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

2024

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

Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, 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

2024

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.

DOUGLAS VINICIUS CALDAS BONIN

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

Trabalho de Conclusão de Curso de Graduação apresentado à disciplina de Trabalho de Conclusão de Curso 2, 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.

Data de aprovação: 02/julho/2024

Prof. Diego Marczal
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Eleandro Maschio
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

---

Prof. Luciano Ogiboski
Doutor
Universidade Tecnológica Federal do Paraná - Campus Guarapuava

GUARAPUAVA
2024

AGRADECIMENTOS

Ao concluir esta etapa tão importante da minha vida acadêmica, gostaria de expressar minha profunda gratidão a todas as pessoas que contribuíram para a realização deste trabalho.

Primeiramente, agradeço a Deus pelos dons, pela sabedoria, saúde e força durante toda a jornada até aqui.

Aos meus pais, pelo amor incondicional, apoio constante e incentivo em todos os momentos. Vocês são minha base e fonte de inspiração.

Aos meus avós e tio, por estarem sempre ao meu lado, oferecendo palavras de encorajamento e compreensão nos momentos de dificuldade.

Ao meu orientador, Prof. Dr. Diego Marczal, e à minha coorientadora, Profª. Drª. Renata Luiza Stange Carneiro Gomes, pela disponibilidade, paciência, orientação e dedicação. Seus conselhos e críticas foram fundamentais para o desenvolvimento deste trabalho.

Aos excelentes professores que tive durante o curso, colegas e amigos que fiz. A convivência e troca de conhecimentos foram essenciais para o meu crescimento pessoal e profissional.

À UTFPR, por proporcionar um ambiente de aprendizado e crescimento. A estrutura e os recursos disponibilizados foram essenciais para o meu desenvolvimento.

A todas as pessoas que, direta ou indiretamente, colaboraram para a realização deste trabalho, expresso meu sincero agradecimento.

Por fim, dedico este trabalho a todos que sempre estiveram comigo, me apoiaram e incentivaram.

Muito obrigado!

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 está dividida 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, foi 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. Este trabalho teve como implementação parte da funcionalidade de visualização e interação dos objetos de aprendizagem na 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 dá continuidade ao avanço da nova versão da FARMA, possibilitando a implementação do *front-end* em uma próxima contribuição.

Palavras-chave: objetos de aprendizagem; ferramenta; ensino; sistemas educacionais.

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 was used to implement the API, the OpenAPI specification was be used to document the API, and GitLab Flow was used to control the workflow with the aim of Continuous Integration/Delivery. This work implemented, part of the functionality for visualization and interacting with learning objects in 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 implementation of *front-end* in a future contribution.

Keywords: learning objects; tool; teaching; educational systems.

LISTA DE FIGURAS

Figura 1 – GitLab Flow 18
Figura 2 – Esteira CI/CD 19
Figura 3 – Página principal da Farma Reborn 22
Figura 4 – Introdução do objeto de aprendizagem 23
Figura 5 – Enunciado do objeto de aprendizagem 24
Figura 6 – Passos de solução no exercício do objeto de aprendizagem 25
Figura 7 – Orientações e dicas do objeto de aprendizagem 26
Figura 8 – Histórico no objeto de aprendizagem 26
Figura 9 – Percentuais no objeto de aprendizagem 26
Figura 10 – Visualização do objeto de aprendizagem na aba turma - Aluno 27
Figura 11 – Objeto de aprendizagem - Aluno 28
Figura 12 – Modelo do banco de dados para visualização e interação de objetos de aprendizagem na FARMA 29
Figura 13 – Execução de testes - PR 2 33
Figura 14 – Execução de testes - PR 3 34
Figura 15 – Execução de testes - PR 4 35
Figura 16 – Execução de testes - PR 5 37
Figura 17 – Execução de testes - PR 6 39
Figura 18 – Execução de testes para modelos - PR 6 41
Figura 19 – Execução de testes - PR 7 42
Figura 20 – Execução de testes - PR 8 44
Figura 21 – Execução de testes - PR 9 45
Figura 22 – Execução de testes - PR 10 45
Figura 23 – Execução de testes - PR 11 49
Figura 24 – Execução de testes - PR 12 50
Figura 25 – Execução de testes - PR 13 51
Figura 26 – Execução de testes - PR 14 52
Figura 27 – Execução de testes - PR 15 53
Figura 28 – Execução de testes - PR 16 54

Figura 29 – Modelo do banco de dados para turmas de objetos de aprendizagem na FARMA 57
Figura 30 – Documentação da rota - PR 2 59
Figura 31 – Documentação da rota (resultado) - PR 2 59
Figura 32 – Documentação da rota - PR 3 60
Figura 33 – Documentação da rota (resultado) - PR 3 60
Figura 34 – Documentação da rota - PR 4 61
Figura 35 – Documentação da rota (resultado) - PR 4 61
Figura 36 – Documentação da rota - PR 5 62
Figura 37 – Documentação da rota (resultado) - PR 5 62
Figura 38 – Documentação da rota - PR 6 63
Figura 39 – Documentação da rota (resultado) - PR 6 63
Figura 40 – Documentação da rota - PR 7 64
Figura 41 – Documentação da rota (resultado) - PR 7 64
Figura 42 – Documentação da rota - PR 8 65
Figura 43 – Documentação da rota (resultado) - PR 8 65
Figura 44 – Documentação da rota - PR 9 66
Figura 45 – Documentação da rota (resultado) - PR 9 66
Figura 46 – Documentação da rota - PR 10 67
Figura 47 – Documentação da rota (resultado) - PR 10 67
Figura 48 – Documentação da rota - PR 11 68
Figura 49 – Documentação da rota (resultado) - PR 11 68
Figura 50 – Documentação da rota - PR 12 69
Figura 51 – Documentação da rota (resultado) - PR 12 69
Figura 52 – Documentação da rota - PR 13 70
Figura 53 – Documentação da rota (resultado) - PR 13 70
Figura 54 – Documentação da rota - PR 14 71
Figura 55 – Documentação da rota (resultado) - PR 14 71
Figura 56 – Documentação da rota - PR 15 72
Figura 57 – Documentação da rota (resultado) - PR 14 72
Figura 58 – Documentação da rota - PR 16 73
Figura 59 – Documentação da rota - PR 16 73

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 10

1.1 Contextualização 10

1.2 Objetivos 12

1.2.1 Objetivo geral 12

1.2.2 Objetivos específicos 12

1.3 Justificativa 13

1.4 Organização 13

2 MATERIAIS E MÉTODOS 14

2.1 Materiais 14

2.1.1 API - Interface de Programação de Aplicações 14

2.2 Métodos 16

2.2.1 Levantamento e especificação de requisitos 17

2.2.2 Projeto e implementação de software 17

2.2.3 Verificação e validação do software 19

2.2.4 Entrega e Implantação 19

3 ANÁLISE E PROJETO 22

3.1 Descrição da FARMA 22

3.1.1 Visualização do objeto de aprendizagem 23

3.2 Requisitos Funcionais e Não Funcionais 25

3.3 Modelagem do Banco de dados 28

4 DESENVOLVIMENTO DA FARMA API 31

4.1 Backlog 31

4.2 Implementação das Rotas da API 32

4.2.1 Rota de detalhe do objeto de aprendizagem para professores, alunos e visitantes 32

4.2.2 Rota de detalhe de página de um objeto de aprendizagem específico para professores, alunos e visitantes 35

4.2.3 Rota de confirmação de visualização em um passo de solução para professores, alunos e visitantes 42

4.2.4 Rota de envio de resposta para o passo de solução para professores, alunos e visitantes 46
4.2.5 Rota de solicitação de dicas para professores, alunos e visitantes 51
4.3 Rotina para limpeza de interações de usuários: professor e visitante . . 54
5 RESULTADOS OBTIDOS 57
5.1 Atualização do Banco de dados 57
5.2 Rotas da API 58
6 CONSIDERAÇÕES FINAIS 74
REFERÊNCIAS 75

1 INTRODUÇÃO

Neste capítulo, será apresentada uma visão geral do trabalho realizado, destacando o cenário no qual ele se aplica, bem como a motivação e as justificativas para seu desenvolvimento. Também serão revisados os objetivos que foram cumpridos para a conclusão do trabalho.

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 mostra 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, 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.

Nesse 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 relaciona-

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

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

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

dos aos objetos de aprendizagem possam ser integrados à aplicação *front-end*[4], desta forma sendo possível observar os objetos de aprendizagem de forma interativa durante a resolução dos exercícios.

1.2 Objetivos

Nesta seção, são apresentados os objetivos gerais e específicos que orientaram a execução deste trabalho. Cada um desses objetivos foi definido para guiar a realização deste trabalho prático, com a aplicação do conhecimento em situações reais.

1.2.1 Objetivo geral

O objetivo geral deste trabalho é desenvolver 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:

[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 interagirá diretamente, seja em softwares, sites, aplicativos, entre outros (TOTVS, 2021).

a) acesso aos recursos do objeto de aprendizagem;

b) acerto;

c) erro;

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 são elencados os materiais e métodos utilizados para o desenvolvimento do trabalho. No Capítulo 3 são delineados os principais requisitos funcionais e não funcionais da API e outros aspectos do projeto. No Capítulo 4 é descrito o desenvolvimento do trabalho, incluindo a implementação da API. No Capítulo 5 são apresentados os principais resultados obtidos com este trabalho. Por fim, no Capítulo 6 são apresentadas as considerações finais do projeto de trabalho de conclusão de curso.

2 MATERIAIS E MÉTODOS

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

2.1 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 foi utilizado o quadro Kanban no GitHub Projects[1] que permite um melhor acompanhamento do andamento das atividades a serem desenvolvidas;
- *Framework* de desenvolvimento: Para o desenvolvimento da API foi utilizado Ruby on Rails[2], um dos frameworks mais renomados e consolidados do mercado (FINCHER; DESMOND, 2023). Essa tecnologia permitiu 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 teve um papel crucial na garantia da qualidade do projeto, assegurando que a API funciona de maneira confiável e sem falhas;
- Especificação da API: Para a composição da documentação da API foi utilizada a especificação da OpenAPI[3], que é uma linguagem de especificação para Protocolo de Transferência de Hipertexto, do inglês *Hypertext Transfer Protocol* (HTTP) API'S. Essa documentação permitiu que os recursos da API fossem descritos a partir de um padrão utilizado no mercado;
- Sistema Gerenciador de Banco de Dados: Para gerenciar o banco de dados foi utilizado PostgreSQL[4]. Essa escolha foi feita baseada no trabalho desenvolvido por Correia (2017), que projeta uma nova modelagem do banco de dados.

2.1.1 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

[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/

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).

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.

REST define um conjunto de operações a partir dos métodos 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, PATCH, 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.

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*[5] 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).

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[6] 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 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).

2.2 Métodos

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

[5] Lugar onde esses pedidos (referidos como chamadas de API) são respondidos (CLOUDFLARE, 2023).

[6] https://protobuf.dev/overview/

2.2.1 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[7] 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.

2.2.2 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, são feitos 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.

Cada *pull request* segue o modelo da Tabela 1:

Title	Título da PR
Issue	Identificador único da PR
Description	Descreve a tarefa de implementação realizada.
Motivation and Context	Descreve a razão da implementação da tarefa.
How has this been tested?	Descreve os tipos de testes que foram realizados.
Types of changes	Descreve o tipo de implementação que foi realizada, podendo ser uma nova funcionalidade, refatoração ou uma correção.
Checklist	Lista de items para confirmar, se o código segue estilo de código do projeto.
Hours Required	Horas necessárias para a realização da implementação.

Tabela 1 – PR Template

Neste projeto, foi utilizado o GitLab Flow[8] como fluxo de trabalho para implementação das funcionalidades. A justificativa para essa escolha está no fato de ser o fluxo adotado no

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

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

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;
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 necessária 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 1.

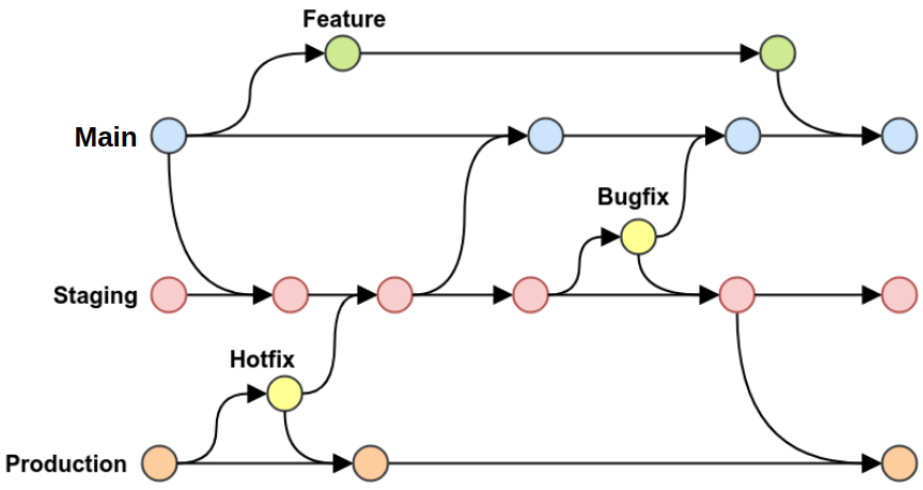


Figura 1 – GitLab Flow
Fonte: https://www.zup.com.br/blog/git-workflow.

2.2.3 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 foram aplicados para verificar as regras de negócio e as rotas da API desenvolvidas. A validação foi realizada durante a revisão do código pelo orientador, onde também ocorreu a análise dos testes automatizados para verificar se foram escritos de acordo com esperado. Os códigos e testes desenvolvidos aprovados pelo orientador foram 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.

2.2.4 Entrega e Implantação

Alinhado ao GitLab Flow foi utilizada uma esteira Integração Contínua, do inglês *Continuous Integration* (CI)/Entrega Contínua, do inglês *Continuous Delivery* (CD) como parte fundamental do processo. Essa abordagem abrange as fases de desenvolvimento, teste, integração e implantação de código de maneira contínua, conforme é apresentado na Figura 2.

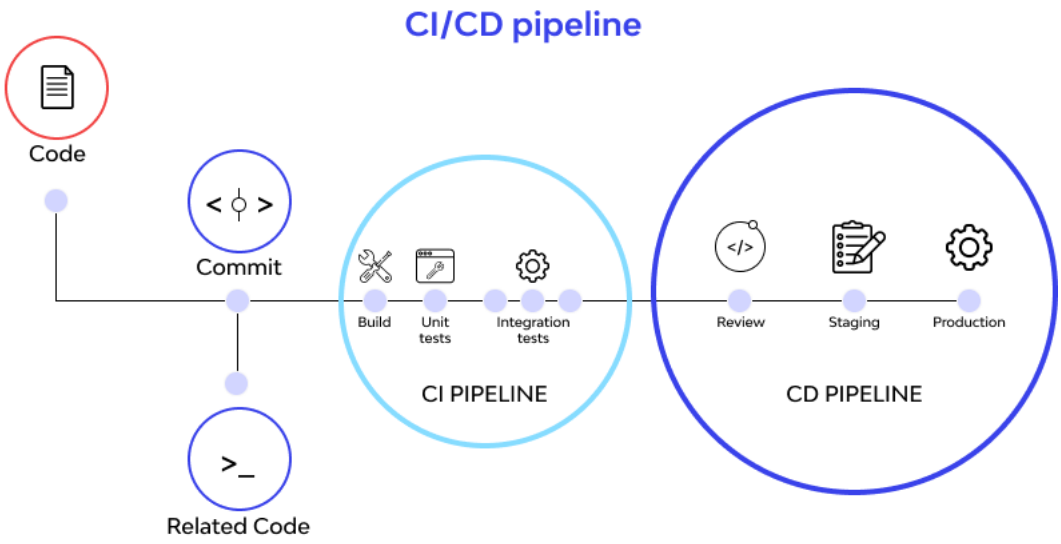


Figura 2 – Esteira CI/CD
Fonte: https://blog.xpeducacao.com.br/pipeline-ci-cd/.

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*[9] 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*[10] 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:

[9] 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.

[10] Processo de implantação do software.

a) É construída uma nova imagem da aplicação, com o código atual;
b) A imagem é registrado no Container Registry do Github;
c) O servidor é acessado via SSH usando a chave privada;
d) É feito *download* da imagem mais atual da aplicação do Container Registry;
e) 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.

3 ANÁLISE E PROJETO

Neste capítulo, será apresentado os resultados da etapa de análise e projeto da API.

3.1 Descrição da 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 3 apresenta a página principal dessa versão.



Figura 3 – 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 in-

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

teração 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.

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.

3.1.1 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 4;



Figura 4 – 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 5;

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 a Figura 6, 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;

d) Orientações e dicas relacionadas ao assunto em caso de respostas incorretas, conforme a Figura 7. 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;

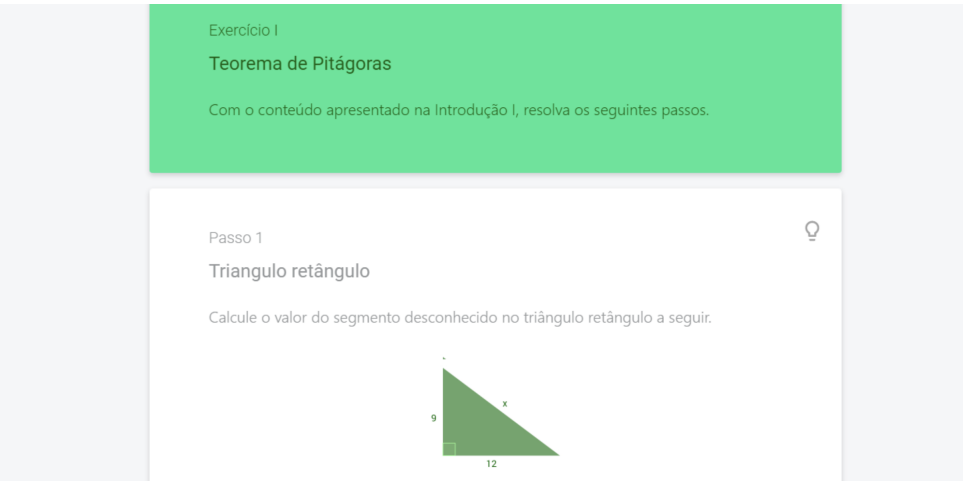


Figura 5 – Enunciado do objeto de aprendizagem
Fonte: http://farma-reborn.educacional.mat.br.

