***Campus* Vilhena**
Coordenação do Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas

PABLO GABRIEL SMOLAK APOLINARIO

Academia FSLab - Desenvolvimento de uma Plataforma MOOC para o FSLab

VILHENA - RO

2025

PABLO GABRIEL SMOLAK APOLINARIO

Academia FSLab - Desenvolvimento de uma Plataforma MOOC para o FSLab

Monografia entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), *Campus* Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor Marco Antônio Augusto de Andrade.

VILHENA - RO

2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Apolinario, Pablo Gabriel Smolak.
Academia FSLab - desenvolvimento de uma plataforma MOOC para o FSLab / Pablo Gabriel Smolak Apolinario. - Vilhena, 2025.
67 f.

Orientador(a): Prof. Me. Marco Antonio Augusto de Andrade.

Trabalho de Conclusão de Curso (Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Vilhena, 2025.

1. Plataforma de cursos. 2. JavaScript. 3. Node.js. 4. Next.js. 5. Docker . I. Andrade, Marco Antonio Augusto de (orient.). II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Rosilene Maria do Couto Marques, CRB-11/321

PABLO GABRIEL SMOLAK APOLINARIO

Academia FSLab - Desenvolvimento de uma Plataforma MOOC para o FSLab

Monografia entregue como Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), *Campus* Vilhena, como requisito parcial para obtenção do grau de Tecnólogo, junto ao Curso Superior em Tecnologia em Análise e Desenvolvimento de Sistemas, sob a orientação do professor Marco Antônio Augusto de Andrade.

Aprovado em: 11/11/2025 pela banca examinadora.

Documento assinado digitalmente
gov.br
ERICK LEONARDO WEIL
Data: 19/11/2025 16:17:40-0300
Verifique em https://validar.iti.gov.br

Erick Leonardo Weil

Examinador Interno

Documento assinado digitalmente
gov.br
WESLEY JHONNES RAMOS ROLIM
Data: 19/11/2025 15:28:05-0300
Verifique em https://validar.iti.gov.br

Wesley Jhonnes Ramos Rolim

Examinador Interno

Documento assinado digitalmente
gov.br
MARCO ANTONIO AUGUSTO DE ANDRADE
Data: 19/11/2025 11:09:52-0300
Verifique em https://validar.iti.gov.br

Marco Antônio Augusto de Andrade

Orientador

Este trabalho é dedicado a todos que fizeram minha vida um pouco mais "Le-gen. . . espera um pouquinho. . . dá-ria!"

Agradecimentos

Em primeiro lugar, agradeço aos meus pais, que foram a base de toda a minha trajetória. À minha mãe, Laudicéia, por todo o amor, incentivo e dedicação, sempre acreditando em mim e me ajudando a chegar até esta etapa da minha vida. Ao meu pai, Joarez, que junto com minha mãe, fez de tudo para que eu tivesse as melhores oportunidades e um futuro melhor. E ao meu pai, Paulo, por sempre estar me apoiando e torcendo pelas minhas conquistas.

Em segundo lugar, agradeço à minha esposa, Lívia, por estar ao meu lado em todos os momentos da minha caminhada acadêmica. Sua paciência, compreensão e apoio constante foram fundamentais para que eu pudesse superar os desafios e concluir mais essa etapa.

Agradeço também ao meu orientador Marco Antônio, pela orientação, paciência e dedicação durante o desenvolvimento deste trabalho, contribuindo de forma essencial para o meu crescimento acadêmico e profissional.

Por fim, agradeço aos meus colegas de classe, que tornaram essa jornada mais leve e enriquecedora. A colaboração, as trocas de experiências e a amizade de todos foram parte importante para que este curso e este Trabalho de Conclusão se tornassem possíveis.

Resumo

Este trabalho apresenta o desenvolvimento da Academia FSLab, uma plataforma de cursos online criada para ajudar no nivelamento técnico entre alunos ingressantes no Laboratório de Fábricas de Software (FSLab) do Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), Campus Vilhena. A dificuldade na integração de novos membros gera uma sobrecarga para a equipe e atrasa o progresso dos projetos. O objetivo principal foi desenvolver um sistema web para facilitar o acesso dos alunos aos cursos necessários para atuarem nas atividades do laboratório. A plataforma foi desenvolvida utilizando JavaScript, com Node.js, Express.js e Prisma ORM no *back-end*, e Next.js no *front-end*. A metodologia de desenvolvimento adotada foi o modelo incremental, com a aplicação conteinerizada em Docker e implantada em um pipeline de CI/CD no GitLab. A plataforma resultante automatiza o acesso a conteúdos, a gestão do progresso e a emissão de certificados, oferecendo um ambiente de aprendizado controlado e eficiente para os novos membros do FSLab.

