* e *front-end*, bem como a funcionalidade para a criação de objetos de aprendizagem, apresentou-se a necessidade de implementar uma API dedicada para a visualização e gerenciamento desses objetos. Portanto, este trabalho propõe desenvolver uma API cujos recursos devem permitir que a aplicação *front-end* utilize as informações oferecidas por esses recursos e apresente os objetos de aprendizagem aos usuários, incluindo professores e alunos. Para isso, será aplicado o conhecimento adquirido ao longo do curso, com ênfase no desenvolvimento *back-end*, especialmente na criação de API's.

## 1.4 Organização

O restante deste trabalho está organizado da seguinte maneira: no Capítulo 2 apresentamos a FARMA para contextualizar o desenvolvimento dos capítulos seguintes. No Capítulo 3 são apresentado alguns tipos de arquiteturas de API. No Capítulo 4 é delineado os materiais e métodos utilizados para o desenvolvimento do trabalho. No Capítulo 5 são apresentados alguns resultado já obtidos. No Capítulo 6 são apresentadas as considerações finais do projeto de trabalho de conclusão de curso.

## 2 FARMA

A FARMA [1] foi desenvolvida como parte de doutorado de Diego Marczal (MARCZAL *et al.*, 2014). O propósito principal dessa ferramenta é oferecer um ambiente para a aprendizagem de conceitos matemáticos. Com esse propósito, a ferramenta incorpora objetos de aprendizagem permitindo uma abordagem flexível para a exploração de tópicos matemáticos. Isso possibilita que monitores e professores de matemática adaptem os conceitos de acordo com as necessidades específicas de cada aluno.

Entre o ano de 2015 e 2018, realizou-se o projeto de "Redesign e Refatoração da FARMA", com a participação de estudantes do curso de Graduação em Sistemas para Internet. Este projeto visou aprimorar a ferramenta, otimizando suas estruturas internas e interfaces que foram projetadas no trabalho de conclusão de curso da acadêmica Erika Leão dos Santos, publicado em Santos (2018). Como resultado desse trabalho, foi proposta a segunda versão da ferramenta, atualmente em fase de desenvolvimento. A Figura 1 apresenta a página principal desta versão.




**Figura 1 – Página principal da Farma Reborn**
**Fonte: http://farma-reborn.educacional.mat.br.**

No trabalho de conclusão de concurso de Correia (2017), foi apresentada uma remodelagem do banco de dados da FARMA. Essa nova modelagem introduziu um sistema de controle de versão para objetos de aprendizagem, permitindo o armazenamento dos contextos de interação do aluno com esses objetos. Com base nesses contextos, a modelagem assegurou a disponibilidade dos dados corretos para gerar estatísticas de desempenho do aprendiz, tanto dentro de um objeto de aprendizagem específico quanto em todo o contexto de uma turma.

[1] Disponível em: http://farma.educacional.mat.br

A fim de adequar várias contribuições anteriores em uma versão estável da FARMA, foi apresentado o projeto descrito em Vitek (2023) que objetivou iniciar o desenvolvimento da nova versão da ferramenta com a criação de objetos de aprendizagem na área do professor. A partir disso, o presente trabalho descreve parte do projeto que dá sequência ao desenvolvimento dessa nova versão, a partir da implementação da API para visualização dos objetos de aprendizagem.

## 2.1 Visualização do objeto de aprendizagem

Na FARMA, os Objetos de Aprendizagem consistem em elementos que incluem:

a) Uma introdução que fornece o contexto sobre o assunto do objeto de aprendizagem, conforme mostrado na Figura 2.

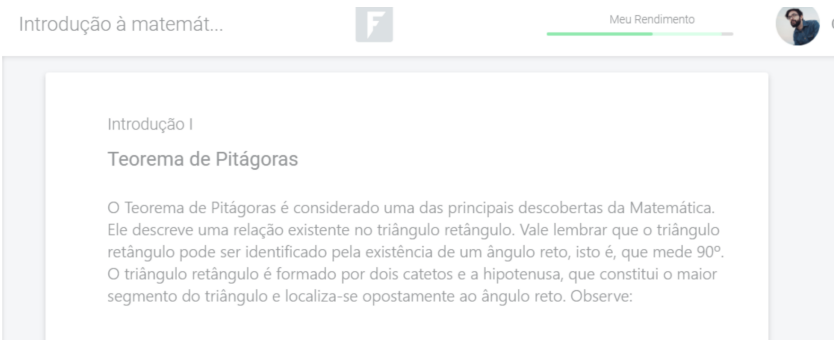



**Figura 2 – Introdução do objeto de aprendizagem**
**Fonte: http://farma-reborn.educacional.mat.br.**

b) Um enunciado que pode conter elementos como imagens, links para vídeos e gráficos, conforme a Figura 3.

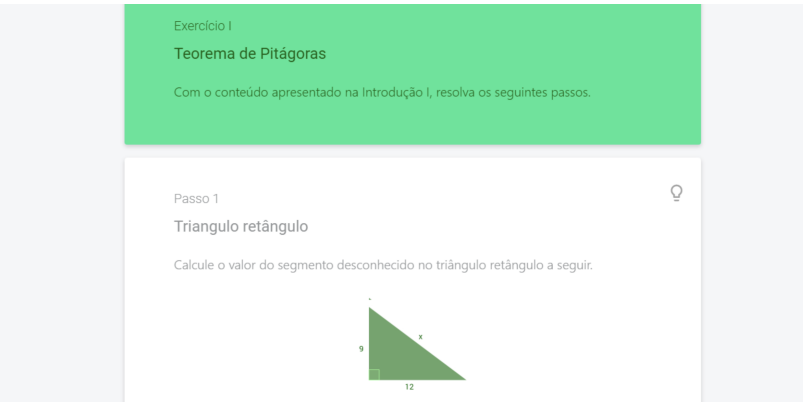



**Figura 3 – Enunciado do objeto de aprendizagem**
**Fonte: http://farma-reborn.educacional.mat.br.**

c) Os passos de solução, nos quais cada passo é explorado uma parte do conteúdo proposto, onde é possível realizar o envio da resposta. Conforme Figura 4, ao visualizar o passo de solução é apresentado um teclado para preencher com a resposta e também sobre ele um circulo vermelho contendo o número de tentativas de respostas enviadas.