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 8, 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).

   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 9. Esses percentuais são divididos em: percentual concluído, percentual explorado e percentual inexplorado.

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 10 mostra um exemplo de visualização de um objeto de aprendizagem por um aluno.

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 11 mostra um exemplo desta visualização.

Dessa mesma maneira que foi apresentada nas figuras 10 e 11, o professor (com algumas pequenas adaptações) e o vistante conseguem visualizar o objeto de aprendizagem.

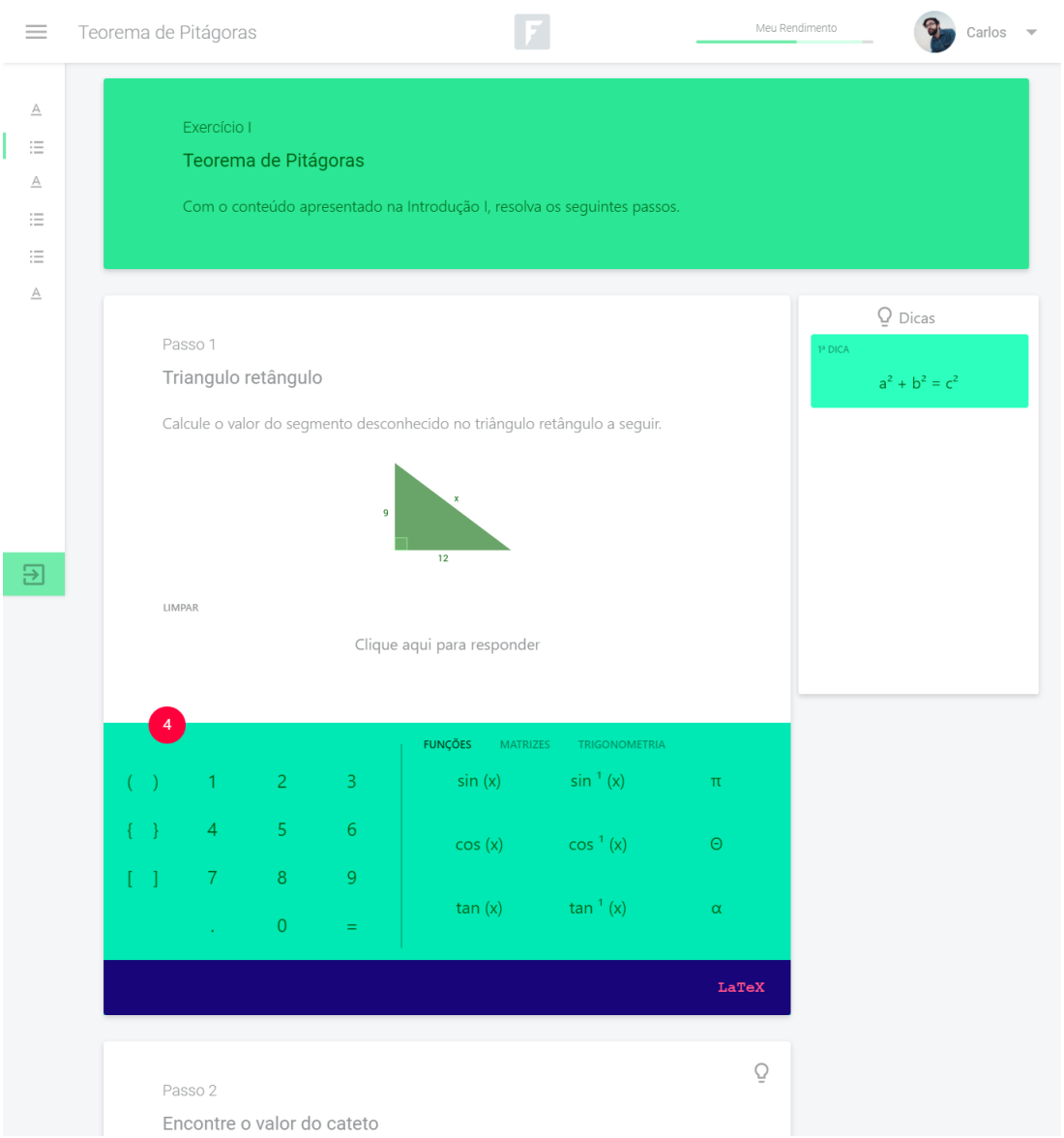


Figura 6 – Passos de solução no exercício do objeto de aprendizagem
Fonte: http://farma-reborn.educacional.mat.br.

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.

3.2 Requisitos Funcionais e Não Funcionais

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

Os requisitos funcionais elencados são:



Figura 7 – Orientações e dicas do objeto de aprendizagem
Fonte: http://farma-reborn.educacional.mat.br.

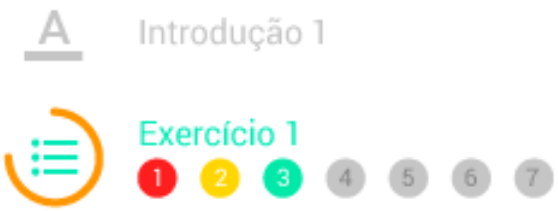


Figura 8 – Histórico no objeto de aprendizagem
Fonte: http://farma-reborn.educacional.mat.br.

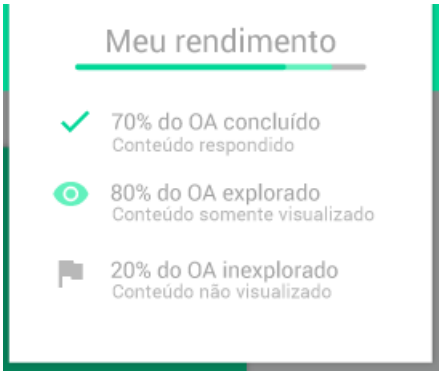


Figura 9 – Percentuais no objeto de aprendizagem
Fonte: http://farma-reborn.educacional.mat.br.

RF01 - 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;

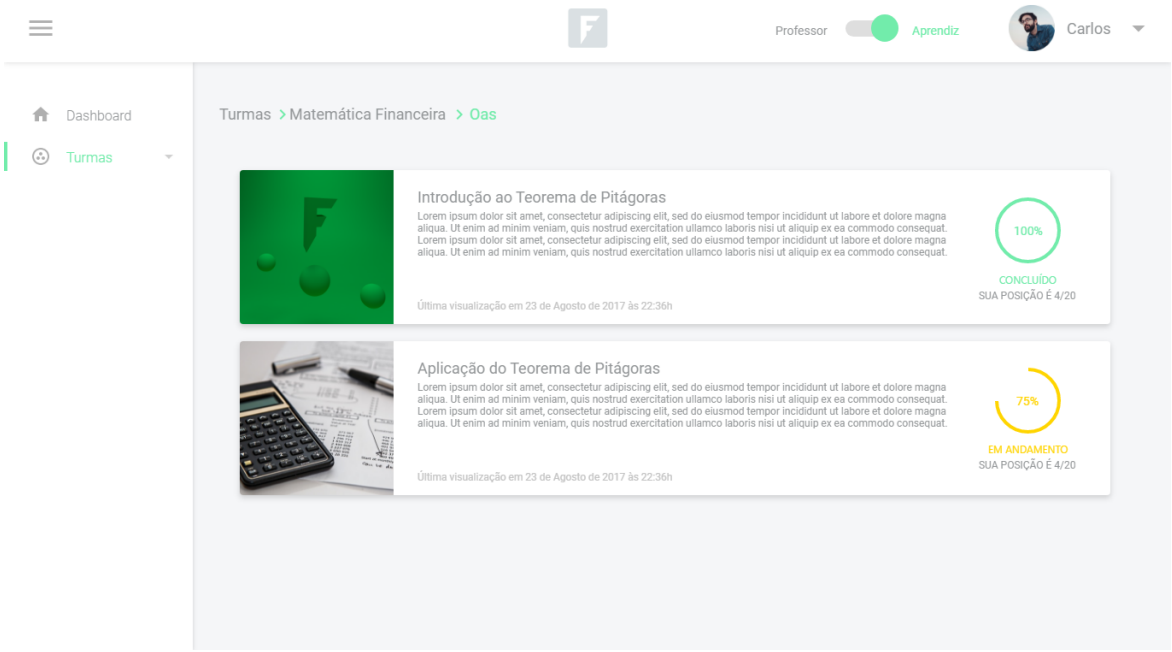


Figura 10 – Visualização do objeto de aprendizagem na aba turma - Aluno
Fonte: http://farma-reborn.educacional.mat.br.

RF02 - 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;

RF03 - 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;

RF04 - A API deve alterar o status da introdução e do exercício para visualizado durante a requisição;

RF05 - A API deve permitir atualizar o status de um passo de solução para visualizado. Esse evento acontece a partir da posição do scroll em relação ao passo de solução;

RF06 - 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 dicas recebidas e se existe uma nova dica ou não nessa lista;

RF07 - A API deve armazenar o histórico de respostas enviadas para um passo de solução;

RF08 - A API deve armazenar o histórico de um contexto de uma resposta enviada para um passo de solução;

RF09 - A API deve permitir solicitar dicas para responder uma questão, armazenando o histórico de solicitações. Deve retornar a nova dica caso exista.



Figura 11 – Objeto de aprendizagem - Aluno
Fonte: http://farma-reborn.educacional.mat.br.

Os requisitos não funcionais identificados são:

RNF01 - O código da API deve seguir boas práticas de programação e ser de fácil manutenção;

RNF02 - A API deve aderir a padrões e princípios estabelecidos, como RESTFULL, para garantir consistência e interoperabilidade.

3.3 Modelagem do 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

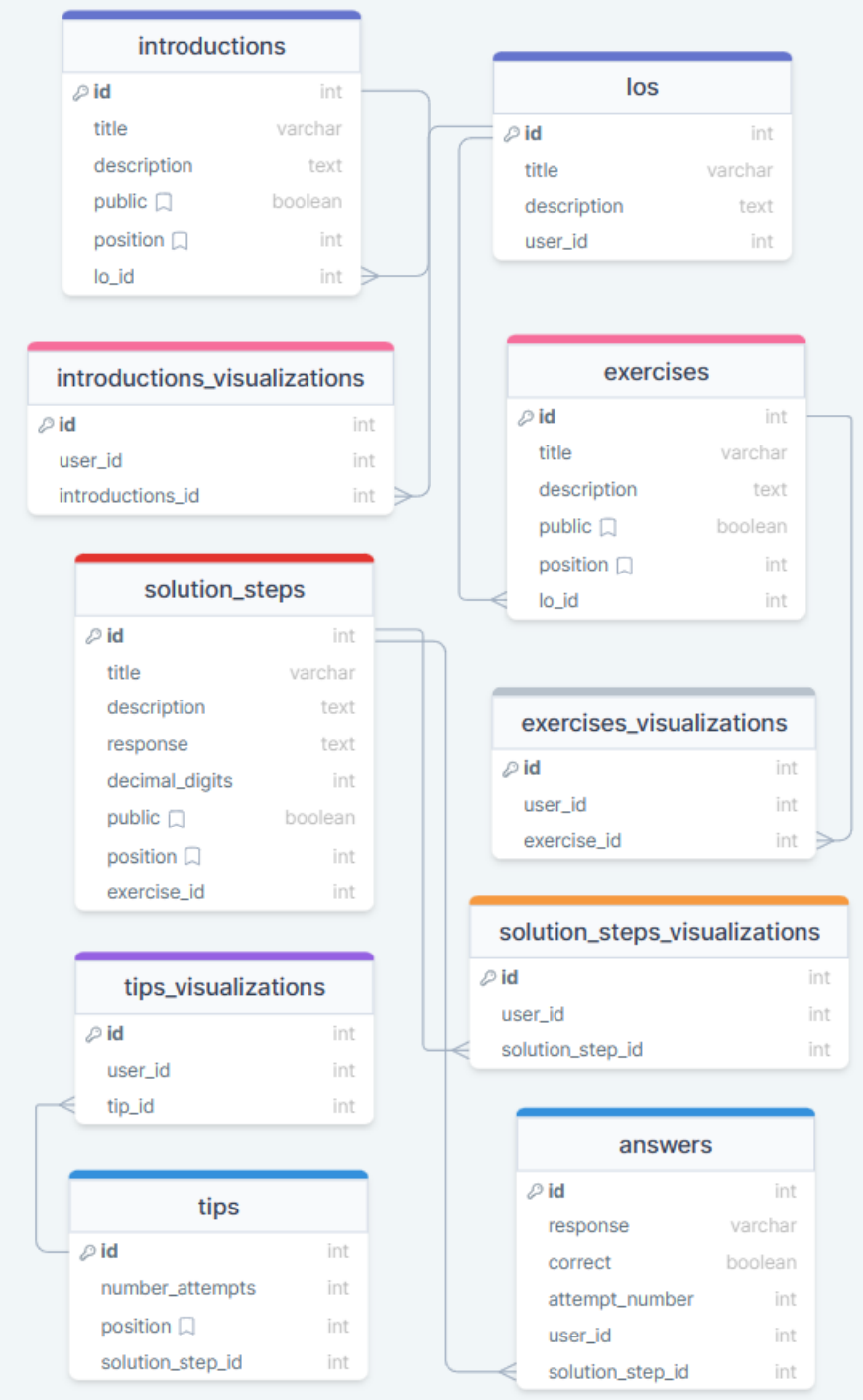


Figura 12 – Modelo do banco de dados para visualização e interação de objetos de aprendizagem na FARMA

Fonte: Autoria prórpia.

`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. Na Seção 4.2, será apresentada uma explicação mais detalhada de como essa modelagem resolve o problema proposto.

4 DESENVOLVIMENTO DA FARMA API

Este capítulo detalha o processo de desenvolvimento da API responsável pela visualização e interação com os objetos de aprendizagem na FARMA. Aqui, serão apresentados os principais aspectos do desenvolvimento, desde a criação até a finalização da API, conforme procedimentos delineados na Seção 2.2.

4.1 Backlog

A seguir, um resumo da sequência de execução das atividades, definida a partir dos requisitos funcionais descritos na Seção 3.2, bem como o comportamento esperado da rota:

1. Criar rota de detalhe de objeto de aprendizagem para aluno sem percentuais;
2. Criar rota de detalhe de objeto de aprendizagem para professor sem percentuais;
3. Criar rota de detalhe de objeto de aprendizagem para visitante sem percentuais;
4. Adicionar lógica de progresso para as rotas do aluno em uma turma;
5. Adicionar lógica de progresso para as rotas do professor;
6. Adicionar lógica de progresso para as rotas do visitante;
7. Criar rota de detalhe de página de um objeto de aprendizagem específico para aluno;
8. Criar rota de detalhe de página de um objeto de aprendizagem específico para professor;
9. Criar rota de detalhe de página de um objeto de aprendizagem específico para visitante;
10. Criar rota de confirmação de visualização no passo de solução pelo aluno;
11. Criar rota de confirmação de visualização no passo de solução pelo professor;
12. Criar rota de confirmação de visualização no passo de solução pelo visitante;
13. Criar a rota de envio de resposta para passo de solução pelo aluno;
14. Criar a rota de envio de resposta para passo de solução pelo professor;
15. Criar a rota de envio de resposta para passo de solução pelo visitante;
16. Criar rota de solicitação de dicas pelo aluno em uma turma;
17. Criar rota de solicitação de dicas pelo professor;

18. Criar rota de solicitação de dicas pelo visitante;

19. Criar `cron` para limpar interações de usuários: professor e visitante.

Foram desenvolvidas rotas para usuários autenticados (aluno e professor) e rotas públicas para usuário vistante. O professor pode obter os dados apenas dos objetos de aprendizagem que criou e assim como o usuário visitante não tem as informações de interação armazenadas.

4.2 Implementação das Rotas da API

A seguir são apresentadas as rotas de API necessárias para atender aos requisitos. As rotas utilizam o prefixo `view`, para indicar que fazem parte do recurso de visualização do objeto de aprendizagem.

4.2.1 Rota de detalhe do objeto de aprendizagem para professores, alunos e visitantes

Esta subseção é resultado da realização dos itens 1, 2 e 3 do *backlog* apresentados na Seção 4.1. Para a visualização do detalhe do objeto de aprendizagem foram desenvolvidas três rotas, sendo uma rota para cada papel de usuário[1]:

- Rota 1: rota de detalhe do objeto de aprendizagem para para visualização do aluno - `GET: view/teams/:team_id/los/:id`
- Rota 2: rota de detalhe do objeto de aprendizagem para visualização do professor - `GET: view/professors/los/:id`
- Rota 3: rota de detalhe do objeto de aprendizagem para visualização do visitante - `GET: view/guests/los/:id`

O dados de retorno esperados para as rotas de detalhe do objeto de aprendizagem são:

- Para aluno, professor e visitante:

    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 estado (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.

[1] Um mesmo usuário pode exercer o papel de aluno, professor ou visitante.

- Complemento:
    - As informações referente aos percentuais e ao estado, foram dados fictícios nesse primeiro momento;
    - No caso de professor, somente deve ser retornado o objeto de aprendizagem se o mesmo tiver sido criado por ele;
    - Configurar modelos `Team`, `UsersTeam` e `LosTeam` necessários para configuração das turmas.

Para a definição do contrato de retorno das rotas, visto que não seguem a mesma estrutura dos modelos da aplicação, foi criado um `Resource`, que é uma classe que define as propriedades e faz o mapeamento do `model` para o que deve ser retornado pela API.