Palavras-chave: Plataforma de cursos; JavaScript; Node.js; Next.js; Docker; Prisma.

Abstract

This work presents the development of Academia FSLab, an online course platform created to help with the technical leveling of new students joining the Software Factory Laboratory (FSLab) at the Federal Institute of Education, Science and Technology of Rondônia (IFRO), Campus Vilhena. The difficulty in integrating new members causes an overload for the team and delays project progress. The main objective was to develop a web system to facilitate students' access to the necessary courses for participating in the laboratory's activities. The platform was developed using JavaScript, with Node.js, Express.js, and Prisma ORM on the back-end, and Next.js on the front-end. The development methodology adopted was the incremental model, with the application containerized in Docker and deployed through a CI/CD pipeline in GitLab. The resulting platform automates content access, progress management, and certificate issuance, offering a controlled and efficient learning environment for new FSLab members.

Keywords: Course Platform; JavaScript; Node.js; Next.js; Docker; Prisma.

Lista de ilustrações

Lista de quadros

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
REST	Transferência de Estado Representacional
CRUD	Acrônimo para Create (criação), Read (consulta), Update (atualização) e Delete (destruição) de dados
IFRO	Instituto Federal de Educação, Ciência e Tecnologia de Rondônia
JS	JavaScript
JSON	JavaScript Object Notation
ORM	Mapeamento Objeto-Relacional
SQL	Linguagem de Consulta Estruturada
UML	Unified Modeling Language

Sumário

1 Introdução

1.1 Contexto e Problema

A formação na área de tecnologia é significativamente enriquecida por experiências práticas que complementam o aprendizado teórico. Nesse sentido, projetos de extensão como as fábricas de software em instituições de ensino são ambientes cruciais, pois permitem que os alunos vivenciem projetos reais de desenvolvimento de software.

O Instituto Federal de Rondônia (IFRO), Campus Vilhena, abriga o Laboratório de Fábricas de Software (FSLab). O FSLab atrai um grande interesse de alunos de todos os períodos, especialmente os ingressantes, que desejam se juntar aos projetos para adquirir experiência prática.

Contudo, um desafio ocorre neste processo de integração: a falta de conhecimento técnico dos novos membros. Muitos estudantes, ao entrarem no laboratório, ainda não possuem os conhecimentos específicos necessários para contribuir efetivamente com os projetos em andamento. Isso resulta em uma curva de aprendizado lenta e em uma sobrecarga para os membros mais experientes e para a coordenação, que precisam dedicar um tempo considerável ao nivelamento e à capacitação individual dos novatos.

Diante disso, este trabalho tenta ajudar a resolver o seguinte problema: Como desenvolver uma plataforma de cursos online que permita aos professores do FSLab criar e ofertar conteúdo educacional personalizado, a fim de capacitar os alunos ingressantes com as competências necessárias para atuar nos projetos do laboratório?

1.2 Objetivos

1.2.1 Objetivo geral

Desenvolver a plataforma web de cursos online chamada Academia FSLab, com o objetivo de facilitar o acesso a conteúdos para a capacitação e o nivelamento técnico dos novos alunos do FSLab do IFRO Campus Vilhena, buscando acelerar sua integração nos projetos de desenvolvimento.

1.2.2 Objetivos específicos

Para alcançar o objetivo geral, este trabalho busca atingir os seguintes objetivos específicos:

- Entender como funciona um MOOC.
- Garantir a autenticação segura por meio de módulo de login e recuperação de senha.
- Disponibilizar recursos para visualização de cursos, incluindo listagem com filtros e página de detalhes.
- Proporcionar acesso ao conteúdo por meio de um player de cursos integrado.
- Oferecer emissão de certificados aos concluintes.
- Permitir a gestão de informações na página de perfil do usuário.

1.3 Justificativa

A criação da plataforma Academia FSLab justifica-se pela necessidade de oferecer ao Laboratório um ambiente totalmente controlado e adaptado às suas demandas internas. Com essa plataforma, o laboratório passa a ter autonomia para gerenciar seus próprios cursos e conteúdos, permitindo que os professores elaborem materiais de acordo com suas metodologias e com a forma como desejam que os alunos desenvolvam os projetos dentro do mesmo.