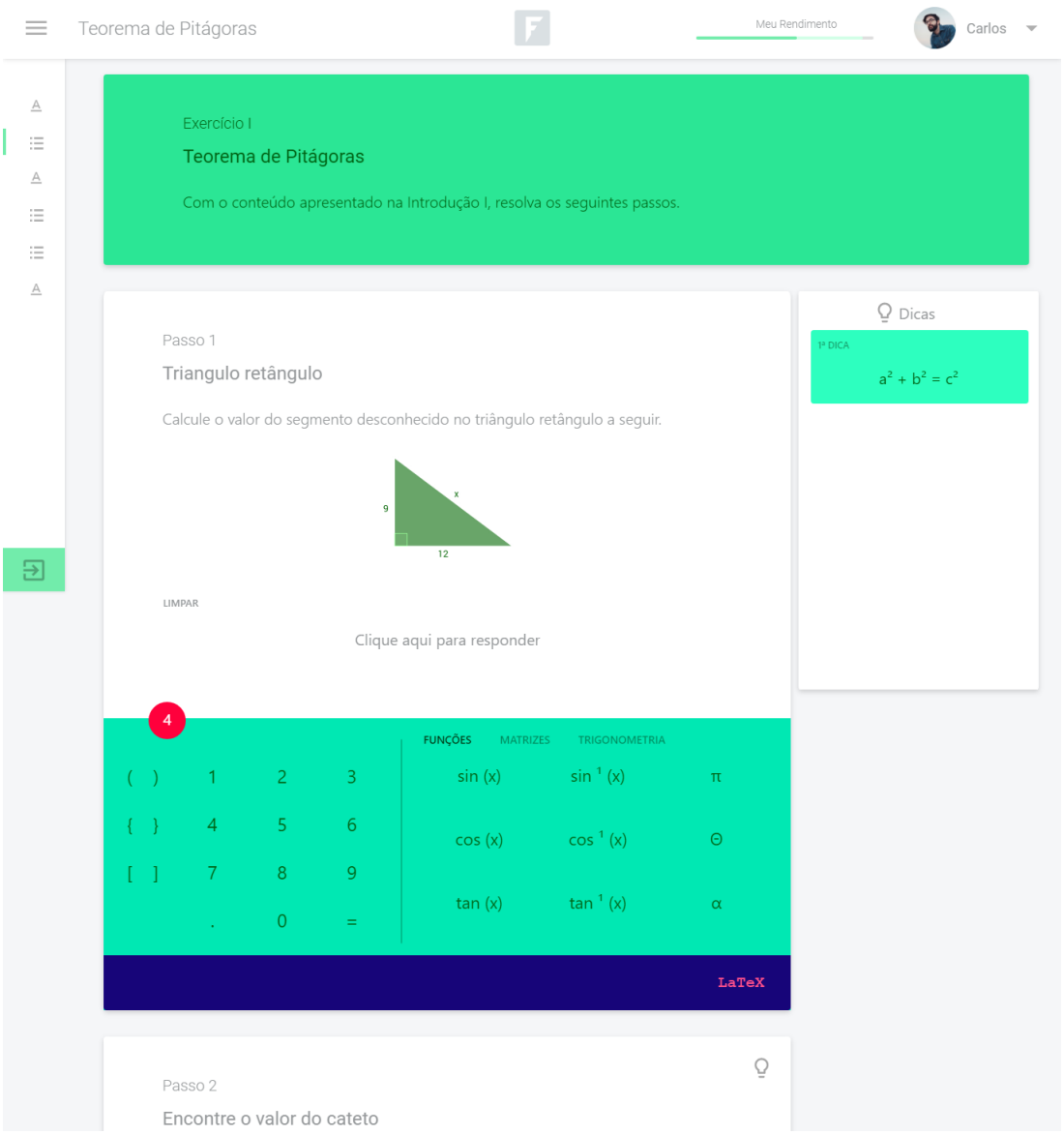



**Figura 4 – Passos de solução no exercício do objeto de aprendizagem**
**Fonte: http://farma-reborn.educacional.mat.br.**

d) Orientações e dicas relacionadas ao assunto em caso de respostas incorretas, conforme a Figura 5. Isso permite que os alunos recebam apoio adicional dos professores por meio de textos explicativos e/ou vídeos, para ajudá-los a compreender o conteúdo e resolver o problema.




**Figura 5 – Orientações e dicas do objeto de aprendizagem**
**Fonte: http://farma-reborn.educacional.mat.br.**

e) Um histórico que registra todas as submissões feitas até que a resposta correta seja alcançada. Isso permite que os alunos analisem seu progresso e aprendizado ao longo do processo, seja no objeto de aprendizagem como um todo ou em um determinado exercício. Conforme mostrado na Figura 6, o histórico do aluno é apresentado, em um exercício onde cada item é um passo de solução. Cada item é representado por uma cor, onde cada uma significa: Cinza (Não acessou), Amarelo (Apenas visualizou), Vermelho (Errou) e Verde (Acertou).

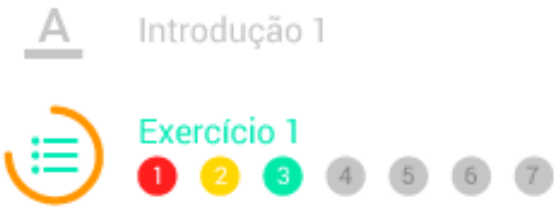



**Figura 6 – Histórico no objeto de aprendizagem**
**Fonte: http://farma-reborn.educacional.mat.br.**

No menu superior também é possível visualizar esse histórico de forma mais genérica, contemplando todo o objeto de aprendizagem e não somente um determinado exercício. Esta visualização é mostrada na forma de percentuais, conforme a Figura 7. Esses percentuais são divididos em: percentual concluído, percentual explorado e percentual inexplorado.

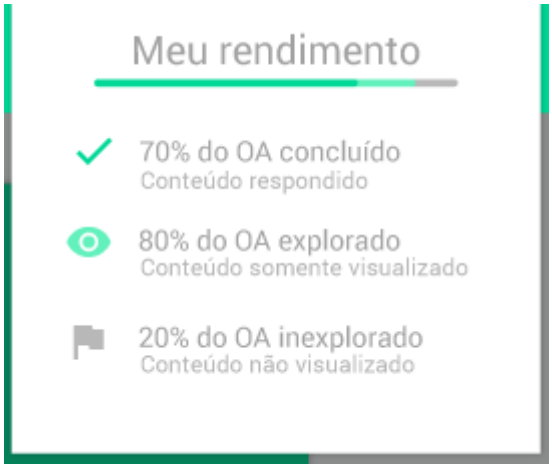



**Figura 7 – Percentuais no objeto de aprendizagem**
**Fonte: http://farma-reborn.educacional.mat.br.**

Os objetos de aprendizagem podem ser visualizados pelos seguintes tipos de usuários: professor, aluno e visitante. O professor é capaz de pré-visualizar o objeto de aprendizagem como um todo ou parte dele. Na visualização do objeto de aprendizagem para o aluno (aba Turmas) são apresentadas apenas as informações de título, descrição e progresso no objeto de aprendizagem. A Figura 8 mostra um exemplo de visualização de um objeto de aprendizagem por um aluno.