A Rota 1 verifica se o usuário logado como aluno está matriculado na turma informada e se a turma possui o objeto de aprendizagem incluído. Neste trabalho, inicialmente essa rota foi desenvolvida retornando os percentuais do aluno com valores fixos, para fins de simulação e teste. Conforme Figura 13, para esta rota foi definido: um teste para o `controller`, garantindo que o retorno é o esperado em caso do aluno matriculado na turma com o objeto de aprendizagem incluído; um teste para o caso de erro, quando o aluno não está matriculado na turma ou o objeto de aprendizagem não está incluso na turma em que o aluno está matriculado. Após os ajustes necessários e testes, foi aberto um PR para criação da rota, no GitHub conforme a Tabela 2.

```
● → farma-api git:(@dviniciusbonin/64/feat/add-lo-detail) ./run test test/controllers/lo_controller_test.rb
Running 2 tests in a single process (parallelization threshold is 50)
Run options: --seed 53897

# Running:

..

Finished in 0.731541s, 2.7340 runs/s, 5.4679 assertions/s.
2 runs, 4 assertions, 0 failures, 0 errors, 0 skips
```

Figura 13 – Execução de testes - PR 2
Fonte: Autoria própria.

A Rota 2 verifica se o usuário logado como professor é o criador do objeto de aprendizagem. Para essa rota foi desenvolvido um teste para o `controller` que garante que o retorno é o esperado em caso do professor ser o criador do objeto de aprendizagem; um teste para o caso de erro, em que o professor tenta acessar um objeto de aprendizagem que não foi criado pelo mesmo, conforme a execução na Figura 14.

Para os testes do modelo do usuário, foram realizados ajustes para garantir que o relacionamentos entre as tabelas criadas no banco de dados, conforme explicado anteriormente, estivessem funcionando após a criação da estrutura para turmas. Após isso foi aberto um PR, no GitHub conforme a Tabela 3.

Title	Add lo detail route
Issue	#64
Description	Add lo detail route without percentages
Motivation and Context	Allow viewing of a learning object
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [x] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	3

Tabela 2 – PR - Rota de detalhe do objeto aprendizagem para o aluno (Rota 1).

```
● → farma-api git:(@dviniciusbonin/67/feat/add-lo-detail-for-professor) ./run test \
    test/controllers/api/view/professors/los_controller_test.rb \
    test/models/lo_test.rb \
    test/models/user_test.rb

  Running 24 tests in a single process (parallelization threshold is 50)
  Run options: --seed 58563

  # Running:

  ........................

  Finished in 2.306297s, 10.4063 runs/s, 17.3438 assertions/s.
  24 runs, 40 assertions, 0 failures, 0 errors, 0 skips

  Task completed in 0m5.881s
```

Figura 14 – Execução de testes - PR 3
Fonte: Autoria própria.

A Rota 3 é uma rota pública, sendo assim diferentemente das rotas de detalhe do objeto de aprendizagem para o aluno em uma turma e para o professor, para essa rota não é necessário autenticação, dispensando a passagem do `token` no cabeçalho da requisição. Essa rota apenas verifica se existe o objeto de aprendizagem, caso exista retorna os dados para visualização, senão retorna que o objeto não foi encontrado. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso do professor ser o criador do objeto de aprendizagem. E um teste para o caso de erro, em que o visitante tenta acessar um objeto de aprendizagem que não existe, conforme é apresentado a execução na Figura 15. Após isso foi aberto um PR, no GitHub conforme a Tabela 4.

Conforme dito anteriormente, de início as rotas foram implementadas com os percentuais retornando valores fixos. Após o desenvolvimento inicial das rotas 1, 2 e 3, foram incluídos os percentuais como resultados dos itens do *backlog*: 4, 5 e 6, isto é adicionando a lógica de progresso para as rotas do aluno em uma turma, professor e visitante.

Title	Lo detail endpoint for professor
Issue	#67
Description	Lo detail for professor Add teams model Refactoring in lo detail for student
Motivation and Context	Allow viewing of a learning object by professor
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature - [x] refactor: A code change that neither fixes a bug nor adds a feature
Checklist	- [x] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	3

Tabela 3 – PR - Rota de detalhe do objeto aprendizagem para o professor.

```
● → farma-api git:(@dviniciusbonin/69/feat/add-lo-detail-for-guest) ./run test \
   test/controllers/api/views/guests/lo_guest_controller_test.rb

Running 2 tests in a single process (parallelization threshold is 50)
Run options: --seed 2988

# Running:

..

Finished in 0.662361s, 3.0195 runs/s, 6.0390 assertions/s.
2 runs, 4 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m7.492s
```

Figura 15 – Execução de testes - PR 4
Fonte: Autoria própria.

4.2.2 Rota de detalhe de página de um objeto de aprendizagem específico para professores, alunos e visitantes

A implementação destas rotas é resultado da realização dos itens do *backlog* apresentados nas seções 7, 8 e 9. Para o detalhe de página de um objeto de aprendizagem específico foram desenvolvidas três rotas, sendo uma para cada papel de usuário, onde cada rota tem como último parâmetro a página solicitada (página de introdução ou de exercício):

- Rota 4: rota de detalhe de página de um objeto de aprendizagem específico para visualização do aluno - `GET: view/teams/:team_id/los/:id/page/:page`
- Rota 5: rota de detalhe de página de um objeto de aprendizagem específico para visualização do professor - `GET: view/professors/los/:id/page/:page`
- Rota 6: rota de detalhe de página de um objeto de aprendizagem específico para visualização do visitante - `GET: view/guests/los/:id/page/:page`

Title	Lo detail endpoint for guest
Issue	#69
Description	Lo detail for guest
Motivation and Context	Allow viewing of a learning object by guest
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	3

Tabela 4 – PR - Rota de detalhe do objeto aprendizagem para o visitante.

O dados de retorno esperados para as rotas de detalhe do objeto de aprendizagem são:

1. Se página de introdução (para aluno, professor ou visitante):
    a) Título;
    b) Conteúdo que pode conter uma descrição, imagens e vídeo;
2. Se página de exercício:
    - Para aluno:
        a) Título;
        b) Conteúdo que pode conter uma descrição, imagens e vídeo;
        c) Passos de solução, onde cada um contém:
            i. Título;
            ii. Conteúdo que pode conter uma descrição, imagens e vídeo;
            iii. Quantidade de tentativas de resposta;
            iv. Status (acerto, erro, visualizado ou não Visualizado);
    - Para professor e visitante:
        a) Título;
        b) Conteúdo que pode conter uma descrição, imagens e vídeo;
        c) Passos de solução, onde cada um contém:
            i. Título;
            ii. Conteúdo que pode conter uma descrição, imagens e vídeo;
    - Complemento:
        - Para o usuário visitante as informações de status dos passos de solução do exercício (acerto, erro, visualizado ou não Visualizado) e quantidade de tentativas de resposta serão armazenados temporariamente.

A Rota 4 verifica se o usuário logado como aluno está matriculado na turma informada, se a turma possui o objeto de aprendizagem incluído e se o objeto de aprendizagem possui a página solicitada. Para o desenvolvimento dessa rota foi criado uma classe `Resource` para cada tipo de página, visto que um objeto de aprendizagem pode ter uma página de introdução e uma página de exercício, onde cada página tem suas respectivas propriedades.

Para essa rota foi, desenvolvida um teste para o `controller`, garantindo que o retorno é o esperado em caso do aluno matriculado na turma com o objeto de aprendizagem incluído com a página solicitada. E um teste para o caso de erro, quando o aluno não está matriculado na turma ou o objeto de aprendizagem não está incluso na turma em que o aluno está matriculado ou a página não existe para o objeto de aprendizagem, conforme é apresentada a execução na Figura 16.

```
● → farma-api git:(@dviniciusbonin/72/feat/add-page-detail-for-student) ✗ ./run test \
test/controllers/api/view/teams/page_controller_test.rb
Running 4 tests in a single process (parallelization threshold is 50)
Run options: --seed 38510

# Running:

....

Finished in 1.428211s, 2.8007 runs/s, 5.6014 assertions/s.
4 runs, 8 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m11.474s
```

Figura 16 – Execução de testes - PR 5
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 5.

Title	Lo Page detail endpoint for student
Issue	#72
Description	Lo Page detail for student
Motivation and Context	Allow the student to view a page of a learning object
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	3

Tabela 5 – PR - Rota de detalhe de uma página especifica do objeto de aprendizagem para o aluno.

Nesse PR, foram feitas algumas refatorações pelo professor orientador, como a criação de uma classe `PageResource` para página que encapsula a verificação do tipo da página conforme o Código 4.1.

```ruby
class PageResource
  def self.to(page)
    if page.instance_of?(Introduction)
      IntroductionPageResource.new(page)
    else
      ExercisePageResource.new(page)
    end
  end
end
```

Listagem 4.1 – Classe PageResource

Também foi criada uma classe `ResourcesByCurrentUserTeams` do tipo `concern`, que no *Rails*, funciona como um módulo para agrupar a lógica para buscar a turma do aluno, deixando assim o código do `controller` mais limpo conforme o Código 4.2.

O último ajuste foi a adaptação do método que retorna o nome do modelo quando o recusando não é encontrado. Anteriormente só funcionava para modelo, visto que a página não tem um modelo em si, foi necessário esse ajuste para funcionar para a página também, conforme o Código 4.3.

```ruby
module ResourcesByCurrentUserTeams
  extend ActiveSupport::Concern

  private

  def team(id)
    current_user.teams.find(id)
  end

  def lo(team_id, lo_id)
    team = team(team_id)
    team.los.find(lo_id)
  end
end
```

Listagem 4.2 – Módulo ResourcesByCurrentUserTeams

```ruby
def model_human(model)
   model = model.constantize if model.is_a?(String) && Object.
  const_defined?(model)
   return model.model_name.human if model.respond_to?(:model_name)

   model ||= controller_name.singularize.downcase
   I18n.t("activerecord.models.#{model}", count: 1)
 end
```

Listagem 4.3 – Método model_human

A Rota 5 verifica se o usuário logado como professor é o criador do objeto de aprendizagem informado e se o objeto de aprendizagem possui a página solicitada. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso do professor ser o criador do objeto de aprendizagem e existir a página solicitada e um teste para o caso de erro, quando o professor não é o criador ou não existe o objeto de aprendizagem ou a página não existe para o objeto de aprendizagem, conforme é apresentado a execução na Figura 17. Após isso foi aberto um PR, no GitHub conforme a Tabela 6.

```
→  farma-api git:(@dviniciusbonin/73/feat/add-page-detail-for-professor) ./run test \
test/controllers/api/view/professors/professor_page_controller_test.rb
Running 4 tests in a single process (parallelization threshold is 50)
Run options: --seed 63182

# Running:

....

Finished in 3.124154s, 1.2803 runs/s, 2.5607 assertions/s.
4 runs, 8 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m11.665s
```

Figura 17 – Execução de testes - PR 6
Fonte: Autoria própria.

Nesse PR, foram solicitados alguns ajustes pelo professor orientador. Foi realizada a implementação do comando `populate`, para que o ambiente local tenha os dados necessários para os testes manuais. Foram refatorados os `resources` das página adotando uso de herança e polimorfismo conforme os Códigos 4.4, 4.5 e 4.6.

Title	Lo Page detail endpoint for professor
Issue	#73
Description	Lo Page detail for professor
Motivation and Context	Allow the professor to view a page of a learning object
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	4

Tabela 6 – PR - Rota de detalhe de uma página especifica do objeto de aprendizagem para o professor.

```
class PageResource
  attr_reader :type, :title, :position, :description

  def initialize(page)
    @type = page.class.name
    @title = page.title
    @position = page.position
    @description = page.description
  end
end
```

Listagem 4.4 – Classe PageResource Refatorada

```
class IntroductionPageResource < PageResource
  attr_reader :status

  def initialize(introduction)
    super(introduction)
    @status = :viewed
  end
end
```

Listagem 4.5 – Classe IntroductionPageResource Refatorada

```ruby
class ExercisePageResource < PageResource
  attr_reader :solution_steps, :status

  def initialize(exercise)
    super(exercise)
    @status = :not_viewed
    @solution_steps = exercise.solution_steps.map do |solution_step|
      solution_step_data(solution_step)
  @@ -17,6 +14,8 @@ def initialize(exercise)
  def solution_step_data(solution_step)
    {
      position: solution_step.position,
      title: solution_step.title,
      description: solution_step.description,
      status: :viewed, # TODO: This is fake data for now
      attempts: 6
    }
  end
end
```

Listagem 4.6 – Classes Resources

E por fim, foram escritos testes para os `models`, conforme a execução na Figura 18.

```
● → farma-api git:(@dviniciusbonin/73/feat/add-page-detail-for-professor) ./run test \
  test/models/lo_test.rb \
  test/models/los_team_test.rb \
  test/models/users_team_test.rb \
  test/models/team_test.rb \
  test/models/logics/lo/page_test.rb
  Running 27 tests in a single process (parallelization threshold is 50)
  Run options: --seed 43959

  # Running:

  ...........................

  Finished in 2.801476s, 9.6378 runs/s, 14.9921 assertions/s.
  27 runs, 42 assertions, 0 failures, 0 errors, 0 skips

  Task completed in 0m7.306s
```

Figura 18 – Execução de testes para modelos - PR 6
Fonte: Autoria própria.

A Rota 6 verifica se o objeto de aprendizagem existe e se o objeto de aprendizagem possui a página solicitada. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso do objeto de aprendizagem existir, assim como a página solicitada também deve existir. E um teste para o caso de erro, quando o objeto de aprendizagem ou a página não existe, conforme é apresentado a execução na Figura 19.



Figura 19 – Execução de testes - PR 7
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 7.

Title	Lo Page detail endpoint for guest
Issue	#74
Description	Lo Page detail for guest
Motivation and Context	Allow the guest to view a page of a learning object
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	1

Tabela 7 – PR - Rota de detalhe de uma página especifica do objeto de aprendizagem para o visitante.

4.2.3 Rota de confirmação de visualização em um passo de solução para professores, alunos e visitantes

A implementação destas rotas é resultado da realização dos itens do *backlog* apresentados nas seções: 10, 11 e 12. Para a confirmação de visualização do passo de solução foram desenvolvidas três rotas, sendo uma para cada papel de usuário, onde cada rota tem como o último parâmetro o identificador do passo de solução.

- Rota 7: rota de confirmação de visualização em um passo de solução para o aluno em uma turma - `POST: view/teams/:team_id/los/:lo_id/exercises/:exercise_id/solution_steps/:solution_step_id/view`
- Rota 8: rota de confirmação de visualização em um passo de solução para o professor - `POST: view/professors/los/:id/exercises/:exercise_id/solution_steps/:solution_step_id/view`
- Rota 9: rota de confirmação de visualização em um passo de solução para o visitante - `POST: view/guests/los/:id/exercises/:exercise_id/solution_steps/:solution_step_id/view`

O dados de retorno esperados para as rotas de detalhe do objeto de aprendizagem são:

1. Para aluno, professor e visitante:
    - A requisição para essas rotas será feita a partir do evento do `scroll` do mouse, devendo retornar sucesso se registrada a visualização.
    - Complemento:
        - No caso de visualização para o aluno, foi configurado o `model SolutionStepsVisualization`, para representar uma visualização no passo de solução, contendo relação com o usuário, passo de solução e turma.

A Rota 7 verifica se o usuário logado como aluno está matriculado na turma informada, se a turma possui o objeto de aprendizagem incluído, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é verificado se o mesmo já foi visualizado pelo usuário, se sim é retornado a visualização, se não cria se um registro na tabela `solution_steps_visualizations`, para representar a visualização do passo de solução.

Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso de a turma, objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de aprendizagem, exercício ou passo de solução não existir. Também foi ajustado o teste do `model` do passo de solução, verificando se ele tem relação com `solution_steps_visualizations` e criado um teste para o `model` da visualização do passo de solução, garantindo que tem as relações com usuário, passo de solução e turma. A execução dos testes é apresentada conforme a Figura 20.

Após isso foi aberto um PR, no GitHub conforme a Tabela 8.

```
● → farma-api git:(@dviniciusbonin/78/feat/add-confirm-view-endpoint) ✗ ./run test \
    test/controllers/api/view/teams/solution_step_controller_test.rb \
    test/models/solution_step_test.rb \ test/models/solution_steps_visuzalization_test.rb
  Running 25 tests in a single process (parallelization threshold is 50)
  Run options: --seed 2652

  # Running:

  .........................

  Finished in 2.165708s, 11.5436 runs/s, 15.2375 assertions/s.
  25 runs, 33 assertions, 0 failures, 0 errors, 0 skips

  Task completed in 0m7.289s
```

Figura 20 – Execução de testes - PR 8
Fonte: Autoria própria.

Title	Confirm view endpoint
Issue	#78
Description	Confirm view in solution step
Motivation and Context	Allow the student to confirm the view in a solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	2

Tabela 8 – PR - Rota de confirmação de visualização no passo de solução pelo aluno em uma turma

A Rota 8 verifica se o usuário logado como professor é o criador do objeto de aprendizagem informado, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é verificado se o mesmo já foi visualizado pelo usuário, se sim é retornado a visualização, se não cria se um registro na tabela `solution_steps_visualizations`, para representar a visualização do passo de solução. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso de o objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando o objeto de aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 21.

Após isso foi aberto um PR, no GitHub conforme a Tabela 9.

A Rota 9 verifica se o objeto de aprendizagem informado existe, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução existe é verificado se o mesmo já foi visualizado pelo usuário, se sim é retornado a visualização, se não cria se um registro na tabela `solution_steps_visualizations`, para representar a visualização do passo de solução. Para esta rota foi desenvolvido um teste

```
● → farma-api git:(@dviniciusbonin/89/feat/add-confirm-view-endpoint-by-professor) ✗ ./run test \
test/controllers/api/view/professors/solution_step_controller_test.rb
Running 7 tests in a single process (parallelization threshold is 50)
Run options: --seed 59178

# Running:

.......

Finished in 2.836374s, 2.4679 runs/s, 3.5256 assertions/s.
7 runs, 10 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m11.910s
```

Figura 21 – Execução de testes - PR 9
Fonte: Autoria própria.