Essa flexibilidade contribui para um processo de ensino mais dinâmico e alinhado à prática do desenvolvimento de software, possibilitando que os novos integrantes adquiram rapidamente as competências necessárias para atuar nos projetos. Além disso, a iniciativa reduz a sobrecarga dos alunos mais experientes, uma vez que os novatos podem se preparar de forma independente, acelerando sua integração e fortalecendo o aprendizado coletivo dentro do laboratório.

2 Fundamentação Teórica

2.1 Ensino a distância

O Ensino a Distância (EaD) refere-se a uma modalidade de educação que utiliza as Tecnologias da Informação e Comunicação (TICs) para promover a interação entre professores e alunos, mesmo quando estes estão fisicamente distantes. Este modelo de ensino pode ser aplicado tanto em cursos de graduação quanto em programas de extensão (FRANÇA et al., 2022), facilitando o interesse dos alunos pela participação e contribuindo para a democratização do acesso ao conhecimento.

A demanda por soluções de ensino com acesso facilitado aumentou significativamente durante a pandemia, período em que não podíamos sair de casa. Esse cenário impulsionou a popularidade das plataformas de cursos online. Atualmente, tecnologias como internet e celulares estão presentes em mais de 90% dos lares brasileiros (IBGE, 2024), com isso, o acesso a cursos online tornou-se muito mais prático. Além disso, a modalidade EAD consegue levar o conhecimento a locais distantes, atendendo a pessoas que não têm a possibilidade de ir até as instituições de ensino presencial.

Na modalidade de Ensino a Distância (EaD), o aluno possui maior flexibilidade e autonomia para controlar seu acesso aos cursos, o que lhe permite administrar melhor seu tempo e absorver o conteúdo de acordo com seu ritmo de aprendizagem. Mesmo sendo um modelo a distância, o aluno ainda pode tirar dúvidas por meio de fóruns e chats, o que possibilita esclarecer questionamentos e facilita o entendimento do conteúdo.

Os Ambientes Virtuais de Aprendizagem (AVAs) são plataformas desenvolvidas para viabilizar a implementação de cursos a distância. Essas plataformas só se tornaram possíveis graças à expansão da tecnologia e da internet, facilitando, assim, esse modelo de ensino. Um exemplo amplamente utilizado dessas plataformas é o Moodle (Modular Object-Oriented Dynamic Learning Environment) (FRANÇA et al., 2022). As plataformas virtuais oferecem aos alunos recursos como materiais didáticos, slides, videoaulas, entre outros, o que se torna essencial para o aprendizado nas modalidades de ensino a distância, proporcionando o acesso ao conteúdo necessário para a aquisição do conhecimento.

2.2 MOOCs

Os Cursos Online Abertos e Massivos, do inglês *Massive Open Online Courses* (MOOCs), são cursos totalmente online e gratuitos que abrangem os mais diversos temas. O que realmente diferencia essa categoria de outras modalidades de ensino online é o fato

de serem projetados para acomodar um grande número de participantes simultaneamente, diferentemente do ensino tradicional, no qual as matrículas são limitadas, muitas vezes considerando a necessidade de interação direta entre professor e aluno. Como os MOOCs são estruturados para que os alunos avancem de forma autônoma em seu aprendizado, torna-se viável oferecer esses cursos em larga escala (AGONÁCS; MATOS, 2020).

No geral, os MOOCs baseiam-se em modelos pedagógicos e em teorias de aprendizagem já conhecidas, como o comportamentalismo (AGONÁCS; MATOS, 2020). Os cursos, quando adaptados para o formato MOOC, são projetados para que os alunos sejam avaliados ao final de cada módulo, permitindo que realizem uma autoavaliação do aprendizado e reforcem os conteúdos, caso necessário (REDAÇÃO LYCEUM, 2019).

Assim como no EaD, os MOOCs oferecem a vantagem de permitir a autonomia do estudante e a flexibilidade no modelo de ensino (REDAÇÃO LYCEUM, 2019). Os alunos podem escolher os horários de estudo que melhor se adequam à sua rotina, organizando-se de forma independente. Além disso, ao concluírem os cursos, têm a possibilidade de obter certificados que comprovam sua participação.

2.3 Trabalhos Similares

Durante a pesquisa, foram encontrados os seguintes softwares proprietários para gerenciamento de cursos:

- **Scaffold Education**[1]: Plataforma proprietária para empresas que querem democratizar acesso à aprendizagem de seus funcionarios.
- **Adobe Learning Manager**[2]: Sistema de gerenciamento de aprendizagem (LMS) da Adobe permite a criação de experiências de aprendizado personalizadas, voltadas para funcionários, parceiros e clientes.