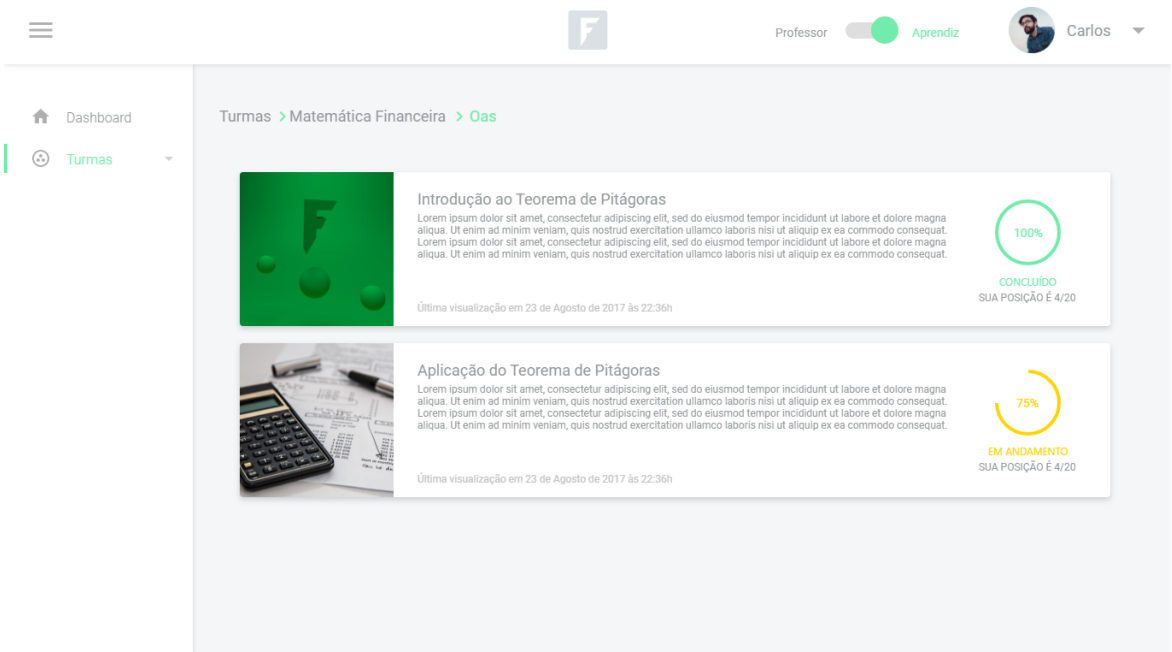



**Figura 8 – Visualização do objeto de aprendizagem na aba turma - Aluno**
**Fonte: http://farma-reborn.educacional.mat.br.**

Por outro lado, durante a resolução do objeto de aprendizagem pelo o aluno é possível visualizar a introdução do objeto de aprendizagem, descrição, algumas imagens, vídeos ou links

sobre o assunto, o rendimento no objeto de aprendizagem e o histórico de tentativas durante a resolução dos exercícios. A Figura 9 mostra um exemplo desta visualização.




**Figura 9 – Objeto de aprendizagem - Aluno**
**Fonte: http://farma-reborn.educacional.mat.br.**

Dessa mesma maneira que foi apresentada nas Figuras 8 e fig:oa-aluno, o professor (com algumas pequenas adaptações) e o vistante conseguem visualizar o objeto de aprendizagem. Porém, o professor pode visualizar apenas o objeto de aprendizagem que criou ou está criando. Já o usuário visitante interage e visualiza da mesma forma que o aluno, mas sem as informações serem armazenadas para relatório. Assim, diferentemente do aluno e do visitante, o professor não tem suas interações com o objeto registradas.

# 3 API - INTERFACE DE PROGRAMAÇÃO DE APLICAÇÕES

API é um conceito utilizado para descrever uma interface de comunicação disponibilizada por um sistema, permitindo que terceiros acessem suas funcionalidades, informações e recursos, sem necessidade de conhecimento sobre a forma como esses elementos são executados internamente. Ela consiste em um conjunto de procedimentos e normas amplamente empregados na Internet, com o propósito de simplificar a integração entre diversos sistemas e aplicações (MELO, 2023).

## 3.1 Padrões arquiteturais

Para o desenvolvimento de uma API existem alguns padrões de arquitetura, dentro os principais utilizados estão: Transferência de Estado Representacional, do inglês *Representational State Transfer* (REST), Linguagem de Consulta de Dados em Grafos, do inglês *Graph Query Language* (GRAPHQL) e Chamada de Procedimento Remoto do Google, do inglês *Google Remote Procedure Call* (gRPC). A seguir, estes padrões serão brevemente descritos.

### 3.1.1 REST E RESTFULL

REST define um conjunto de operações a partir dos métodos Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol* (HTTP) POST, GET, PUT, PATCH, DELETE, entre outros, que os utilizadores podem empregar para recuperar informações do servidor. A comunicação entre clientes e servidores ocorre por meio do protocolo HTTP, sendo o tipo de API de maior destaque e versatilidade disponíveis na internet atualmente (AWS, 2023a).

A característica primordial da REST API reside na sua abordagem sem estado. A falta de estado significa que os servidores não retêm informações dos clientes entre as requisições. As solicitações dos clientes aos servidores assemelham-se aos URLs que são inseridas no navegador para acessar um site. A resposta do servidor consiste em dados simples, sem a típica formatação gráfica encontrada em uma página da web AWS (2023a). Assim se baseando em 4 princípios:

1. Uso dos métodos HTTP, como GET, POST, PUT e DELETE, para realizar as operações a partir dos recursos.

2. Uso de Localizador Uniforme de Recursos, do inglês *Uniform Resource Locator* (URL)'s para identificar recursos específicos.

3. Transferência de dados entre cliente e servidor em um formato padrão, geralmente utilizado Notação de Objetos Javascript, do inglês *Javascript Object Notation* (JSON) por ser um formato independente entre linguagens de programação.

4. Preservação do estado da aplicação no lado do cliente, em contraposição ao armazenamento no servidor.

A RESTFULL API segue os princípios do REST, aderindo às características mencionadas anteriormente e adicionando uma implementação mais abrangente. Para uma API ser reconhecida como RESTFULL, ela deve satisfazer certos critérios adicionais além dos princípios básicos do REST (ANDRADE, 2023). Sendo eles:

1. Interface uniforme: A API deve oferecer uma interface consistente e padronizada para acessar e manipular recursos.

2. Clientes sem estado: Cada solicitação enviada pelo cliente ao servidor deve conter todas as informações necessárias para sua compreensão, sem depender de qualquer contexto armazenado no servidor.

3. Operações baseadas em recursos: As ações executadas pela API devem estar centradas em recursos identificados por URLs exclusivas.

Assim a principal distinção entre REST e RESTFULL reside no grau de conformidade com os princípios do REST. Sendo RESTFULL uma implementação mais completa e rigorosa desses princípios, incorporando critérios adicionais.

### 3.1.2 GRAPHQL

O GRAPHQL foi lançado pelo Facebook em 2012 e se tornou código aberto em 2015. Por ser uma nova forma de solicitar dados para um servidor, é uma linguagem de consulta, especificação e coleção de ferramentas desenvolvida para operar sobre um único endpoint[1] por meio do HTTP, que otimiza o desempenho e a flexibilidade. Tendo foco no cliente, sua estrutura permite que as aplicações clientes decidam quais informações precisam buscar e quanta informação querem receber do servidor. Diferentemente do REST, onde a estrutura é toda definida no servidor, e portanto, o servidor é quem controla a arquitetura (VIDANYA, 2021).