Title	Confirm view endpoint for professor
Issue	#89
Description	Confirm view in solution step
Motivation and Context	Allow the professor to confirm the view in a solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	2

Tabela 9 – PR - Rota de confirmação de visualização no passo de solução pelo professor

para o `controller`, garantindo que o retorno é o esperado em caso de o objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando o objeto de aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 22.

```
● → farma-api git:(@dviniciusbonin/98/feat/add-confirm-view-endpoint-by-guest) ✗ ./run test \
test/controllers/api/view/guests/solution_step_controller_test.rb

Running 3 tests in a single process (parallelization threshold is 50)
Run options: --seed 60641

# Running:

...

Finished in 2.664492s, 1.1259 runs/s, 1.8765 assertions/s.
3 runs, 5 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m12.075s
```

Figura 22 – Execução de testes - PR 10
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 10.

Title	Confirm view endpoint for guest
Issue	#98
Description	Confirm view in solution step
Motivation and Context	Allow the guest to confirm the view in a solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	1

Tabela 10 – PR - Rota de confirmação de visualização no passo de solução pelo visitante

4.2.4 Rota de envio de resposta para o passo de solução para professores, alunos e visitantes

A implementação destas rotas é resultado da realização dos itens do *backlog* apresentados nas seções: 13, 14 e 15. Para a o envio de resposta para o passo de solução foram desenvolvidas três rotas, sendo uma para cada papel de usuário, onde cada rota tem como o último parâmetro o identificador do passo de solução.

- Rota 10: rota de envio de resposta para o passo de solução para o aluno em uma turma - `POST: view/teams/:team_id/los/:lo_id/exercises/:exercise_id/solution_steps/:solution_step_id/answer`
- Rota 11: rota de envio de resposta para o passo de solução para o professor - `POST: view/professors/los/:lo_id/exercises/:exercise_id/solution_steps/:solution_step_id/answer`
- Rota 12: rota de envio de resposta para o passo de solução para o visitante - `POST: view/guests/los/:lo_id/exercises/:exercise_id/solution_steps/:solution_step_id/answer`

O dados de retorno esperados para as rotas de detalhe do objeto de aprendizagem são:

- Para aluno, professor e visitante
    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.
- Complemento:

- Configurar o modelo `Answer` para representar as resposta, contendo campos como resposta, número da tentativa, se é resposta correta e relação com o passo de solução.
- Configurar o modelo `TipsVisualization`, para representar a visualização da dica pelo estudante, contendo relação com usuário e dica.

A Rota 10 verifica se o usuário logado como aluno está matriculado na turma informada, se a turma possui o objeto de aprendizagem incluído, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é criado uma resposta, conforme o que foi enviado no corpo da requisição. Para o retorno foi implementado uma classe que mapeia os dados retorno, sendo: se a resposta esta correta, o histórico de respostas enviadas, as dicas já visualizadas e se existe alguma dica que não foi visualizada, conforme o Código 4.7.

```
class AnswerResource
  attr_reader :correct, :response_history, :tips_viewed, :
   tip_available

  def initialize(answer)
    @correct = answer.correct
    @response_history = answer.solution_step.answers.where(user:
   answer.user, team: answer.team)
    @tips_viewed = map_tips_viewed(answer.solution_step.tips, answer.
   user, answer.team)
    @tip_available = available_tip(answer.solution_step, answer.user,
    answer.team)
  end

  def map_tips_viewed(tips, user, team)
    tips.select do |tip|
      tip.visualizations.exists?(user: user, team: team)
    end
  end

  def available_tip(solution_step, user, team)
    solution_step.tips.each do |tip|
      return true unless tip.visualizations.exists?(user: user, team:
    team)
    end

    false
  end
end
```

Listagem 4.7 – Classe AnswerResource

Para essa rota foram configurados os `models` referentes as tabelas `answers` e `tips_visualizations`. Assim foram ajustados os testes para os `models`: `tip`, `solution_step` e criado testes para `tips_visualization` e `answer`. Foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso de a

turma, objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 23.

```
● → farma-api git:(@dviniciusbonin/80/feat/add-send-answer) ✗ ./run test \
test/controllers/api/view/teams/solution_step_controller_test.rb \
test/models/answer_test.rb \
test/models/solution_step_test.rb \
test/models/tip_test.rb \
test/models/tips_visualization_test.rb
Running 40 tests in a single process (parallelization threshold is 50)
Run options: --seed 15348

# Running:

........................................

Finished in 2.847018s, 14.0498 runs/s, 21.0747 assertions/s.
40 runs, 60 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m16.020s
```

Figura 23 – Execução de testes - PR 11
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 11.

Title	Add send answer endpoint for student
Issue	#80
Description	Send answer endpoint for student
Motivation and Context	Allow sending answer to solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	5

Tabela 11 – PR - Rota de envio de resposta ao passo de solução pelo aluno em uma turma

A Rota 11 verifica se o usuário logado como professor é o criador do objeto de aprendizagem informado, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é criado uma resposta, conforme o que foi enviado no corpo da requisição. Para o retorno foi implementado uma classe que mapeia os dados retorno, sendo: se a resposta esta correta, o histórico de respostas enviadas, as dicas já visualizadas e se existe alguma dica que não foi visualizada.

Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso o objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de

aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 24.

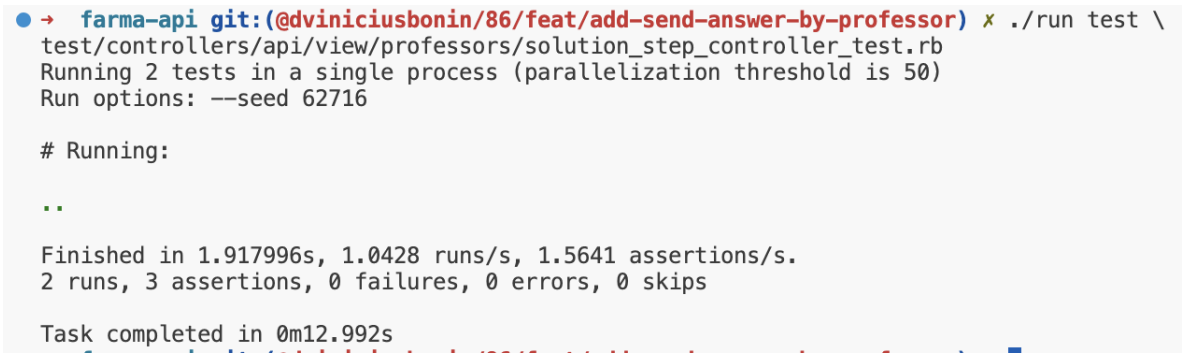


Figura 24 – Execução de testes - PR 12
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 12.

Title	Add send answer endpoint for professor
Issue	#86
Description	Send answer endpoint for professor
Motivation and Context	Allow sending answer to solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	3

Tabela 12 – PR - Rota de envio de resposta ao passo de solução pelo professor

A Rota 12 verifica se o objeto de aprendizagem informado existe, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é criada uma resposta, conforme o que foi enviado no corpo da requisição. Para o retorno foi implementado uma classe que mapeia os dados retorno, sendo: se a resposta esta correta, o histórico de respostas enviadas, as dicas já visualizadas e se existe alguma dica que não foi visualizada.

Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso o objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 25.

Após isso foi aberto um PR, no GitHub conforme a Tabela 13.

```
● → farma-api git:(@dviniciusbonin/96/feat/add-send-answer-by-guest) ✗ ./run test \
test/controllers/api/view/guests/solution_step_controller_test.rb
Running 5 tests in a single process (parallelization threshold is 50)
Run options: --seed 55542

# Running:

.....

Finished in 1.451891s, 3.4438 runs/s, 5.5101 assertions/s.
5 runs, 8 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m5.910s
```

Figura 25 – Execução de testes - PR 13
Fonte: Autoria própria.

Title	Add send answer endpoint for guest
Issue	#96
Description	Send answer endpoint for guest
Motivation and Context	Allow sending answer to solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	1

Tabela 13 – PR - Rota de envio de resposta ao passo de solução pelo visitante

4.2.5 Rota de solicitação de dicas para professores, alunos e visitantes

A implementação destas rotas é resultado da realização dos itens do *backlog* apresentados nas seções 16, 17 e 18. Para a solicitação de dicas para resolução do passo de solução foram desenvolvidas três rotas, sendo uma para cada papel de usuário, onde cada rota tem como o último parâmetro o identificador do passo de solução.

- Rota 13: rota de solicitação de dicas para o aluno em uma turma - `GET: view/teams/:team_id/los/:lo_id/exercises/:exercise_id/solution_steps/:solution_step_id/tips/request`
- Rota 14: rota de solicitação de dicas para o professor - `GET: view/professors/los/:id/exercises/:exercise_id/solution_steps/:solution_step_id/tips/request`
- Rota 15: rota de solicitação de dicas para o visitante - `GET: view/guests/los/:id/exercises/:exercise_id/solution_steps/:solution_step_id/tips/request`

O dados de retorno esperados para as rotas de detalhe do objeto de aprendizagem são:

- Pelo aluno em uma turma, pelo professor e pelo visitante:
    1. Uma dica, se ainda existir alguma disponível para visualização;
- Complemento:
    - A cada solicitação a dica é sinalizada como visualizada.

A Rota 13 verifica se o usuário logado como aluno está matriculado na turma informada, se a turma possui o objeto de aprendizagem incluído, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é verificado se o passo de solução possui alguma dica que não ainda não foi visualizada, seguindo a ordem prioridade na qual foi configurada a dica pelo criador. Se existir é adicionado uma registro na tabela `tips_visualizations`, para representar a visualização da dica. Para isso foram desenvolvidos 2 métodos no `model` da dica, um para verificar se a dica já foi visualizada e um para visualizar a dica. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso de a turma, objeto de aprendizagem existir, assim como o exercicio e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de aprendizagem, exercicio ou passo de solução não existir. Também foi ajustado o teste do `model` da dica, adicionando testes para os metodos `view` e `viewed`. A execução dos testes é apresentada conforme a Figura 26.

```
● → farma-api git:(@dviniciusbonin/82/feat/add-tips-request) ./run test \
test/controllers/api/view/teams/tip_controller_test.rb \
test/models/tip_test.rb
Running 10 tests in a single process (parallelization threshold is 50)
Run options: --seed 11229

# Running:

..........

Finished in 2.327578s, 4.2963 runs/s, 9.4519 assertions/s.
10 runs, 22 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m15.777s
```

Figura 26 – Execução de testes - PR 14
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 14.

A Rota 14 verifica se o usuário logado como professor é o criador do objeto de aprendizagem informado, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é verificado se o passo de solução possui alguma dica que não ainda não foi visualizada, seguindo a ordem prioridade na

Title	Tips request endpoint
Issue	#82
Description	Tips request endpoint for student
Motivation and Context	Allow the student to request tips to solve the solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	4

Tabela 14 – PR - Rota de solicitação de dicas para o aluno em uma turma

qual foi configurada a dica pelo criador. Se existir é adicionado uma registro na tabela `tips_visualizations`, para representar a visualização da dica. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso de a turma, objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 27.

```
● → farma-api git:(@dviniciusbonin/87/feat/add-tips-request-by-professor) ✗ ./run test \
test/controllers/api/view/professors/tip_controller_test.rb
Running 2 tests in a single process (parallelization threshold is 50)
Run options: --seed 44861

# Running:

..

Finished in 1.178360s, 1.6973 runs/s, 3.3945 assertions/s.
2 runs, 4 assertions, 0 failures, 0 errors, 0 skips

Task completed in 0m11.624s
```

Figura 27 – Execução de testes - PR 15
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 15.

A Rota 15 verifica se o objeto de aprendizagem informado existe, se o objeto de aprendizagem possui o exercício e se o exercício possui o passo de solução informado. Caso o passo de solução exista, é verificado se o passo de solução possui alguma dica que não ainda não foi visualizada, seguindo a ordem prioridade na qual foi configurada a dica pelo criador. Se existir é adicionado uma registro na tabela `tips_visualizations`, para representar a visualização da dica. Para esta rota foi desenvolvido um teste para o `controller`, garantindo que o retorno é o esperado em caso de a turma, objeto de aprendizagem existir, assim como o exercício e o passo de solução também deve existir. E um teste para o caso de erro, quando a turma, objeto de aprendizagem, exercício ou passo de solução não existir. A execução dos testes é apresentada conforme a Figura 28.

Title	Tips request endpoint for professor
Issue	#87
Description	Tips request endpoint for professor
Motivation and Context	Allow the professor to request tips to solve the solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	2

Tabela 15 – PR - Rota de solicitação de dicas para o professor



Figura 28 – Execução de testes - PR 16
Fonte: Autoria própria.

Após isso foi aberto um PR, no GitHub conforme a Tabela 16.

Title	Tips request endpoint for guest
Issue	#97
Description	Tips request endpoint for guest
Motivation and Context	Allow the guest to request tips to solve the solution step
How has this been tested?	This task was tested with the created unit tests and manually
Types of changes	- [x] feat: A new feature
Checklist	- [X] My code follows the code style of this project. - [ ] My change requires a change to the documentation. - [ ] I have updated the documentation accordingly.
Hours Required	1

Tabela 16 – PR - Rota de solicitação de dicas para o visitante

4.3 Rotina para limpeza de interações de usuários: professor e visitante

Após o desenvolvimento das rotas foi implementada uma rotina para ser executada diariamente, em um horário pŕe-determinado. Essa rotina realiza a limpeza dos registros referentes

às visualizações e interações dos usuários professor e visitante. Além disso, a rotina realiza a limpeza de todos os usuários visitantes. Essa rotina é necessária para manter apenas os dados do alunos em uma turma, para geração de estatísticas na FARMA. Essa implementação é referente ao Item 19 do *backlog*. Para a implementação foi utilizado a `gem whenever`. Após a instalação da `gem`, foi criado um arquivo `schedule.rb` conforme o Código 4.8 na pasta `config`, no qual é carregado as variáveis de ambiente, configurado a saída para os `logs` e configurado a rota para executar um `script` previamente configurado conforme o Código 4.9.

```
1 set :output, 'log/cron_log.log'
2 ENV.each { |k, v| env(k, v) }
3 set :environment, ENV.fetch('RAILS_ENV', nil)
4
5 every '0 3 * * *' do
6   rake 'clean:clean_data'
7 end
```

Listagem 4.8 – schedule.rb

```ruby
namespace :clean do
  desc 'Clean visualizations, answers, and guest users with null team'

  task clean_data: :environment do
    puts 'Cleaning introduction visualizations...'
    IntroductionsVisualization.where(team: nil).delete_all

    puts 'Cleaning exercise visualizations...'
    ExercisesVisualization.where(team: nil).delete_all

    puts 'Cleaning solution step visualizations...'
    SolutionStepsVisualization.where(team: nil).delete_all

    puts 'Cleaning tip visualizations...'
    TipsVisualization.where(team: nil).delete_all

    puts 'Cleaning answers...'
    Answer.where(team: nil).delete_all

    puts 'Cleaning guest users...'
    User.where(guest: true).delete_all

    puts 'Visualizations, answers, and guest users with null team cleaned successfully.'
  end
end
```

Listagem 4.9 – Script para limpeza de interações

5 RESULTADOS OBTIDOS

A seguir são apresentados os principais resultados obtidos com a finalização deste trabalho. Além da implementação das rotas da API também foram implementadas novas tabelas no banco de dados para atender aos requisitos.

5.1 Atualização do Banco de dados

Além dos recursos para visualização e interação com os objetos de aprendizagem, este trabalho teve como resultado a implementação da estrutura do banco de dados para as turmas. Na Figura 29, é apresentado o modelo de banco de dados (parcial) necessário para as turmas de objeto de aprendizagem e alunos na FARMA. A tabela `teams`, representa as turmas onde pode ter um nome, um código e o seu criador. A tabela `los_teams` representa os objetos de aprendizagem que a turma possui, assim como a tabela `users_teams` representa os alunos que a turma possui. Essa estrturua não será utiliza somente pela visualização e interação de objetos de aprendizagem, mas será utilizada por futuras implementações, como por exemplo: configuração de turmas pelo professor.

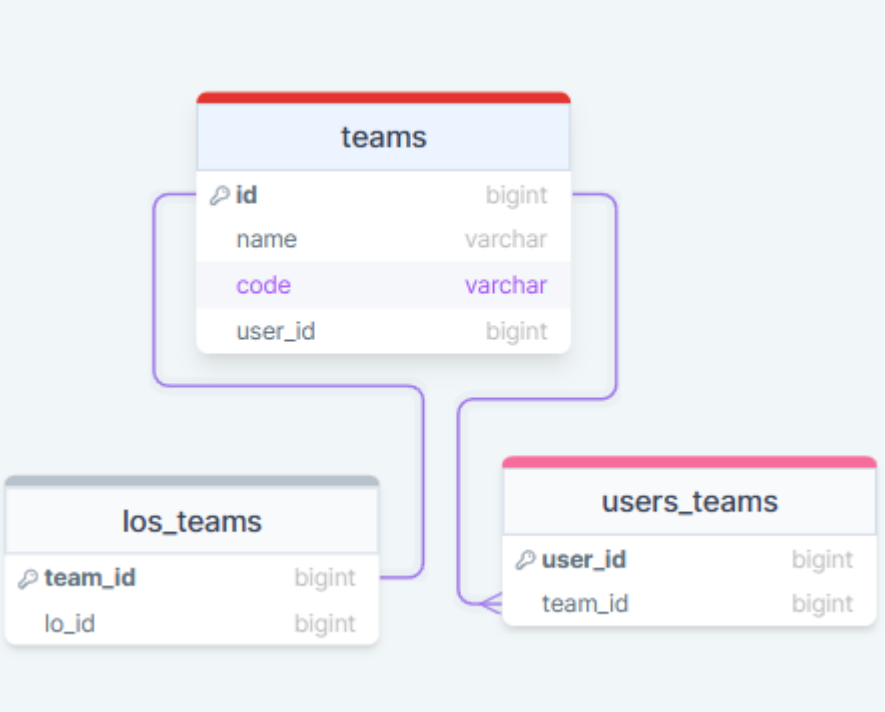


Figura 29 – Modelo do banco de dados para turmas de objetos de aprendizagem na FARMA
Fonte: Autoria própria.