Além disso, os seguintes softwares de código aberto para gerenciamento de cursos foram identificados:

- **Open edX**[3]: Plataforma de e-learning gratuita e de código aberto que oferece cursos online personalizados para alunos e educadores.
- **Moodle**[4]: Plataforma de aprendizagem virtual de código aberto que permite a criação de salas de aula virtuais e o desenvolvimento de ambientes de aprendizagem personalizados.

[1] Disponível em <https://scaffoldeducation.com.br/>
[2] Disponível em <https://business.adobe.com/br/products/learning-manager/adobe-learning-manager.html>
[3] Disponível em <https://openedx.org/pt/>
[4] Disponível em <https://moodle.com/pt-br/>

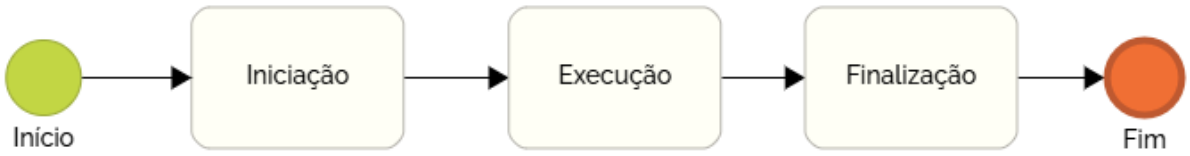
3 Materiais e métodos

Neste capítulo são descritos os materiais e métodos utilizados para o desenvolvimento da solução proposta.

3.1 Ciclo de Vida de Desenvolvimento da Aplicação

O desenvolvimento da plataforma foi realizado em três fases principais: Iniciação, Execução e Finalização, conforme mostrado na Figura 1. O modelo adotado foi o incremental, no qual cada módulo era desenvolvido primeiro na API e, em seguida, no *front-end*. Essa abordagem permitiu entregas parciais e um desenvolvimento mais flexível.

Figura 1 – Ciclo de vida do desenvolvimento



Fonte: elaborado pelo autor (2025).

3.1.1 Fase de Iniciação

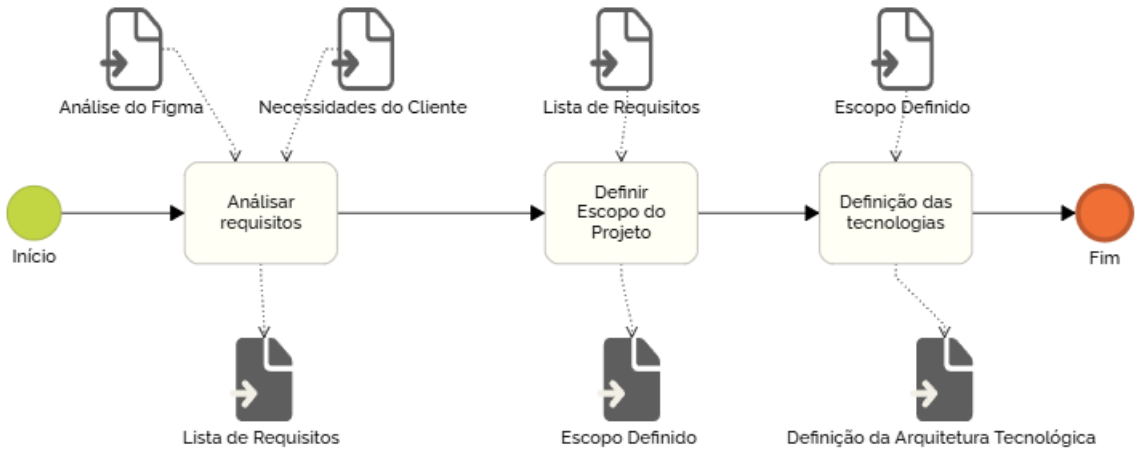
A fase de Iniciação marcou o ponto de partida do projeto, momento em que foram definidos seus objetivos e o escopo funcional, conforme mostrado na Figura 2. Por ser uma aplicação destinada ao FSLab, foi realizada uma análise para compreender as necessidades da equipe e os requisitos mínimos desejáveis. A análise de requisitos, portanto, desempenhou um papel central nesta etapa, sendo fundamental para planejar os passos subsequentes do desenvolvimento.