### 3.1.3 gRPC

O gRPC é uma tecnologia que permite que clientes e servidores se comuniquem em diferentes ambientes e linguagens. Os servidores definem serviços com métodos, especificando parâmetros e tipos de retorno. Ele usa buffers de protocolo[2] como a linguagem de definição de interface (IDL) por padrão, mas oferece a flexibilidade de usar outras alternativas, se necessário. Além disso, as APIs mais recentes do Google também disponibilizam versões gRPC de suas

[1] lugar onde esses pedidos (referidos como chamadas de API) são respondidos (CLOUDFLARE, 2023).
[2] https://protobuf.dev/overview/

interfaces para facilitar a incorporação das funcionalidades do Google em aplicações (GRPC, 2022). gRPC foi criado com foco na capacidade de desenvolver APIs de alto desempenho destinadas a ambientes de microsserviços distribuídos em data centers. Essa tecnologia é particularmente vantajosa para sistemas internos que necessitam de transmissão em tempo real e precisam lidar com grandes volumes de dados e em cenários de arquiteturas de microsserviços que envolvem diversas linguagens de programação (AWS, 2023b).

# 4 MATERIAIS E MÉTODOS

Neste capítulo, serão detalhados os materiais utilizados na elaboração deste projeto, juntamente com os métodos escolhidos para atingir os resultados planejados.

## 4.1 Materiais

As ferramentas e tecnologias adotadas para o desenvolvimento do trabalho de conclusão de curso são elencadas a seguir:

- Gestão de projeto: Para a gestão do projeto será utilizado o quadro Kanban no *github projects*[1] que permitirá um melhor acompanhamento do andamento das atividades a serem desenvolvidas.
- Framework de desenvolvimento: Para o desenvolvimento da API será utilizado Ruby on Rails[2], um dos frameworks mais renomados e consolidados do mercado (FINCHER; DESMOND, 2023). Essa tecnologia permitirá não apenas desenvolver uma solução sólida e eficiente, mas também aproveitar o suporte integrado para testes automatizados oferecido pelo framework. A implementação de testes automatizados terá um papel crucial na garantia da qualidade do projeto, assegurando que a API funcione de maneira confiável e sem falhas.
- Especificação da API: Para a composição da documentação da API será utilizada a especificação da OpenAPI[3], que é uma linguagem de especificação para HTTP API'S. Essa documentação permitirá que os recursos da API sejam descritos a partir de um padrão utilizado no mercado.
- Sistema Gerenciador de Banco de Dados: Para gerenciar o banco de dados será utilizado PostgreSQL[4]. Essa escolha foi feita baseada no trabalho desenvolvido por Correia (2017), projeta uma nova modelagem do banco de dados.

## 4.2 Métodos

A sequência das atividades que serão desenvolvidas de forma incremental para a realização deste trabalho são descritas a seguir:

[1] https://docs.github.com/pt/issues/organizing-your-work-with-project-boards/managing-project-boards/about-project-boards
[2] https://rubyonrails.org/
[3] https://www.openapis.org/
[4] https://www.postgresql.org/

### 4.2.1 Levantamento e especificação de requisitos

É fundamental compreender como a FARMA atende às necessidades dos seus usuários. A partir desse entendimento, é possível estabelecer uma definição mais precisa dos requisitos do sistema, para a compreensão das necessidades atendidas pela FARMA. Feito isso, os requisitos serão organizados com o uso do quadro KANBAN, onde cada coluna representará uma etapa de desenvolvimento. Os requisitos estabelecidos serão integrados ao Backlog[5] e, a medida que forem progressivamente trabalhados, seguirão o fluxo do quadro Kanban (*Planning*, *Ready*, *In Progress*, *In Review* e *Done*). Em etapas posteriores, esses requisitos serão divididos em subtarefas de implementação, ou seja, em funcionalidades menores e autônomas a serem desenvolvidas.

### 4.2.2 Projeto e implementação de software

Com o projeto já clonado, inicialmente se atualiza o repositório local para garantir sua conformidade com o repositório remoto. A tarefa é selecionada de acordo com sua prioridade, e uma nova *branch* é criada para dar início à sua implementação. Após a conclusão dessa implementação, é feito os *commits* e o envio das alterações para o repositório. Em seguida, é aberta uma solicitação de *pull request* para revisão do código pelo orientador. Se aprovada, segue o fluxo para implantação nos ambientes. Se reprovada, retorna ao desenvolvimento para realizar os ajustes e correção solicitados durante a revisão de código.

Neste projeto, será utilizado o GitLab Flow[6] como fluxo de trabalho para implementação das funcionalidades. A justificativa para esta esolha é que este é o fluxo adotado no projeto da FARMA. O GitLab Flow é um fluxo de trabalho simplificado que consiste em três *branches* principais:

1. Main - É a *branch* principal do projeto, onde contém todo código desenvolvido até o momento (não necessariamente esta disponível em algum ambiente);
2. Staging - É a *branch* responsável pelo ambiente de teste/homologação, quando finalizado os testes, é incrementada a *production*;
3. Production - É a *branch* responsável pelo ambiente de produção, sendo o ambiente o qual o usuário final acessa.

Além disso, as *branches* de suporte, como:

1. Feature - Criada a partir da main para desenvolvimento de novas funcionalidades;

[5] Itens de trabalho que foram identificados e estão aguardando serem priorizados e movidos para o fluxo de trabalho ativo

[6] https://www.zup.com.br/blog/git-workflow

2. BugFix - Criada a partir de staging quando são necessárias correções. Quando finalizado o desenvolvimento é incrementado a main e staging;

3. HotFix - Criada a partir de production, quando é necessário a correção de erros urgentes em ambiente de produção. Quando finalizado o desenvolvimento é incrementado a main, staging e production.

Essa divisão ajuda a permitir controle e versionamento de código, proporcionando a rastreabilidade, a entrega contínua de novas funcionalidades, correção de erros e acomodação de correções críticas, garantindo um ciclo de desenvolvimento completo. O fluxo do GitLab Flow segue conforme a Figura 10.

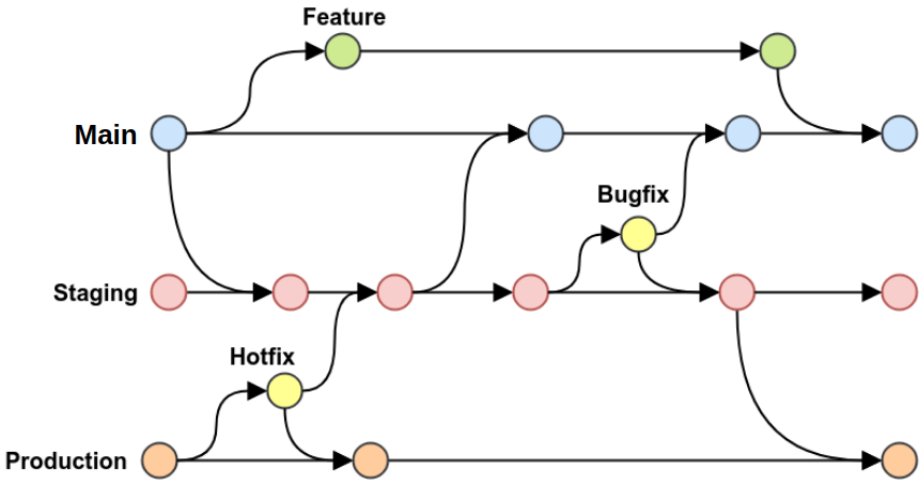



**Figura 10 – Gitlab flow**
**Fonte: https://www.zup.com.br/blog/git-workflow.**

### 4.2.3 Verificação e validação do software

Considerando que o projeto é focado no desenvolvimento de uma API e não inclui validação com usuários, os testes automatizados serão aplicados para verificar as regras de negócio e as rotas da API desenvolvidas. A validação será realizada durante a revisão do código pelo orientador, onde também ocorrerá a análise dos testes automatizados para verificar se foram escritos de acordo com esperado. Os códigos e testes desenvolvidos sendo aprovados pelo orientador serão incrementados no ambiente de teste e posteriormente no de produção. Após a integração no ambiente de produção, qualquer problema desencadeará a criação de uma tarefa prioritária para resolver quaisquer problemas encontrados.