5.2 Rotas da API

Os resultados obtidos com a implementação da API para visualização e interação com objetos de aprendizagem estão sumarizados na Tabela 17 a seguir:

Requisito	**Rota**	**Descrição**	**Exemplo de Visualização**
RF01	Rota 1	Visualização para aluno em uma turma	Figuras 30 e 31
	Rota 2	Visualização para o professor	Figuras 32 e 33
	Rota 3	Visualização para o visitante	Figuras 34 e 35
RF02, RF03, RF04	Rota 4	Visualização para aluno em uma turma	Figuras 36 e 37
	Rota 5	Visualização para o professor	Figuras 38 e 39
	Rota 6	Visualização para o visitante	Figuras 40 e 41
RF05	Rota 7	Rota para confirmação de visualização do passo de solução para o aluno em uma turma	Figuras 42 e 43
	Rota 8	Rota para confirmação de visualização do passo de solução para o professor	Figuras 44 e 45
	Rota 9	Rota para confirmação de visualização do passo de solução para o visitante	Figuras 46 e 47
RF06, RF07	Rota 10	Envio de resposta ao passo de solução para o aluno em uma turma	Figuras 48 e 49
	Rota 11	Envio de resposta ao passo de solução para o professor	Figuras 50 e 51
	Rota 12	Envio de resposta ao passo de solução para o visitante	Figuras 52 e 53
RF09	Rota 13	Solicitação de dicas para o aluno em uma turma	Figuras 54 e 55
	Rota 14	Solicitação de dicas para o professor	Figuras 56 e 57
	Rota 15	Solicitação de dicas para o visitante	Figuras 58 e 59

Tabela 17 – Resumo dos resultados obtidos a partir do requisitos funcionais

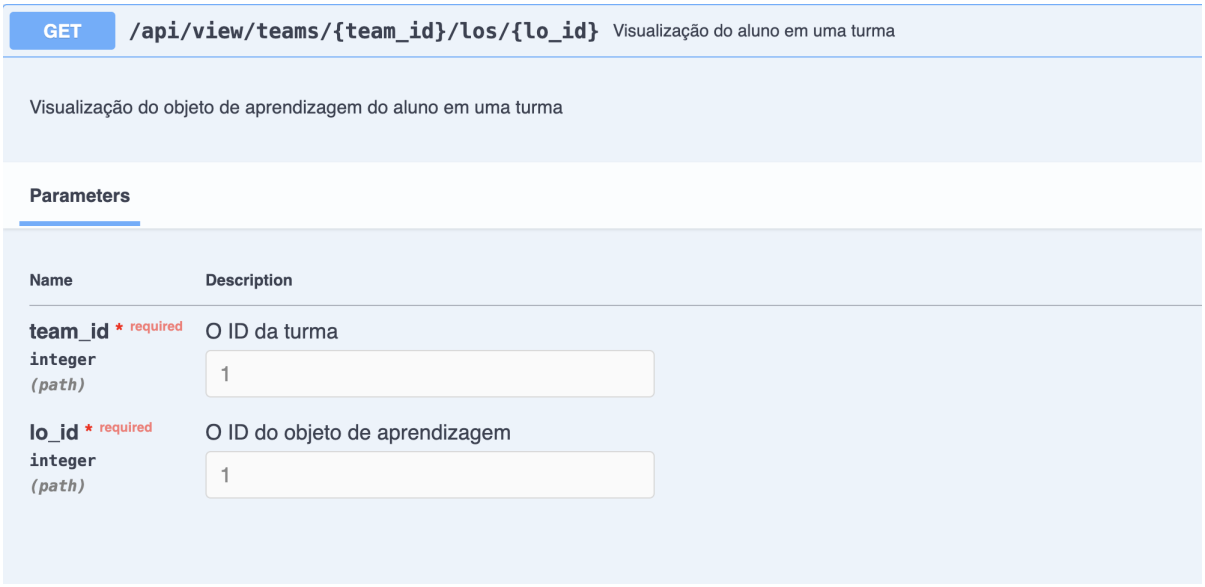


Figura 30 – Documentação da rota - PR 2
Fonte: Autoria própria.

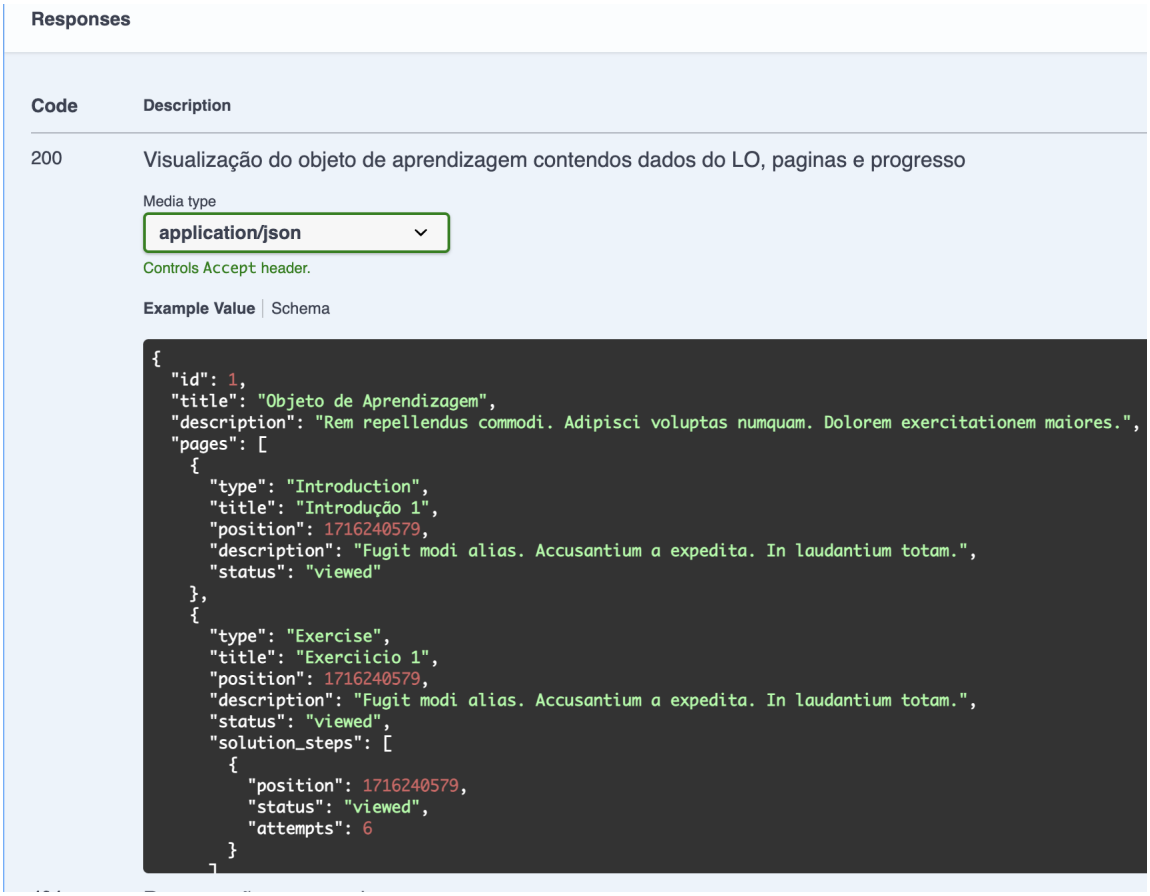


Figura 31 – Documentação da rota (resultado) - PR 2
Fonte: Autoria própria.

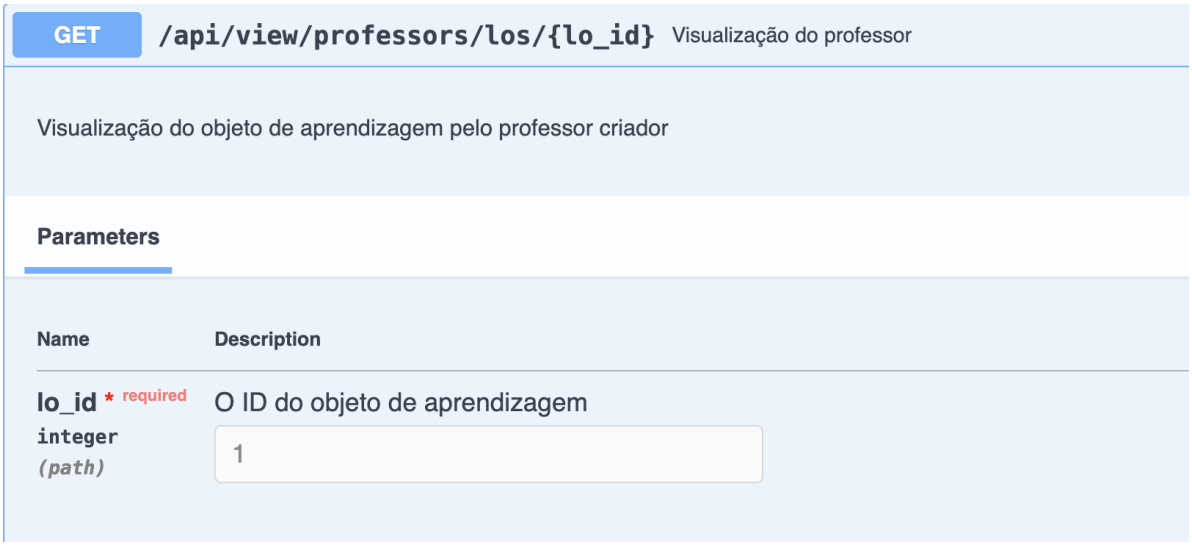


Figura 32 – Documentação da rota - PR 3
Fonte: Autoria própria.

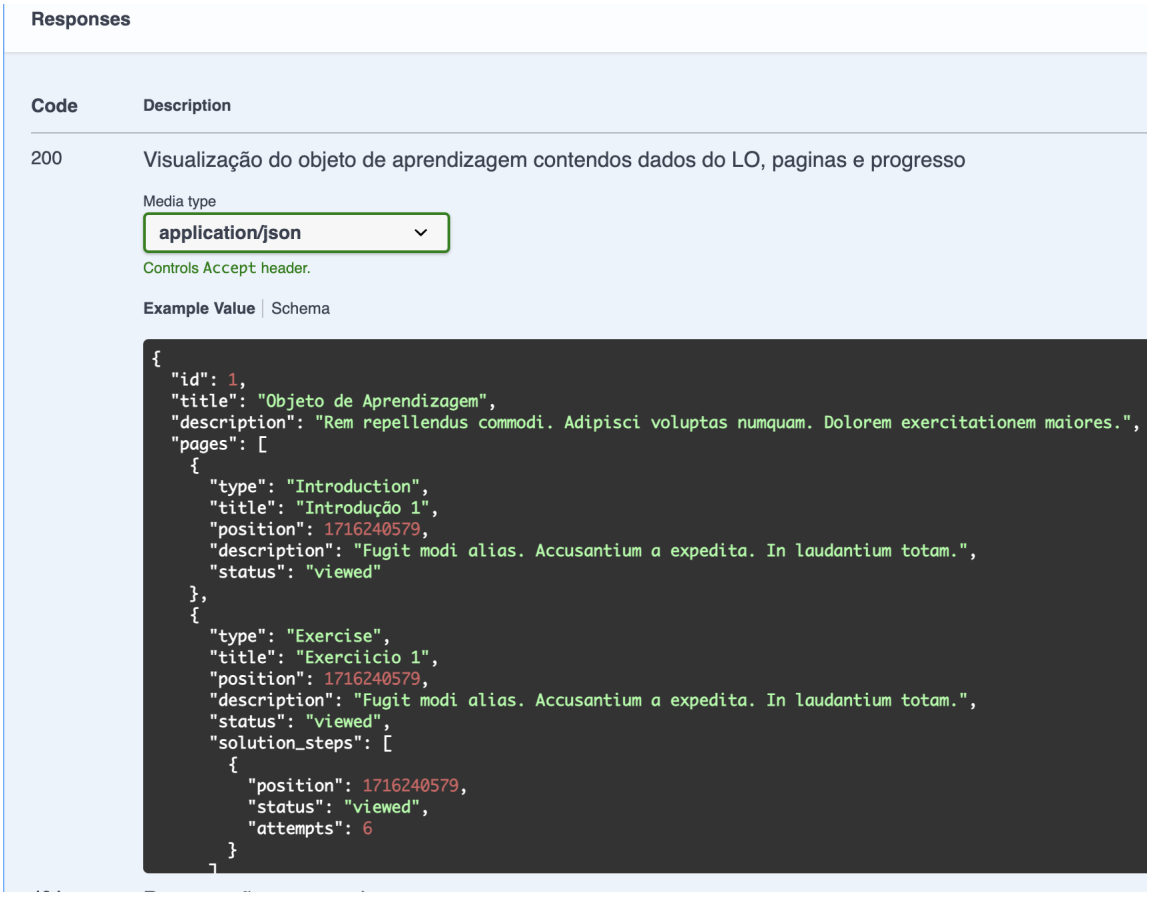


Figura 33 – Documentação da rota (resultado) - PR 3
Fonte: Autoria própria.

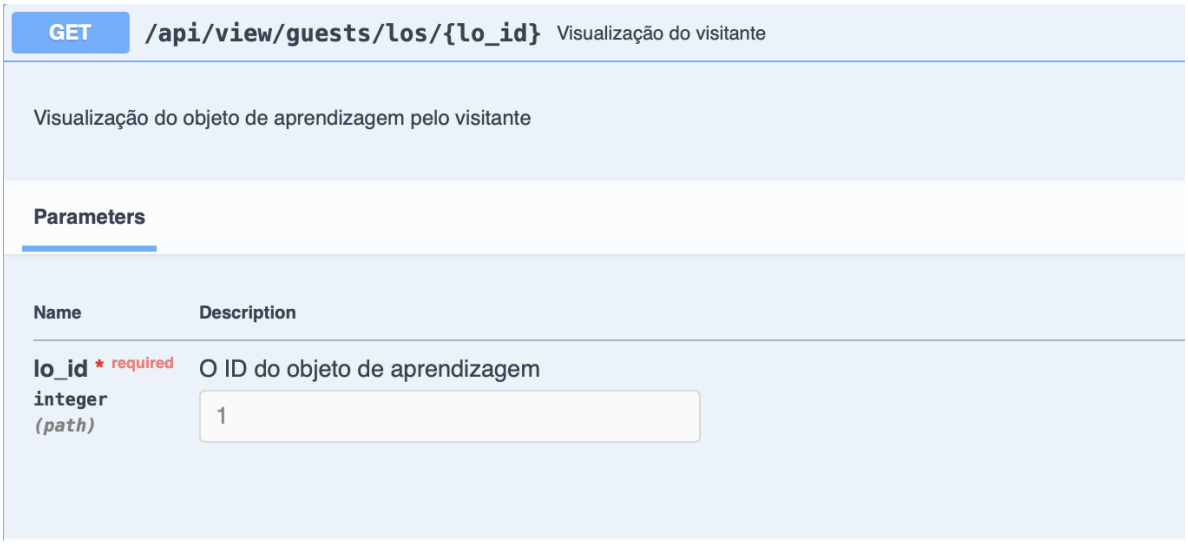


Figura 34 – Documentação da rota - PR 4
Fonte: Autoria própria.

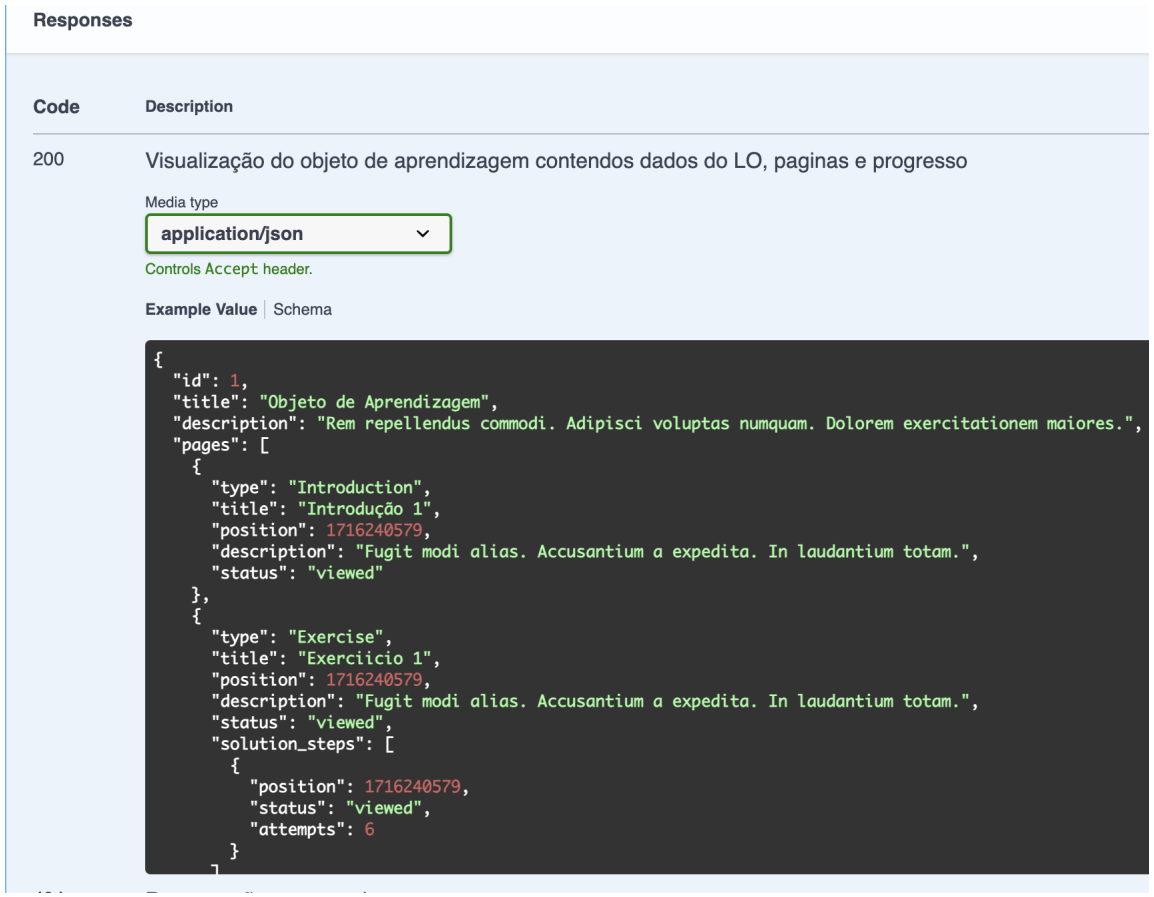


Figura 35 – Documentação da rota (resultado) - PR 4
Fonte: Autoria própria.