Adicionalmente, foi nesta fase que se decidiu o ecossistema tecnológico. Para o desenvolvimento da API (*back-end*), a escolha foi por JavaScript[1], Node.js[2], Express.js[3], MinIO[4] para o armazenamento de mídia e o banco de dados MySQL[5]. Essas tecnologias se mostraram adequadas às necessidades do projeto e aos requisitos não funcionais mapeados

[1] Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>
[2] Disponível em: <https://nodejs.org/pt>
[3] Disponível em: <https://expressjs.com/>
[4] Disponível em: <https://min.io/>
[5] Disponível em: <https://www.mysql.com/>

junto ao FSLab. Para o *front-end*, optou-se pelo *framework* Next.js[6], que otimiza o processo de desenvolvimento e entrega uma experiência de usuário mais rica e performática.

Figura 2 – Fase de Iniciação



Fonte: elaborado pelo autor (2025).

3.1.2 Fase de Execução

A fase de Execução foi a etapa em que se colocou em prática o que foi definido na fase anterior. Nela, o projeto foi materializado com a meta de entregar todas as funcionalidades planejadas, garantindo a eficiência do processo e a entrega de um sistema funcional.

A Execução do projeto foi pautada em um modelo de desenvolvimento incremental e iterativo, conforme ilustrado nas Figuras 3 e 4. O processo foi estruturado em ciclos de desenvolvimento, nos quais cada ciclo era responsável pela entrega de um módulo funcional completo do sistema.

Conforme o ciclo de vida incremental (Figura 3), o escopo total do projeto foi coberto por meio do planejamento e da execução sequencial de cada módulo.

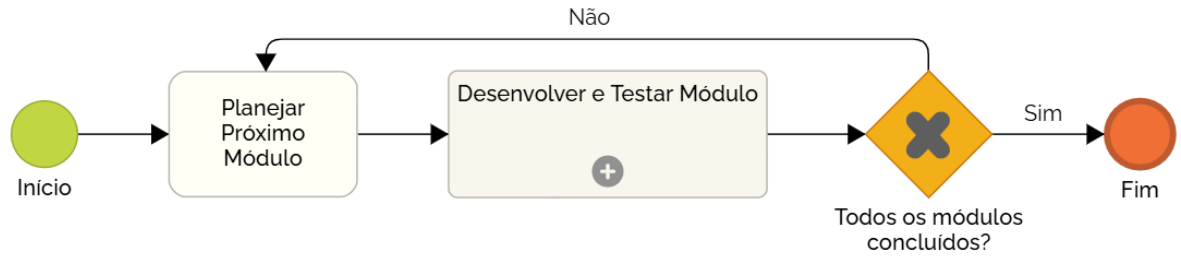
O fluxo de desenvolvimento para cada módulo individual, detalhado na Figura 4, seguia um processo rigoroso de qualidade, iniciando pela implementação da API, que era submetida a testes manuais e de integração, somente após a validação de seu funcionamento, o desenvolvimento do *front-end* era iniciado.

O desenvolvimento do *front-end*, baseado nos protótipos do Figma, avançava com a integração das rotas da API, sendo seguido por testes manuais e de ponta a ponta (E2E) que validavam a funcionalidade completa. Esse modelo permitiu a validação rápida e a