### 4.2.4 Entrega e Implantação

Alinhado ao GitLab Flow será utilizado uma esteira Integração Contínua, do inglês *Continuous Integration* (CI)/Entrega Contínua, do inglês *Continuous Delivery* (CD) como parte fundamental do processo. Essa abordagem abrangerá as fases de desenvolvimento, teste, integração e implantação de código de maneira contínua, conforme é apresentado na Figura 11.

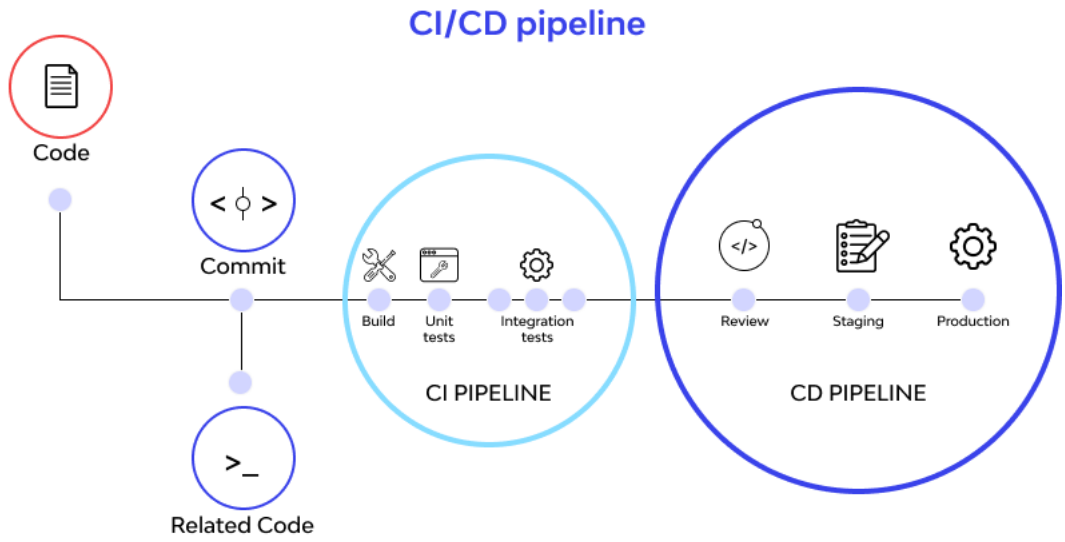



**Figura 11 – Esteira CI/CD**
**Fonte: https://blog.xpeducacao.com.br/pipeline-ci-cd/.**

A esteira CI é responsável por garantir que o código incrementado ao repositório segue os padrões pré-estabelecidos e que após o desenvolvimento o código ainda continua funcionando da forma esperada. Para isso foi configurado a esteira CI no Github actions com 2 passos (*jobs*), sendo eles:

O primeiro é a execução dos testes, onde aplicação é executada em uma máquina Ubuntu a partir de um container com a imagem do Ruby 3.2.2 juntamente com um container executando o PostgreSQL. Em seguida são instaladas as dependências necessárias, o banco de dados de teste é preparado e então são executados os testes.

O segundo é empregado para avaliar a integridade do código. Ele segue uma estrutura muito similar ao passo anterior, executando em uma máquina Ubuntu, a partir de um container com a imagem do Ruby 3.2.2, juntamente com a instalação das dependências necessárias. Serão executados três verificadores de código, que são:

1. RuboCop, um analisador estático e formatador de código Ruby. Além de identificar problemas no código, o RuboCop também oferece correções automáticas para alguns erros;

2. Brakeman, uma ferramenta de análise estática que verifica aplicativos Ruby on Rails em busca de possíveis vulnerabilidades de segurança;

3. RubyCritic, uma ferramenta que combina análises estáticas de *gems*[7] como Reek, Flay e Flog para gerar um relatório sobre a qualidade do código Ruby.

A esteira de CD é responsável por coordenar a entrega contínua e a implantação da aplicação em seus ambientes específicos, garantindo um fluxo eficiente e automatizado do desenvolvimento até produção.

Para realizar a entrega, um das abordagens é configurar um Servidor Virtual Privado, do inglês *Virtual Private Server* (VPS). Nessa VPS, implementamos o Docker para gerenciar os containers, e o Docker Compose para coordenar os serviços relacionados à aplicação e ao banco de dados. Adicionalmente, o servidor de aplicações nginx foi adicionado à VPS para desempenhar a função de *proxy* reverso, encaminhando as solicitações para o container da aplicação. Para garantir a segurança no acesso à VPS, efetuamos a configuração do Shell Seguro, do inglês *Secure Shell* (SSH) utilizando chaves públicas e privadas.

Com a VPS devidamente configurada, foram estabelecidos os procedimentos a serem executados pelo Github Actions para a entrega contínua. Nesse contexto, um *deploy*[8] é acionado a cada atualização recebida na *branch* principal. Com o objetivo de evitar inconsistências, a inclusão de novas funcionalidades e correções só é permitida por meio de Pedido de Mesclagem, do inglês *Pull Request* (PR). Após a aceitação de um PR na *branch* principal, o processo de *deploy* é iniciado, seguindo os passos descritos a seguir:

1. Etapas do CI

   Se não houver erro em alguma etapa do CI, é feito o *deploy* a partir dos passos:

2. É construída uma nova imagem da aplicação, com o código atual;

3. A imagem é registrado no Container Registry do Github;

4. O servidor é acessado via SSH usando a chave privada;

5. É feito download da imagem mais atual da aplicação do Container Registry;

6. É reiniciada a aplicação com a imagem mais atual.

Essas automações são feitas a partir do *github actions* que é um serviço integrado ao github que permite automatizar tarefas, facilitando o processo de desenvolvimento diretamente no GitHub.

---

[7] Bibliotecas, também chamadas de pacotes, são conjuntos de módulos, que são simples arquivos de código, destinadas a facilitar o desenvolvimento de novos projetos de software. No ruby sendo chamada de gem.

[8] Processo de implantação do software.

# 5 RESULTADOS PARCIAIS

Neste capítulo, serão expostos os resultados alcançados em cada fase do desenvolvimento deste projeto.