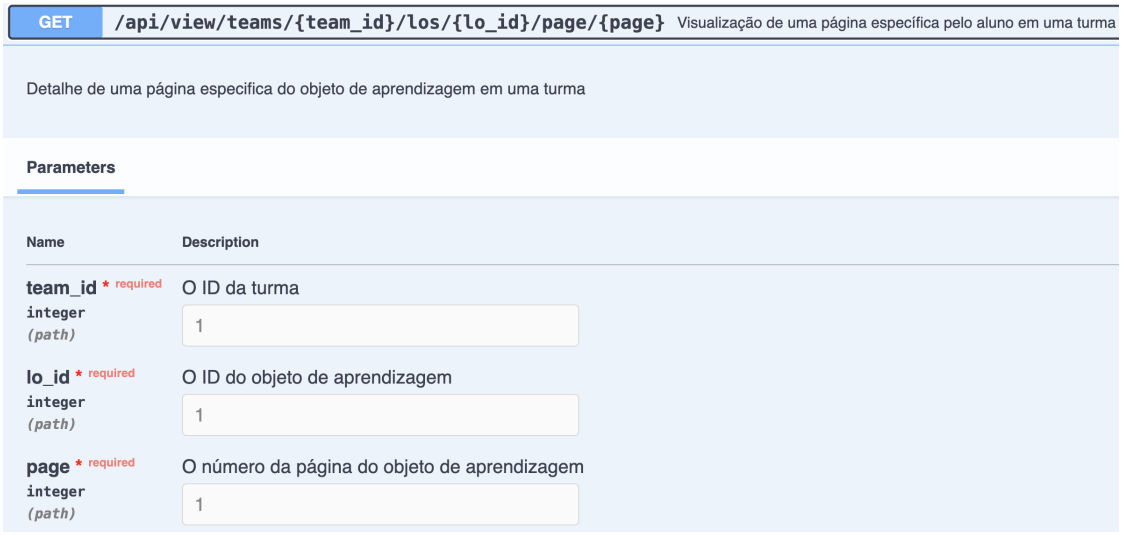


Figura 36 – Documentação da rota - PR 5
Fonte: Autoria própria.

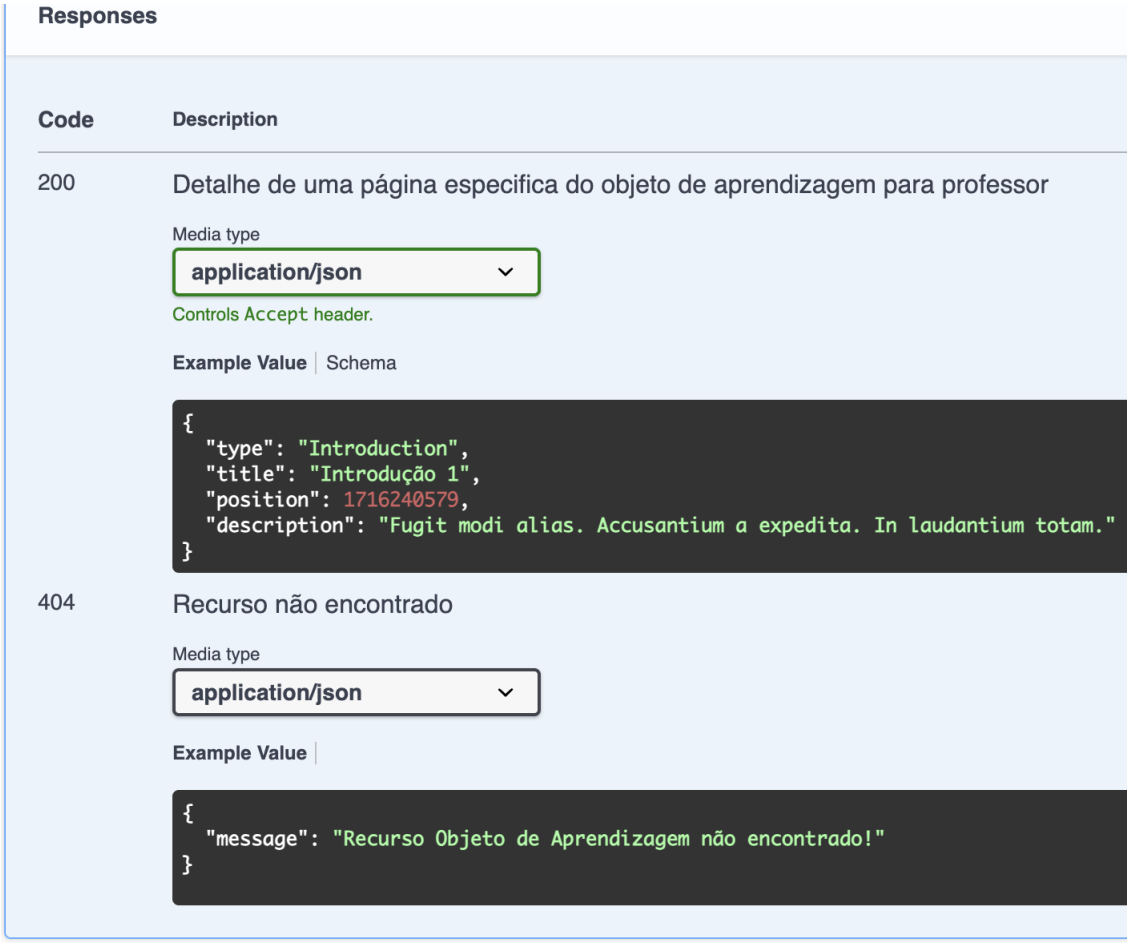


Figura 37 – Documentação da rota (resultado) - PR 5
Fonte: Autoria própria.

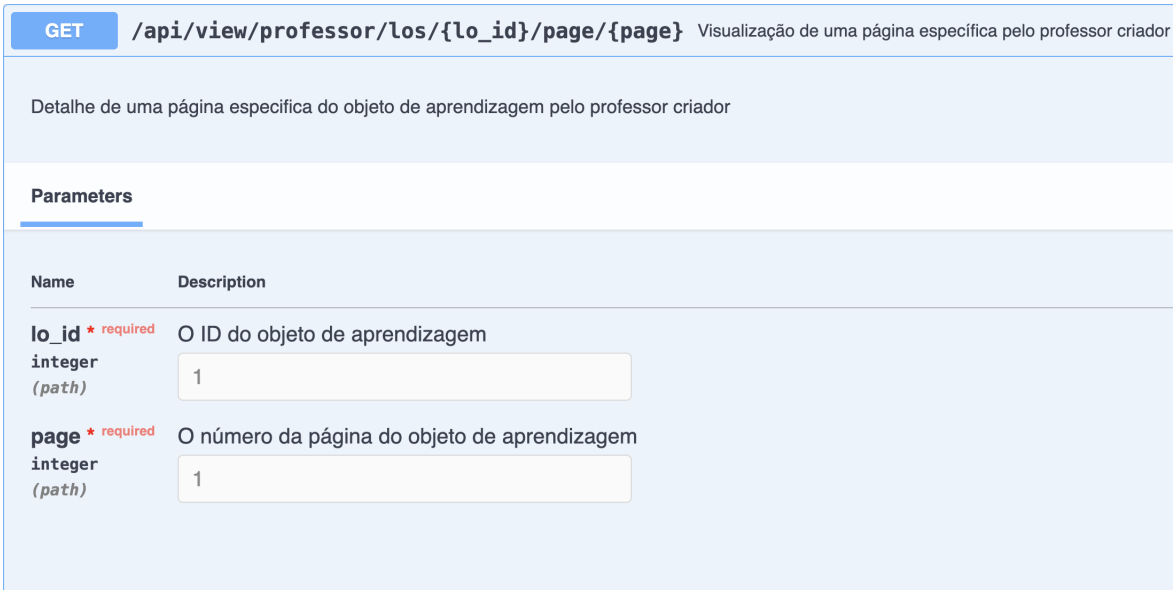


Figura 38 – Documentação da rota - PR 6
Fonte: Autoria própria.

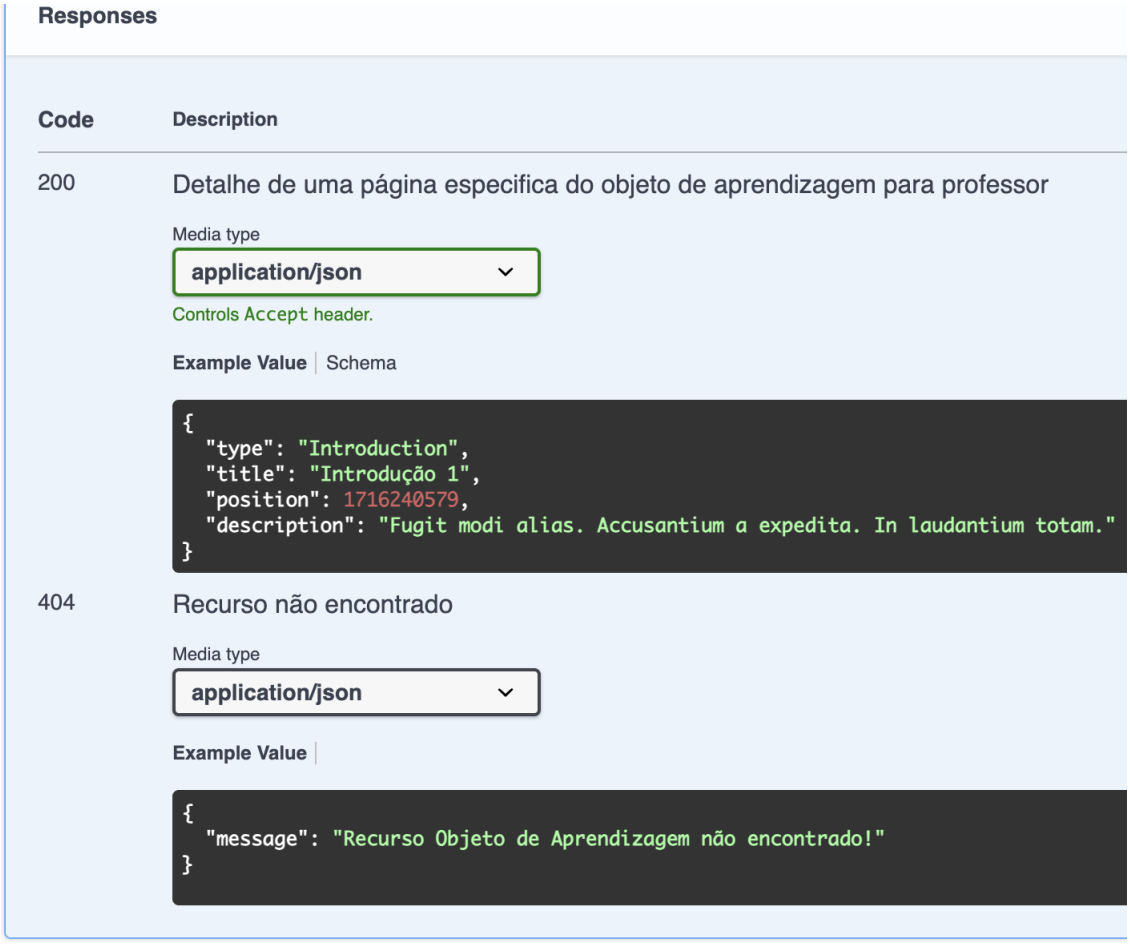


Figura 39 – Documentação da rota (resultado) - PR 6
Fonte: Autoria própria.

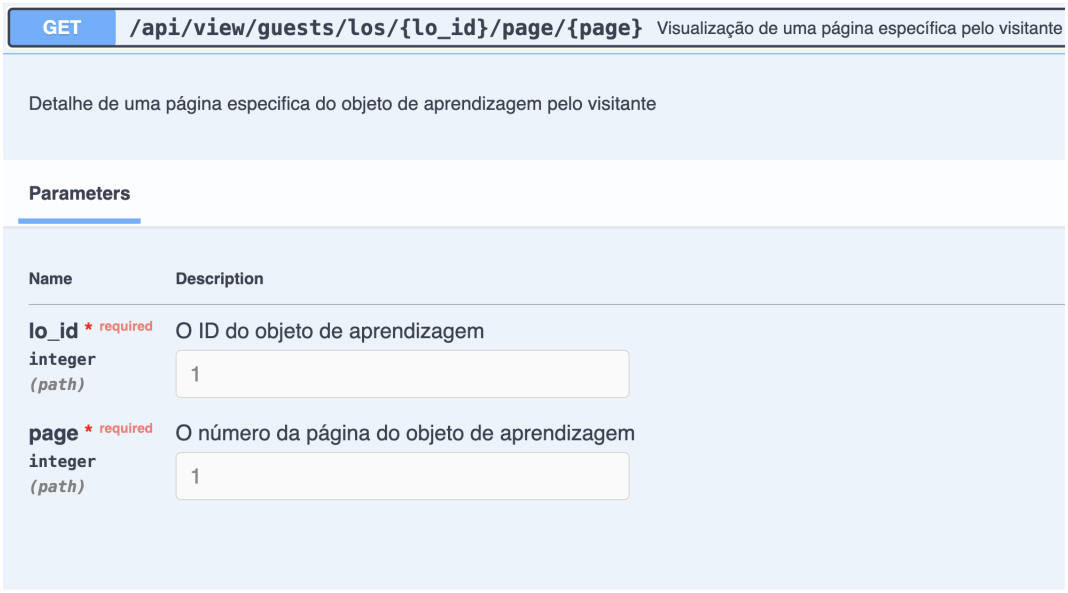


Figura 40 – Documentação da rota - PR 7
Fonte: Autoria própria.

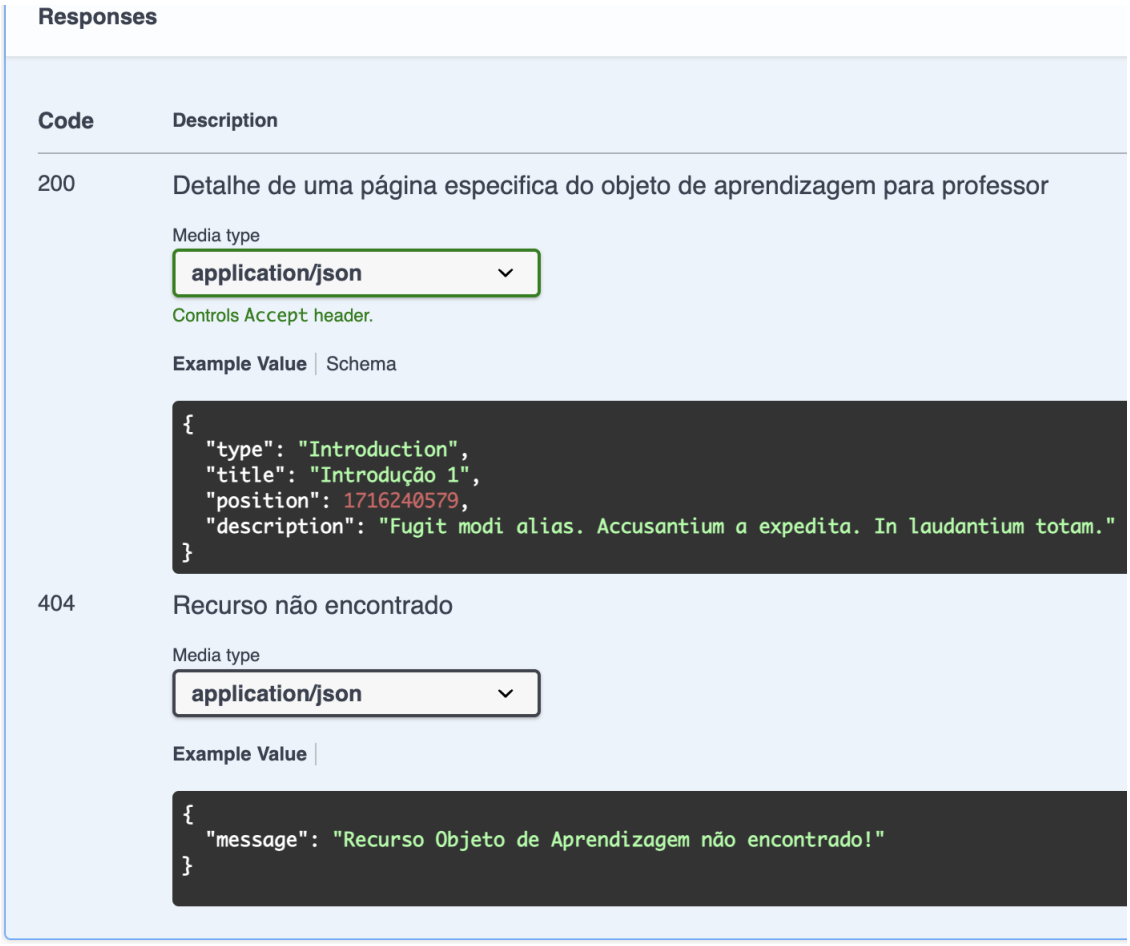


Figura 41 – Documentação da rota (resultado) - PR 7
Fonte: Autoria própria.