[6] Disponível em: <https://nextjs.org/>

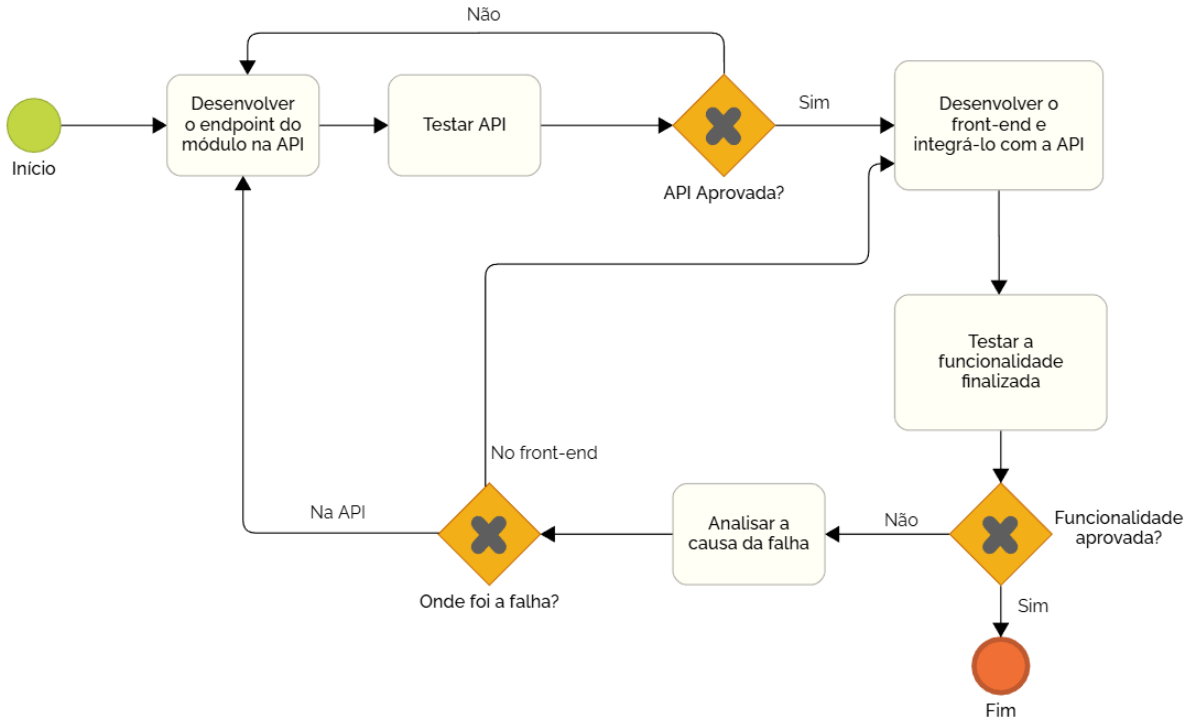
garantia de que a API atendia plenamente às necessidades da interface. Falhas identificadas nos testes eram imediatamente direcionadas à etapa de desenvolvimento correspondente (*back-end* ou *front-end*), assegurando que cada funcionalidade fosse entregue sem *bugs* e não se tornasse um impedimento para a implementação dos módulos subsequentes.

Figura 3 – Ciclo de vida incremental da Fase de Execução



Fonte: elaborado pelo autor (2025).

Figura 4 – Detalhamento do fluxo de desenvolvimento de um módulo



Fonte: elaborado pelo autor (2025).

3.1.3 Fase de Finalização

A Fase de Finalização é a última etapa do projeto, na qual é realizada a documentação, o versionamento e a disponibilização do software, conforme ilustra a Figura 5. Nesta

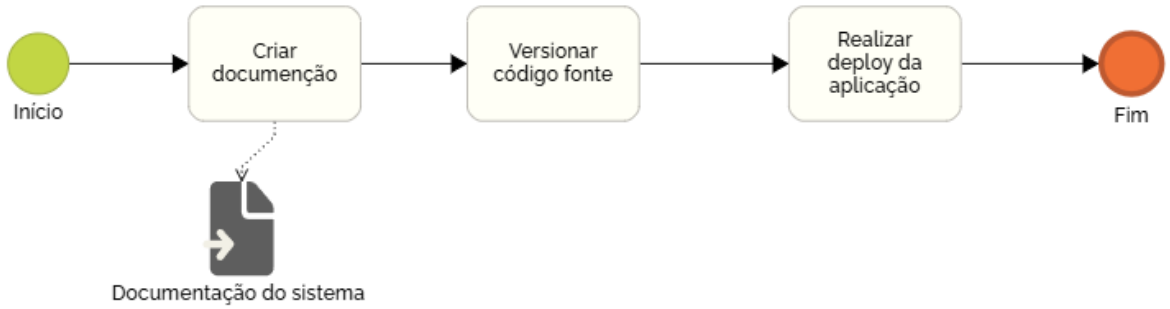
etapa, após a conclusão do código, foi elaborada a documentação da API, com o objetivo de criar um material de fácil acesso para a compreensão do seu funcionamento e simplificar futuras manutenções.

Com todo o processo de desenvolvimento finalizado, foi possível gerar uma nova versão do projeto, devidamente testada e pronta para uso.

A partir da nova versão, realizou-se o *deploy*. O projeto foi empacotado em uma imagem Docker, facilitando o uso do *pipeline* de CI/CD da plataforma GitLab. O ambiente de *deploy* é um servidor próprio que utiliza Kubernetes como orquestrador de contêineres.

A conclusão dessas atividades encerrou esta fase e, consequentemente, o ciclo de vida de desenvolvimento da aplicação.

Figura 5 – Fase de finalização



Fonte: elaborado pelo autor (2025).