## 5.1 Requisitos

A fase inicial do desenvolvimento deste trabalho consistiu na identificação e coleta dos requisitos essenciais para a API, conforme detalhado a seguir.

### 5.1.1 Requisitos funcionais

- A API deve fornecer os dados para visualização de um objeto de aprendizagem, contendo as informações de percentual no objeto de aprendizagem (percentual concluído, percentual explorado e percentual não explorado), páginas do objeto de aprendizagem e dados da primeira página do objeto de aprendizagem;
- A API deve fornecer os dados para visualização de uma página de introdução de um objeto de aprendizagem, retornando as informações de tema, descrição e imagens, vídeos ou links relacionados ao tema;
- A API deve fornecer os dados para visualização de uma página de exercício de um objeto de aprendizagem, retornando as informações de enunciado, passos de solução, dicas e quantidade de respostas enviadas;
- A API deve alterar o status da introdução e do exercício para visualizado durante a requisição;
- A API deve permitir atualizar o status de um passo de solução para visualizado. Este evento acontece a partir da posição do scroll em relação ao passo de solução;
- A API deve permitir enviar uma resposta para o passo de solução. Deve retornar se a resposta é correta ou não, o histórico de respostas, um lista de dica recebidas e se existe uma nova dica ou não nessa lista;
- A API deve armazenar o histórico de respostas enviadas para um passo de solução;
- A API deve armazenar o histórico de um contexto de uma resposta enviada para um passo de solução;
- A API deve permitir solicitar dicas para responder uma questão, armazenando o histórico de solicitações. Deve retornar a nova dica caso exista.

### 5.1.2 Requisitos não funcionais

- O código da API deve seguir boas práticas de programação e ser de fácil manutenção;
- A API deve aderir a padrões e princípios estabelecidos, como RESTFULL, para garantir consistência e interoperabilidade.

## 5.2 Banco de dados

A criação do modelo de banco de dados envolve a coleta, análise, categorização e exploração abrangente de dados e informações que serão fundamentais para dar suporte a uma aplicação. Essa fase é crucial no desenvolvimento de sistemas, pois todo software é concebido com metas específicas, visando atender às demandas dos usuários dentro desse contexto (REDES, 2020).

Na Figura 12 é apresentado parte do modelo do banco de dados necessário para visualização e interação com os objetos de aprendizagem na FARMA.

A tabela `los` representa os objetos de aprendizagem, cada um identificado por um ID único. Cada objeto de aprendizagem tem um título e uma descrição mais detalhada. O campo `user_id` está associado ao professor responsável pela criação do objeto de aprendizagem.

Ambas as tabelas `exercises` e `introductions` contêm dados específicos para Exercícios e Introduções, como título, descrição, visibilidade pública e posição na sequência de um objeto de aprendizagem. A relação com a tabela los através do campo `lo_id` permite vincular Exercícios e Introduções a um objeto de aprendizagem específico.

A tabela `solution_steps` armazena informações detalhadas sobre os passos de solução para os exercícios, incluindo título, descrição, resposta esperada e precisão necessária. A relação com a tabela `exercises` é estabelecida pelo campo `exercise_id`, conectando os passos de solução aos exercícios correspondentes.

A tabela `tips` armazena dicas relacionadas aos exercícios, incluindo o número de tentativas permitidas e a posição na sequência do exercício. A relação com a tabela `solution_steps` é estabelecida pelo campo `solution_step_id`, associando as dicas aos passos de solução correspondentes.

As tabelas `exercises_visualizations`, `introductions_visualizations`, `solution_steps_visualizations`, `tips_visualizations` rastreiam as visualizações de diferentes elementos (exercícios, introduções, passos de solução e dicas) pelos usuários. A partir disso, é possível calcular quanto o usuário explorou do objeto de aprendizagem.

A tabela `answers` armazena as respostas enviadas para um passo de solução pelos usuários, registrando a resposta enviada, se está correta ou não e qual é o número da tentativa.

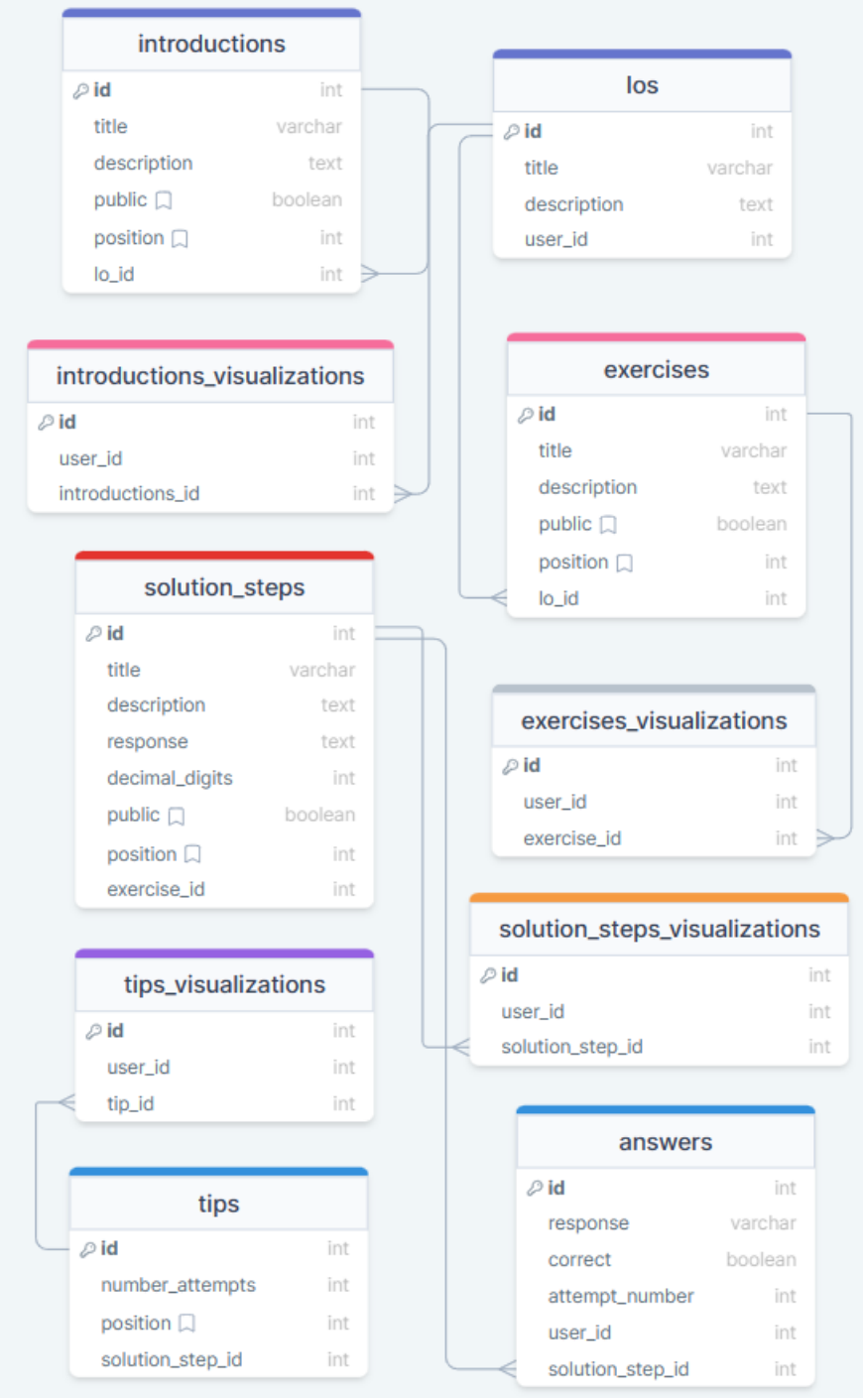



**Figura 12 – Modelo do banco de dados para visualização e interação de objetos de aprendizagem na FARMA**

**Fonte: Autoria prórpia.**

Na seção 5.3, será apresentada uma explicação mais detalhada de como essa modelagem resolve o problema proposto.