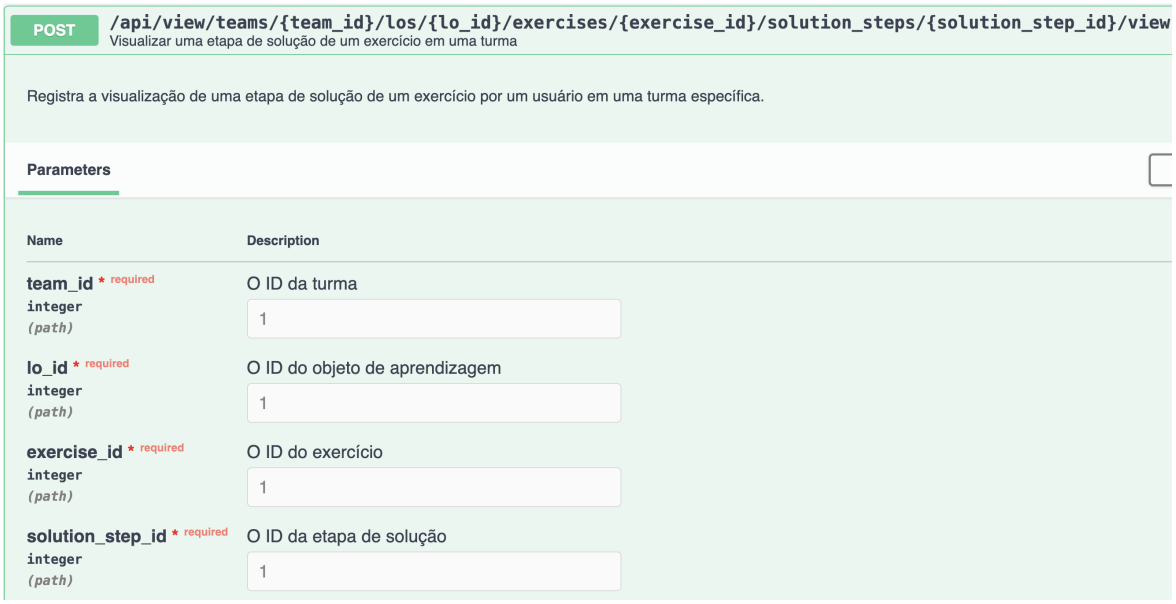


Figura 42 – Documentação da rota - PR 8
Fonte: Autoria própria.

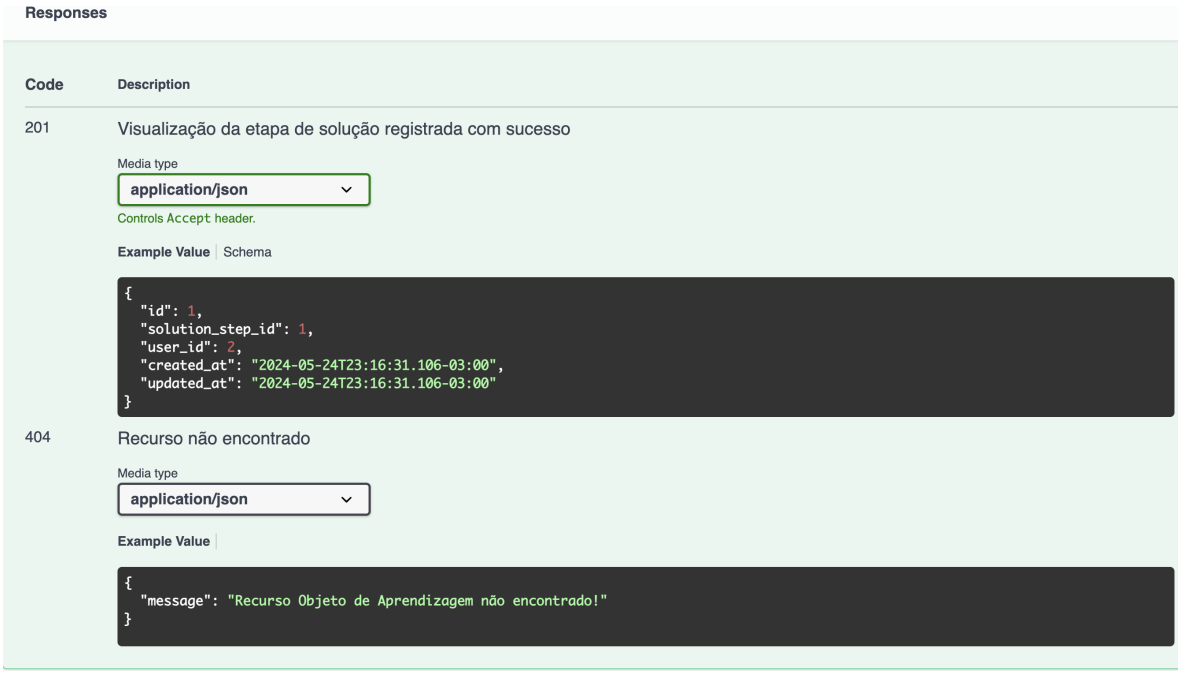


Figura 43 – Documentação da rota (resultado) - PR 8
Fonte: Autoria própria.

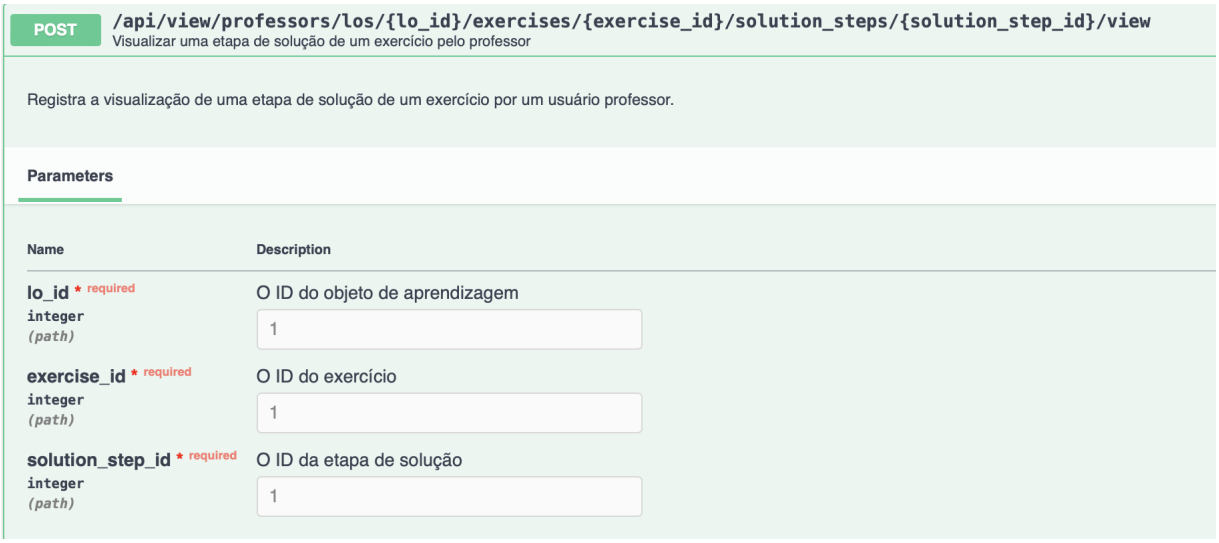


Figura 44 – Documentação da rota - PR 9
Fonte: Autoria própria.

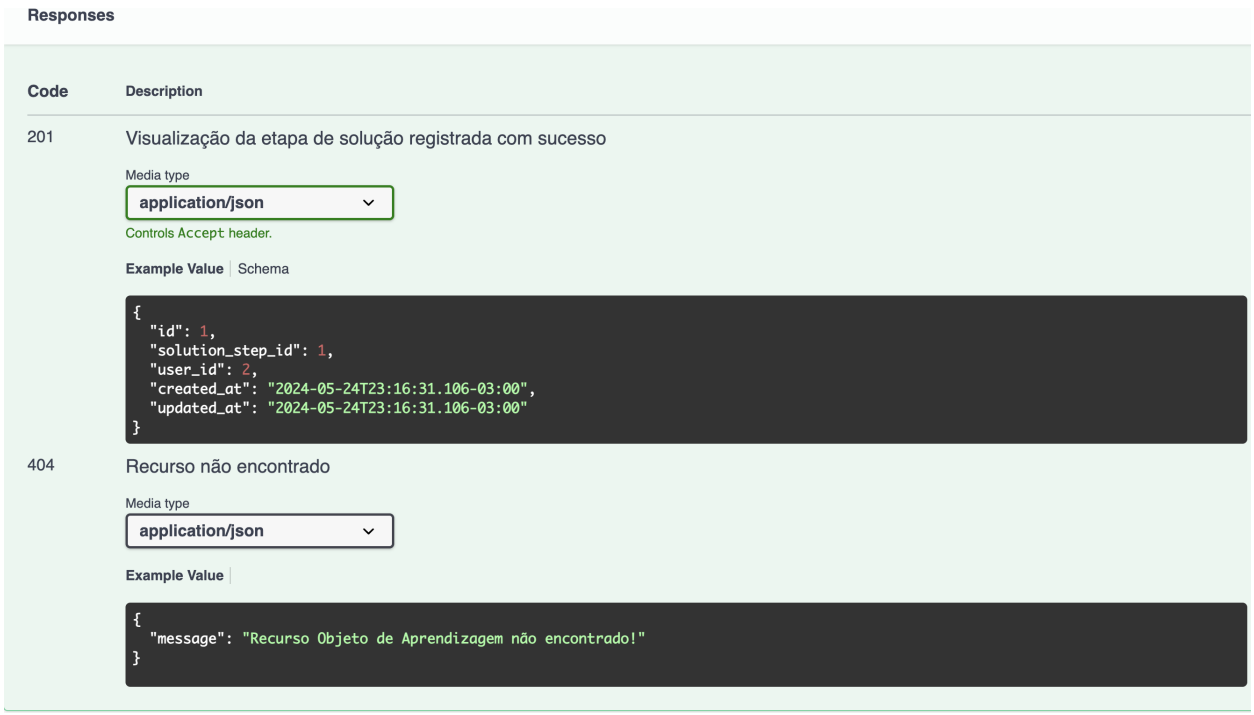


Figura 45 – Documentação da rota (resultado) - PR 9
Fonte: Autoria própria.

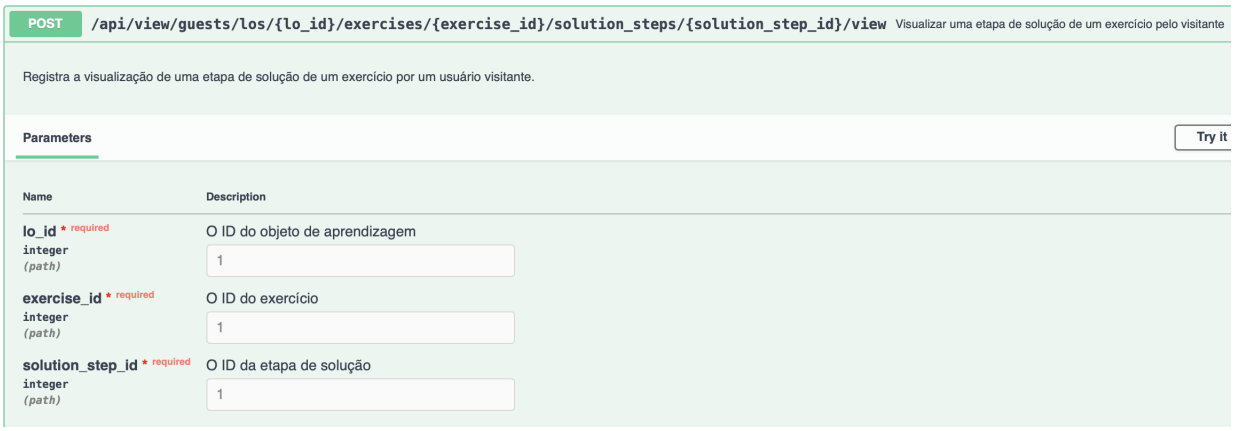


Figura 46 – Documentação da rota - PR 10
Fonte: Autoria própria.

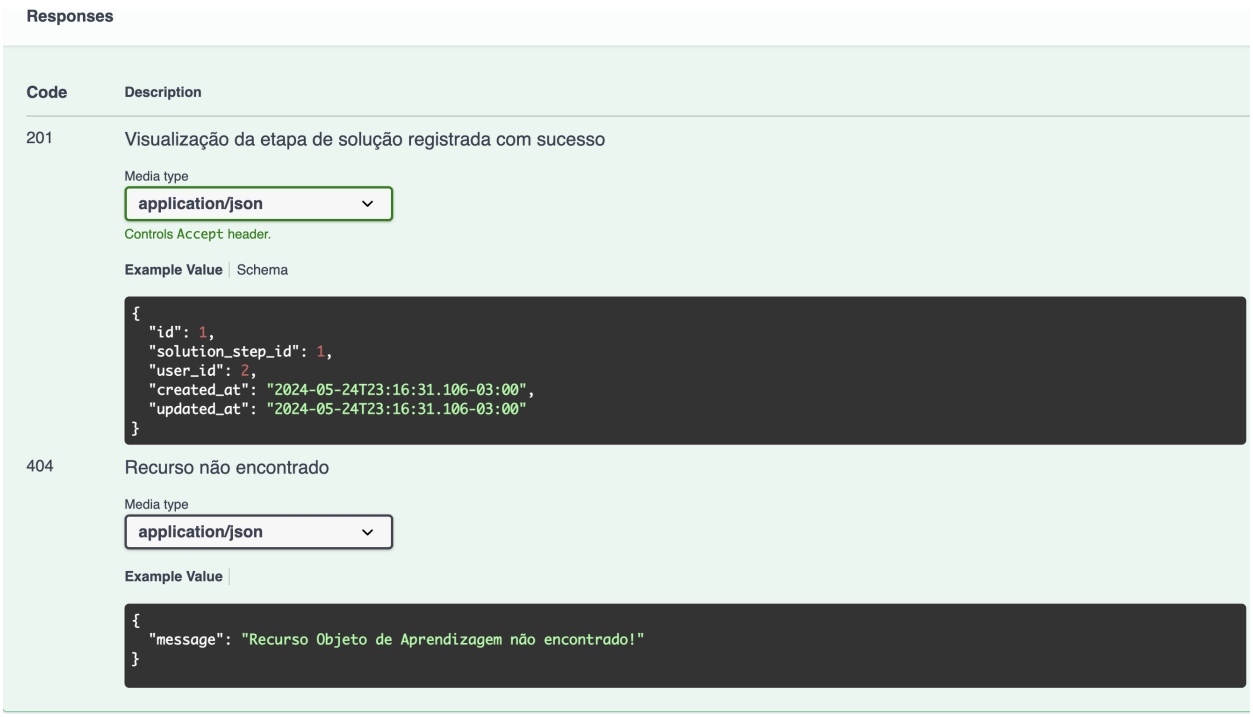


Figura 47 – Documentação da rota (resultado) - PR 10
Fonte: Autoria própria.

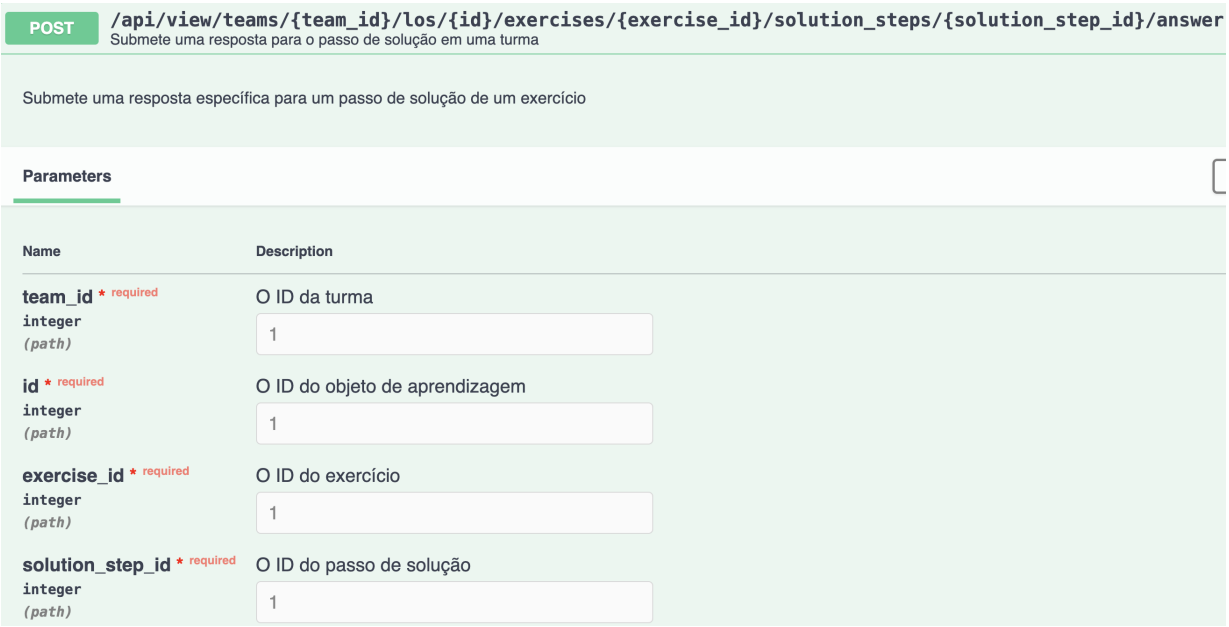


Figura 48 – Documentação da rota - PR 11
Fonte: Autoria própria.