3.2 Ferramentas e tecnologias utilizadas

3.2.1 JavaScript

A linguagem JavaScript, é uma das linguagens de programação mais populares do mundo, de acordo com a pesquisa realizada pelo Stack Overflow em 2022 (OVERFLOW, 2022). Isso se deve principalmente ao fato de que o JavaScript é a linguagem padrão que os navegadores interpretam e, mais recentemente, com a criação do Node.js em 2009, começou a ser usada também no *Back-end* das aplicações (ALURA, 2023a).

3.2.2 Node.js

O Node.js[7] é um ambiente de execução de código JavaScript no lado do servidor (ALURA, 2023b). Trata-se de uma plataforma de código aberto e multiplataforma. Atualmente, o Node.js é um dos *web frameworks* mais utilizados, conforme a pesquisa realizada

[7] Disponível em <https://nodejs.org/pt>

pelo Stack Overflow em 2022 (OVERFLOW, 2022). Ele utiliza o mecanismo V8 JavaScript, que é o núcleo do Google Chrome, rodando fora do navegador. Isso permite que o Node.js ofereça alto desempenho e seja amplamente adotado para o desenvolvimento de aplicações escaláveis(NODE.JS, 2024).

3.2.3 API RESTful

Uma Interface de Programação de Aplicações (API) é um conjunto de regras e protocolos que permite a comunicação entre diferentes aplicativos de software, facilitando a troca de dados, recursos e funcionalidades (IBM, 2024).

A arquitetura de uma API é comumente descrita em termos de cliente e servidor: o cliente é a aplicação que faz a solicitação (requisição), enquanto o servidor é a aplicação que processa a solicitação e envia a resposta (SERVICES, 2023).

A Transferência de Estado Representacional (REST) é um estilo de arquitetura de software que define um conjunto de restrições para essa comunicação cliente-servidor. As APIs que seguem o padrão REST, conhecidas como APIs RESTful, utilizam solicitações HTTP para executar funções. Cada solicitação utiliza um verbo HTTP mapeado para uma operação específica em um recurso, como GET (obter), POST (criar), PUT/PATCH (atualizar) e DELETE (remover) (IBM, 2025).

3.2.4 MinIO

O MinIO[8] é um servidor de *Object Storage* distribuído e de alto desempenho, projetado para infraestrutura de nuvem privada em larga escala. O MinIO utiliza volumes persistentes (PVs) para fornecer um *Object Storage* escalável, compatível com as APIs REST do Amazon S3. Ele é especialmente adequado para o armazenamento de dados não estruturados, como fotos, vídeos e imagens de contêiner (IBM, 2021b).

3.3 Requisitos

Os requisitos representam elementos essenciais para o desenvolvimento do sistema, definindo suas funcionalidades e atributos de qualidade. Eles são classificados em dois tipos:

- **Requisitos Funcionais:** descrevem o que o sistema deve realizar, ou seja, suas funcionalidades e comportamentos esperados.
- **Requisitos Não Funcionais:** referem-se às restrições e atributos de qualidade do sistema, como desempenho, segurança e usabilidade.

[8] Disponível em <https://min.io/>

A seguir, são apresentados os requisitos funcionais e não funcionais, tanto do *back-end* quanto do *front-end* do sistema.

Tabela 1 – Tabela de Requisitos funcionais do Sistema

Código	**História de Usuário**	**Descrição**
RF01	Como usuário, quero conseguir me cadastrar na plataforma.	Permitir que o usuário se cadastre na plataforma para conseguir acesso aos cursos.
RF02	Como usuário, quero alterar minhas informações pessoais para mantê-las atualizadas.	Permitir que o usuário gerencie todas as suas informações pessoais na plataforma, incluindo a possibilidade de deletar o perfil caso prefira.
RF03	Como usuário, quero recuperar minha senha caso a esqueça.	Permitir que o usuário recupere o acesso à sua conta por meio de redefinição de senha utilizando e-mail.
RF04	Como cursante, quero filtrar os cursos de meu interesse por ministrante, categoria ou nome.	Permitir que o cursante pesquise e filtre cursos com base em ministrante, categoria ou nome para facilitar a localização dos cursos desejados.
RF05	Como cursante, quero acessar as informações de um curso específico.	Permitir que o cursante visualize a descrição, conteúdo, carga horária, ministrantes e demais informações detalhadas de um curso.
RF06	Como cursante, quero me inscrever nos cursos de interesse.	Permitir que o cursante realize a inscrição nos cursos disponíveis para ter acesso ao conteúdo.
RF07	Como cursante, quero acessar os conteúdos dos cursos e concluí-los.	Permitir que o cursante acesse aulas, materiais e atividades de cada curso, registrando seu progresso até a conclusão.
RF08	Como cursante, quero acessar meus certificados após concluir os cursos.	Permitir que o cursante visualize e baixe os certificados emitidos para cursos concluídos.
RF09	Como cursante, quero adicionar meus certificados ao LinkedIn.	Permitir que o cursante envie seus certificados diretamente para seu perfil no LinkedIn.
RF10	Como cursante, quero visualizar facilmente todos os meus certificados.	Permitir que o cursante acesse uma área dedicada com todos os certificados já obtidos.