## 5.3 Rotas da API

A seguir são apresentadas as rotas de API necessárias para atender os requisitos. Serão desenvolvidas rotas para usuários autenticados: aluno e professor e rotas públicas para usuário vistante. O professor pode obter os dados apenas apenas dos objetos de aprendizagem que criou e assim como o usuário vistante não tem as informações de interação armazenadas.

Rota 1: Rota de detalhe do objeto de aprendizagem - `GET: /los/:id`

Rota 2: Rota de detalhe de página de um objeto de aprendizagem específico - `GET: /los/:id/page/:page`

Rota 3: Rota de solicitação de dicas - `GET: /exercises/:exercise_id/solution_steps/:id/tips/request`

Rota 4: Rota de confirmação de visualização de passo de solução - `PATCH: /exercises/:exercise_id/solution_steps/:id/view`

Rota 5: Rota de envio de resposta para passo de solução - `POST: /exercises/:exercise_id/solution_steps/:id/response`

### 5.3.1 Rota de detalhe do objeto de aprendizagem

A Rota 1 retorna os dados necessários para montar a página de visualização do objeto de aprendizagem. Contendo as informações de:

1. Lista com as páginas do objeto de aprendizagem (Introduções e Exercícios). Se exercício, deve retornar os passos de solução do exercício, com seu respectivo status (Acerto, Erro, Visualizado ou Não Visualizado);
2. Dados da primeira página do objeto de aprendizagem;
3. Percentuais referente ao rendimento no objeto de aprendizagem: percentual concluído, percentual explorado e percentual inexplorado.

Para obter essas informações, são coletados dados da tabela `los`, estabelecendo relações com as tabelas `introductions` e `exercises` para recuperar as páginas do objeto de aprendizagem. Os passos de solução são obtidos através dos dados da tabela `solution_steps`, que se relaciona com a tabela

exercises. O cálculo do percentual é realizado considerando a quantidade de visualizações de passos de solução e páginas do objeto de aprendizagem, extraídas das tabelas: exercises_visualizations, introductions_visualizations, solution_steps_visualizations e tips_visualizations.

Assim, atendendo a necessidade de fornecer os dados para visualização de um objeto de aprendizagem. Além disso durante a requisição o status da primeira página é atualizado para visualizado pelo usuário.

### 5.3.2 Rota de detalhe de página de um objeto de aprendizagem específico

A Rota 2 retorna os dados necessários para visualização de uma página do objeto de aprendizagem, podendo ser ela uma página de: Introdução ou Exercício. Para página de introdução, contém as informações de:

1. Título;
2. Conteúdo que pode conter uma descrição, imagens e vídeo;

Esses são obtidos da tabela introductions.
Para página de exercício, contém as informações de:

1. Título;
2. Conteúdo que pode conter uma descrição, imagens e vídeo;
3. Passos de solução, onde cada um contém:
    a) Título;
    b) Conteúdo que pode conter uma descrição, imagens e vídeo;
    c) Quantidade de tentativas de resposta;
    d) Status (Acerto, Erro, Visualizado ou Não Visualizado);
    e) Dica.

Esses dados podem ser obtidos da tabela exercises.

### 5.3.3 Rota de solicitação de dicas

A Rota 3 retorna uma dica se ainda existir alguma disponível para visualização, a cada solicitação a dica é sinalizada como visualizada.

A dica é obtida da tabela tips. Para verificar se existe alguma dica ainda disponível para ser visualizada é feita uma consulta verificando qual dica não foi visualizada, verificando qual dica não existe na tabela tips_visualizations.

### 5.3.4 Rota de confirmação de visualização de passo de solução

A Rota 4 confirma a visualização de um passo de solução. A requisição para essa rota será feita a partir do evento do scroll do mouse, quando tiver acima do passo de solução é chamada a rota para confirmar a visualização, enviando o identificador do usuário.

Para registrar a visualização é cadastrado um novo registro na tabela `solutions_steps_visualizations`, com o identificador do passo de solução e do usuário que visualizou.

### 5.3.5 Rota de envio de resposta para passo de solução

A Rota 5 permite enviar uma resposta para o passo de solução. Retorna se a resposta está correta ou não, o histórico de outras respostas enviadas, a lista de dicas já visualizados e se tem uma dica disponível ainda.

## 5.4 Backlog

A seguir é apresentado o backlog do projeto para o desenvolvimento dos recursos necessário para visualização e interação de objeto de aprendizagem.

### 5.4.1 Configuração de modelos necessários

Para o desenvolvimento dos recursos, configurar os modelos a seguir:

1. Configurar o modelo `LearningObject` para representar os dados de um objeto de aprendizagem, incluindo campos como título, descrição e por qual usuário foi criado;
2. Configurar o modelo `Introduction` para representar uma página de introdução, contendo campos como título, descrição (texto descritivo, imagens e vídeo), público, posição, relação ao objeto de aprendizagem e relação com a tabela *introductions_visualiations* para ser possível verificar os usuários que visualizaram;
3. Configurar o modelo `Exercise` para representar uma página de exercício, incluindo campos como título, descrição (texto descritivo, imagens e vídeo), público, posição e relação ao objeto de aprendizagem a que pertence e relação com a tabela *exercises_visualiations* para ser possível verificar os usuários que visualizaram;;
4. Configurar o modelo `SolutionStep` para representar o passos de solução com campos como título, descrição, resposta esperada, digítos decimais aceitos, posição, se é público ou não, relação ao exercício e relação com a tabela *solutions_steps_visualiations* para ser possível verificar os usuários que visualizaram;

5. Configurar o modelo `Tip` para representar as dicas, contendo campos como descrição, número de tentativas, posição, relação com o passo de solução e relação com a tabela *tips_visualiations* para ser possível verificar os usuários que visualizaram.

6. Configurar o modelo `Answer` para representar as resposta, contendo campos como resposta, número da tentativa, se é resposta correta e relação com o passso de solução.

### 5.4.2 Criar rota de detalhe de objeto de aprendizagem para aluno e professor

Criar Rota 1 (conforme seção 5.3) para detalhe do objeto de aprendizagem, contendo os seguintes dados no retorno

1. Lista com as páginas do objeto de aprendizagem (Introduções e Exercícios). Se exercício, deve retornar os passos de solução do exercício, com seu respectivo status (Acerto, Erro, Visualizado ou Não Visualizado);

2. Dados da primeira página do objeto de aprendizagem;

3. Percentuais referente ao rendimento no objeto de aprendizagem: percentual concluído, percentual explorado e percentual inexplorado.