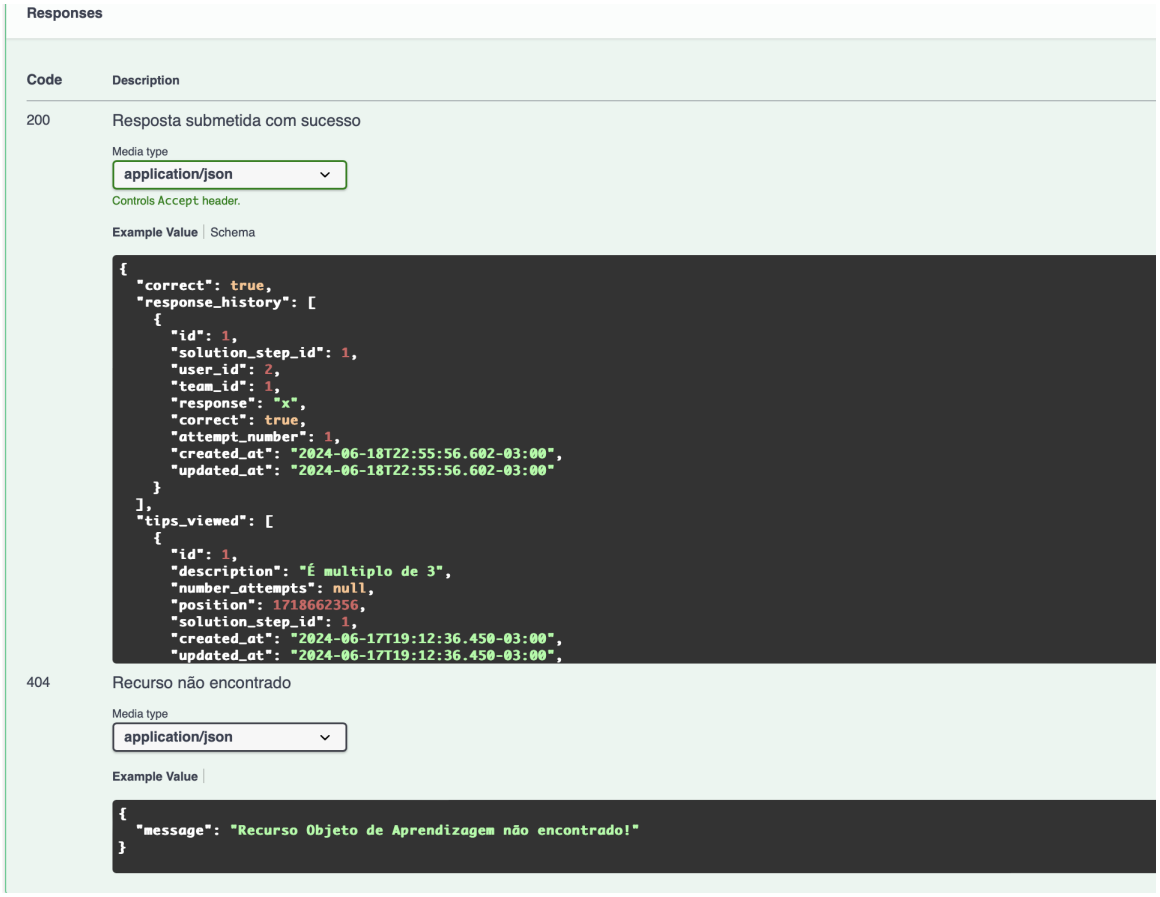


Figura 49 – Documentação da rota (resultado) - PR 11
Fonte: Autoria própria.

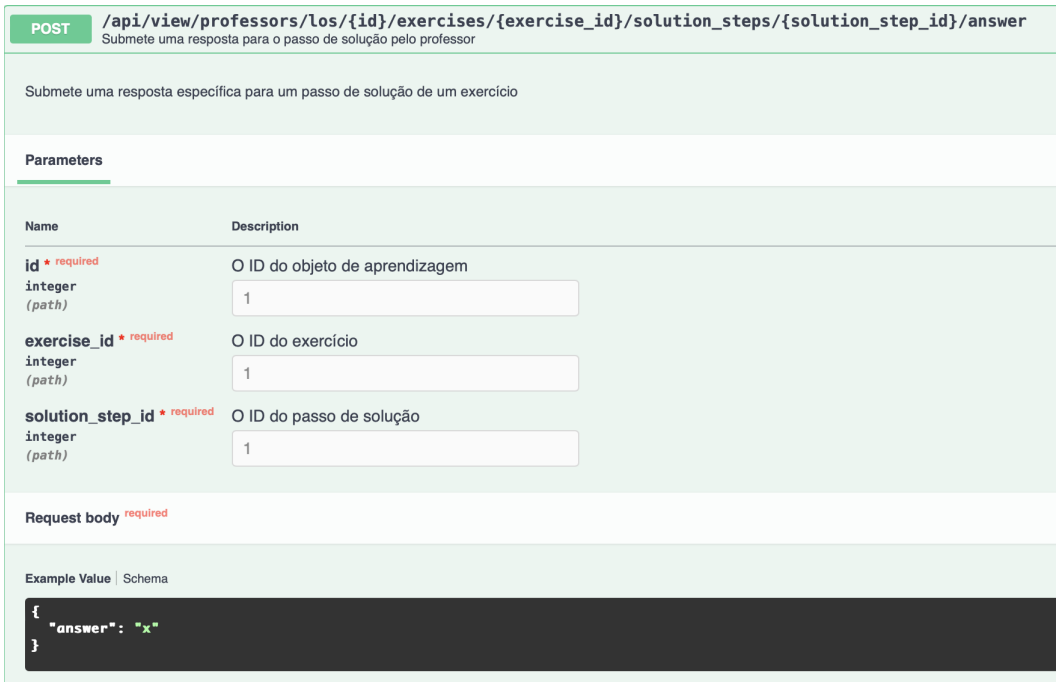


Figura 50 – Documentação da rota - PR 12

Fonte: Autoria própria.

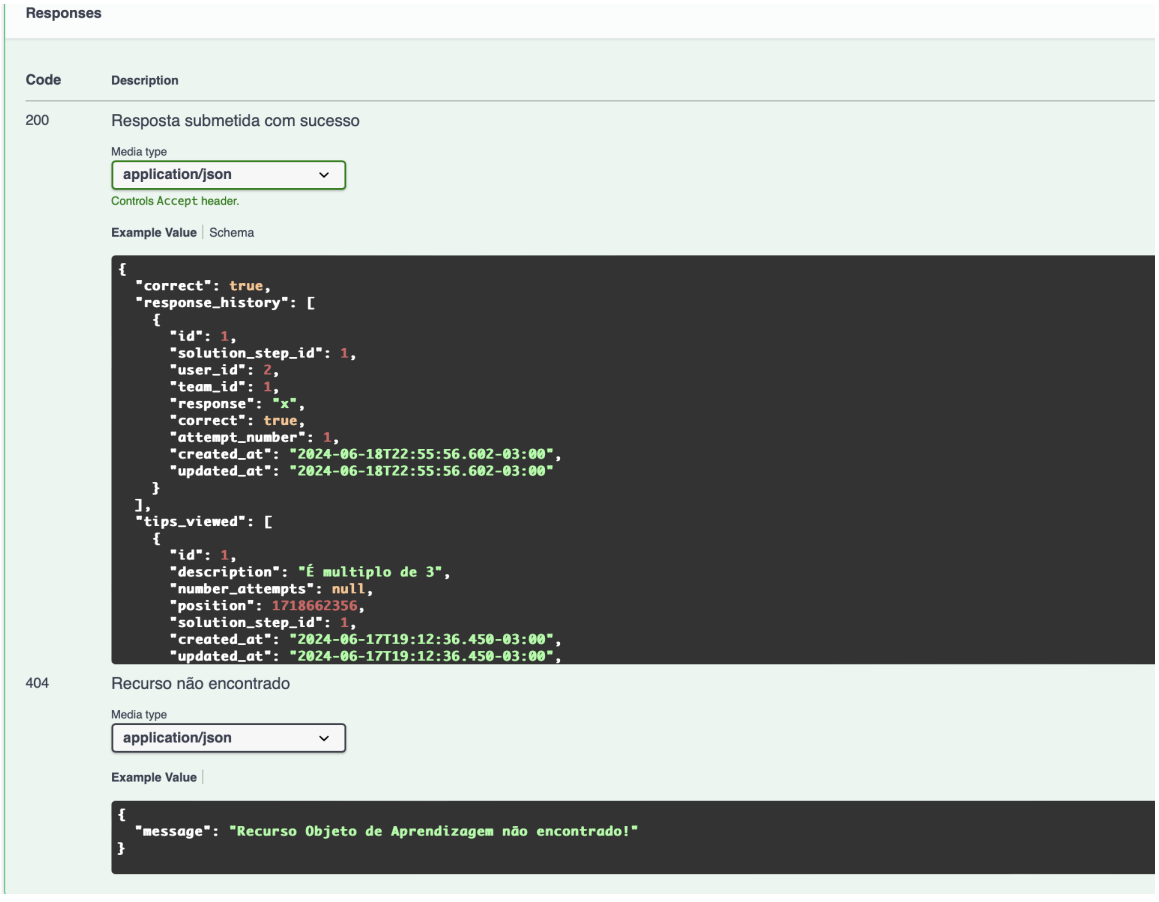


Figura 51 – Documentação da rota (resultado) - PR 12

Fonte: Autoria própria.



Figura 52 – Documentação da rota - PR 13
Fonte: Autoria própria.

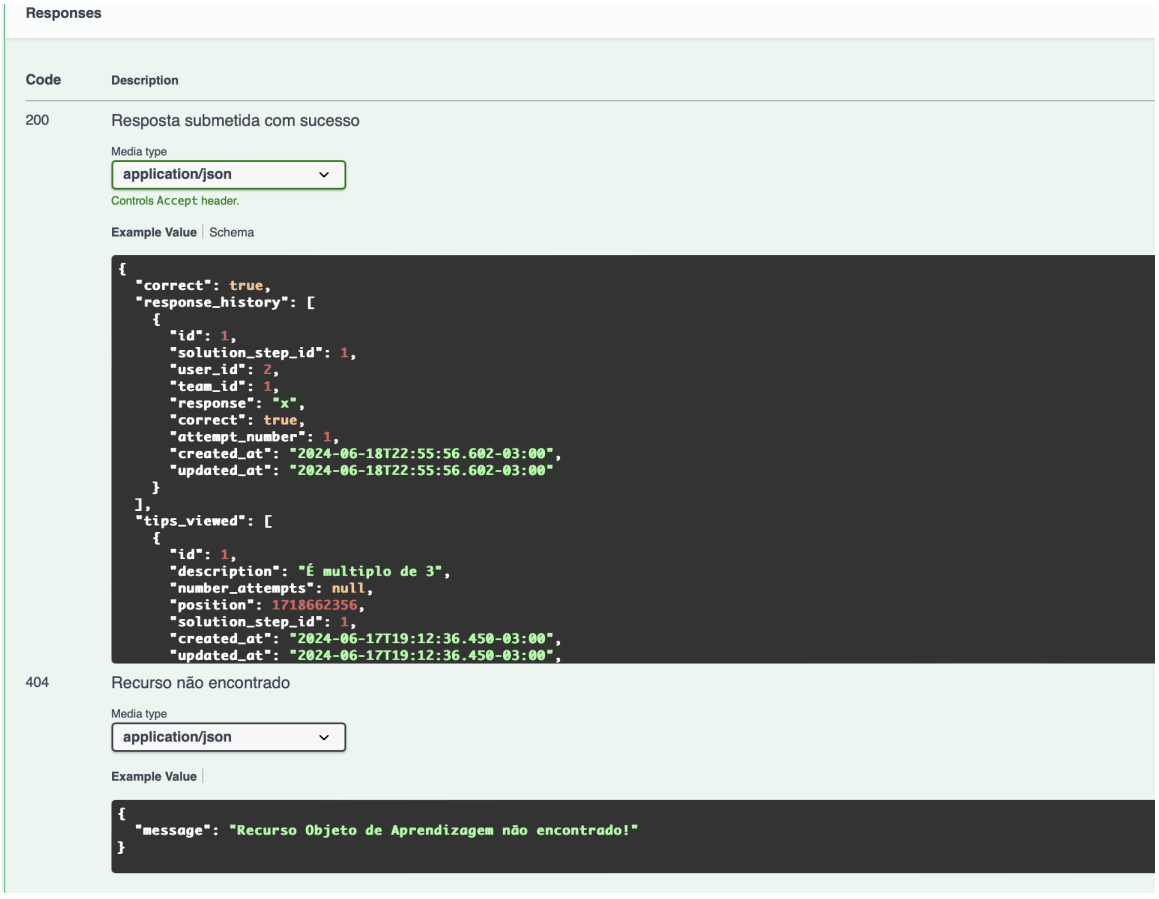


Figura 53 – Documentação da rota (resultado) - PR 13
Fonte: Autoria própria.

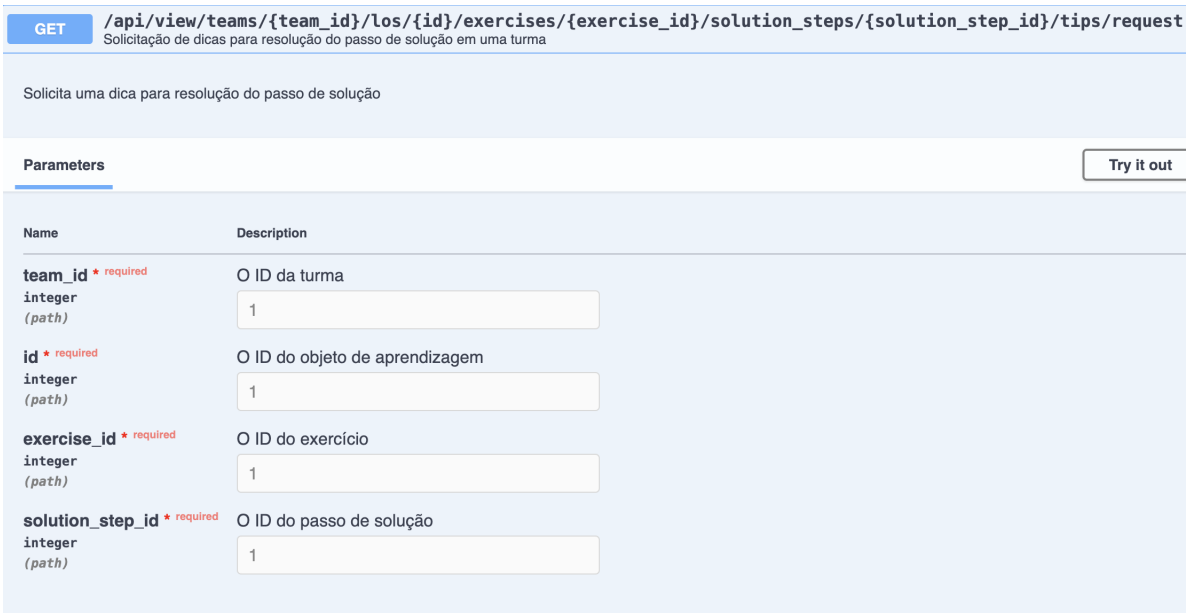


Figura 54 – Documentação da rota - PR 14
Fonte: Autoria própria.

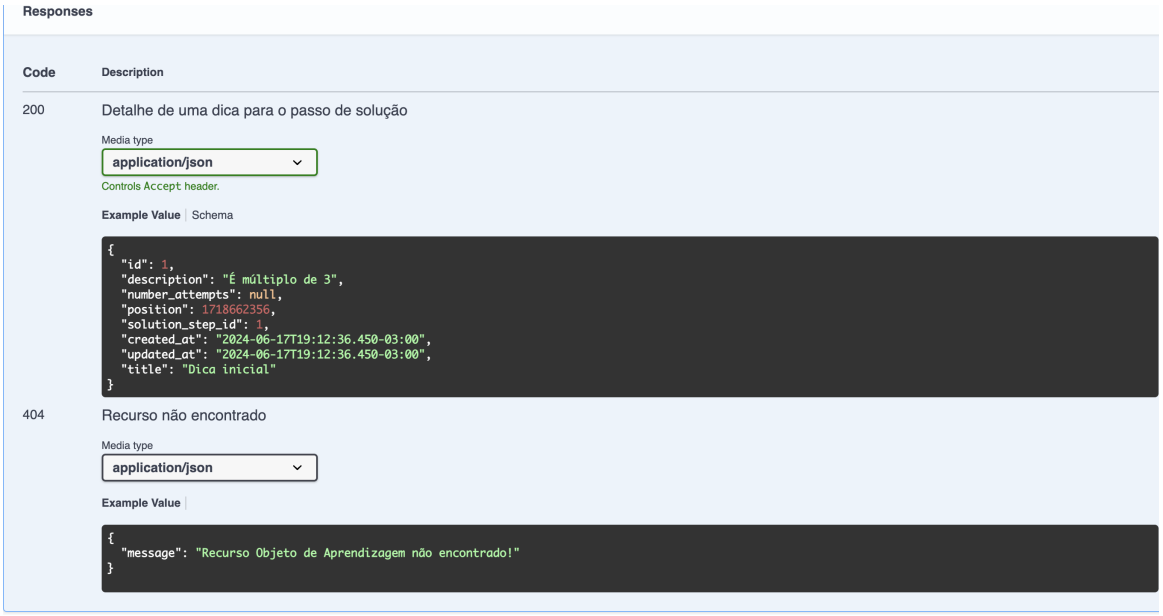


Figura 55 – Documentação da rota (resultado) - PR 14
Fonte: Autoria própria.



Figura 56 – Documentação da rota - PR 15
Fonte: Autoria própria.



Figura 57 – Documentação da rota (resultado) - PR 14
Fonte: Autoria própria.



Figura 58 – Documentação da rota - PR 16
Fonte: Autoria própria.



Figura 59 – Documentação da rota - PR 16
Fonte: Autoria própria.

6 CONSIDERAÇÕES FINAIS

Conforme mencionado na Seção 3.1, a FARMA iniciou uma nova versão com o objetivo de incorporar as contribuições apresentadas nos trabalhos de Correia (2017) e Santos (2018), resultando em uma versão estável Vitek (2023). Assim, este trabalho disponibiliza a implementação da estrutura de modelos para turmas, um aspecto que não estava inicialmente mapeado para este projeto, mas que se mostrou necessário. Essa implementação não será utilizada apenas para a visualização e interação com os objetos de aprendizagem, mas também para futuras contribuições, como a configuração de turmas pelo professor.

Considerando os objetivos gerais, este trabalho disponibiliza os recursos necessários para visualização e interação com objetos de aprendizagem na FARMA. Embora não tenham sido completamente atingidos os objetivos específicos referentes ao contexto de erro e acerto e aos detalhes dessa interação, as rotas implementadas e especificadas atendem à visualização e interação com os objetos de aprendizagem, permitindo a implementação do *front-end* em uma próxima contribuição.

O desenvolvimento deste trabalho seguiu boas práticas, utilizando conceitos como herança e responsabilidade única. Também foram seguidos os princípios do REST, onde as rotas foram definidas a partir dos recursos que a aplicação gerencia. Este trabalho contribui significativamente para a nova versão da FARMA, permitindo novas contribuições e proporcionando uma estrutura bem definida e fácil de expandir com novas funcionalidades.

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.