Se o solicitante for um professor, somente deve ser retornado o objeto de aprendizagem se o mesmo tiver sido criado por ele.

### 5.4.3 Criar rota de detalhe de página de um objeto de aprendizagem específico

Criar Rota 2 (conforme seção 5.3), contendo os seguintes dados de retorno:
Se página de introdução:

1. Título;

2. Conteúdo que pode conter uma descrição, imagens e vídeo;

Se página de exercício:

1. Título;

2. Conteúdo que pode conter uma descrição, imagens e vídeo;

3. Passos de solução, onde cada um contém:

    a) Título;

    b) Conteúdo que pode conter uma descrição, imagens e vídeo;

c) Quantidade de tentativas de resposta;

d) Status (Acerto, Erro, Visualizado ou Não Visualizado);

e) Dica.

Para o usuário visitante não serão armazenadas as informações no servidor, sendo assim, as informações de status dos passos de solução do exercício (Acerto, Erro, Visualizado ou Não Visualizado) e quantidade de tentativas de resposta serão armazenados do lado do cliente.

### 5.4.4 Criar rota de solicitação de dicas

Criar Rota 3 (conforme 5.3), contendo os seguintes dados de retorno:

1. Uma dica se ainda existir alguma disponível para visualização;

A cada solicitação a dica é sinalizada como visualizada.

### 5.4.5 Criar rota de confirmação de visualização

Criar Rota 4 (conforme seção 5.3), que confirma a visualização de um passo de solução. A requisição para essa rota será feita a partir do evento do scroll do mouse, devendo retornar sucesso se registrada a visualização.

### 5.4.6 Criar a rota de envio de resposta para passo de solução

Criar Rota 5 (conforme seção 5.3), contendo os seguintes dados de retorno:

1. Indicação se a resposta está correta ou não;
2. Histórico de outras respostas enviadas;
3. Lista de dicas já visualizadas;
4. Indicação se tem dicas disponíveis ainda.

### 5.4.7 Criar rotas públicas para usuário visitante

Replicar as Rotas 1,2,3,4 e 5 (conforme seção 5.3) para usuário visitante. Essas rotas tem como diferença o não armazenamento das interações do usuário para geração de estatísticas.

# 6 CONSIDERAÇÕES FINAIS

Conforme mencionado no Capítulo 2, a FARMA iniciou uma nova versão com o objetivo de incorporar contribuições apresentadas nos trabalhos de Correia (2017) e Santos (2018), resultando em uma versão estável Vitek (2023). O presente trabalho tem como meta dar continuidade ao desenvolvimento dessa nova versão, focando nos recursos *back-end* necessários para a visualização de objetos de aprendizagem na FARMA. Para a implementação da API, será utilizado o framework Ruby On Rails, e a documentação seguirá a especificação da OpenAPI. A expectativa é que parte da funcionalidade de visualização e interação de objetos de aprendizagem seja incorporada à nova versão da FARMA, abrindo espaço para a implementação do *front-end* em futuras contribuições.

# REFERÊNCIAS


AGUIAR, E. V. B.; FLÔRES, M. L. P. Objetos de aprendizagem: conceitos básicos. **Objetos de aprendizagem: teoria e prática. Porto Alegre: Evangraf**, p. 12–28, 2014.

ANDRADE, E. Entendendo as diferenças entre apis rest e restfull. 2023. Disponível em: https://www.dio.me/articles/entendendo-as-diferencas-entre-apis-rest-e-restful. Acesso em: 07 nov. 2023.

AWS. O que é uma api (interface de programação de aplicações)? 2023. Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 07 nov. 2023.

AWS, D. Qual é a diferença entre grpc e rest? 2023. Disponível em: https://aws.amazon.com/pt/compare/the-difference-between-grpc-and-rest/#:~:text=No%20gRPC%2C%20um%20componente%20(o,atualiza%20os%20dados%20no%20servidor. Acesso em: 09 nov. 2023.

CLOUDFLARE. O que é um endpoint de api? 2023. Disponível em: https://www.cloudflare.com/pt-br/learning/security/api/what-is-api-endpoint/. Acesso em: 28 nov. 2023.

CORREIA, L. Controle de versão para armazenamento do contexto da interação do aprendiz com objetos de aprendizagem. 2017.

FINCHER, J.; DESMOND, K. The 11 best backend frameworks – 2023. 2023. Disponível em: https://codingnomads.co/blog/best-backend-frameworks/. Acesso em: 29 set. 2023.

GRPC, D. Core concepts, architecture and lifecycle. 2022. Disponível em: https://grpc.io/docs/what-is-grpc/core-concepts/. Acesso em: 08 nov. 2023.

LIMA, M. F. d.; ARAúJO, J. F. S. d. A utilização das tecnologias de informação e comunicação como recurso didático-pedagógico no processo de ensino e aprendizagem. **Revista Educação Pública**, v. 21, n. 23, 2021. Disponível em: https://educacaopublica.cecierj.edu.br/artigos/21/23/a-utilizacao-das-tecnologias-de-informacao-e-comunicacao-como-recurso-didatico-pedagogico-no-processo-de-ensino-aprendizagem. Acesso em: 04 out. 2023.

MARCZAL, D. *et al.* Farma: Uma ferramenta de autoria para objetos de aprendizagem de conceitos matemáticos. Universidade Federal do Paraná, 2014.

MELO, D. O que é uma api? [guia para iniciantes]. **Tecnoblog**, 2023. Disponível em: https://tecnoblog.net/responde/o-que-e-uma-api-guia-para-iniciantes. Acesso em: 04 nov. 2023.

REDES, E. S. de. O que é a modelagem de banco de dados e quais os seus principais conceitos. 2020. Disponível em: https://esr.rnp.br/desenvolvimento-de-sistemas/open-5/. Acesso em: 25 nov. 2023.

SANTOS, E. L. D. S. Projeto de uma nova interface grafica para a ´ ferramenta de autoria de objetos de aprendizagem matematicos farma. 2018.

SOUZA, R. Educação. Brasil Escola, 2018. Disponível em: https://brasilescola.uol.com.br/educacao. Acesso em: 09 set. 2023.

TOTVS. O que é back-end e qual seu papel na programação? 2020. Disponível em: https://www.totvs.com/blog/developers/back-end/. Acesso em: 28 nov. 2023.

TOTVS. Front end: O que é, como funciona e qual a importância. 2021. Disponível em: https://www.totvs.com/blog/developers/front-end/. Acesso em: 28 nov. 2023.

VIDANYA, B. Graphql vs. rest: Qual o melhor para o desenvolvimento de api? 2021. Disponível em: https://www.hostinger.com.br/blog/graphql-vs-rest-qual-o-melhor-para-o-desenvolvimento-de-api/. Acesso em: 07 nov. 2023.

VITEK, I. Desenvolvimento da Área destinada À criaÇÃo de objetos de aprendizagem para ferramenta de autoria farma. 2023.