Código	História de Usuário	Descrição
RF11	Como cursante, quero acessar facilmente meus cursos em andamento.	Permitir que o cursante veja rapidamente seus cursos já iniciados e continue de onde parou.
RF12	Como ministrante, quero gerenciar categorias de cursos.	Permitir que o ministrante crie, altere ou remova categorias para organizar os cursos.
RF13	Como ministrante, quero criar cursos.	Permitir que o ministrante registre novos cursos.
RF14	Como ministrante, quero adicionar outros ministrantes como instrutores nos cursos que criei ou nos quais sou instrutor.	Permitir que o ministrante atribua outros instrutores a cursos específicos.
RF15	Como ministrante, quero adicionar ou remover categorias dos cursos que criei ou nos quais sou instrutor.	Permitir que o ministrante vincule ou desvincule categorias aos cursos.
RF16	Como ministrante, quero criar, alterar, visualizar ou remover tópicos dos cursos que criei ou nos quais sou instrutor.	Permitir que o ministrante gerencie os tópicos de cada curso.
RF17	Como ministrante, quero criar, alterar, visualizar ou remover conteúdos dos tópicos dos cursos que criei ou nos quais sou instrutor.	Permitir que o ministrante gerencie os vídeos associados a cada tópico.
RF18	Como ministrante, quero visualizar todos os certificados emitidos dos cursos que criei ou nos quais sou instrutor.	Permitir que o ministrante acesse a lista de certificados emitidos.
RF19	Como administrador, quero ter todas as permissões de ministrante e cursante.	Garantir que o administrador possa executar todas as ações disponíveis para ministrantes e cursantes sem restrições.
RF20	Como administrador, quero visualizar, atualizar ou deletar todas as informações do sistema.	Permitir que o administrador gerencie globalmente cursos, usuários, categorias, conteúdos e configurações do sistema.

Tabela 2 – Tabela de Requisitos não funcionais do back-end do Sistema

Código	Requisito	Descrição
RNF01	O sistema deve ser desenvolvido em JavaScript	O back-end deve ser implementado utilizando Node.js com Express para criação de rotas e serviços.
RNF02	O banco de dados deve ser relacional	O sistema deve utilizar MySQL como banco de dados principal.
RNF03	O sistema deve permitir múltiplas requisições simultâneas	A API deve suportar concorrência e escalabilidade, garantindo resposta a vários usuários ao mesmo tempo.
RNF04	O sistema deve estar conteinerizado	Toda a aplicação back-end deve ser executada dentro de containers Docker, facilitando a portabilidade e o deploy.
RNF05	O sistema deve possuir documentação	A API deve ser documentada utilizando Swagger (OpenAPI), permitindo fácil consulta e integração.
RNF06	O sistema deve possuir tratamento de erros	O back-end deve implementar tratamento de exceções e respostas padronizadas para erros de execução e requisições inválidas.
RNF07	O sistema deve possuir integração e entrega contínua	O projeto deve estar configurado com CI/CD no GitLab, garantindo automação nos processos de build e deploy.

Tabela 3 – Tabela de Requisitos não funcionais do front-end do Sistema

Código	Requisito	Descrição
RNF01	O front-end deve ser desenvolvido em JavaScript	O sistema deve utilizar Next.js para renderização SSR e SPA, garantindo performance.
RNF02	O front-end deve ser responsivo	A interface deve se adaptar a diferentes dispositivos (desktop, tablet e mobile).
RNF03	O sistema deve permitir múltiplas requisições simultâneas	O front-end deve suportar chamadas simultâneas à API, sem travamentos ou perda de dados.

Código	Requisito	Descrição
RNF04	O front-end deve estar conteinerizado	A aplicação deve ser executada dentro de Docker, garantindo padronização de ambiente.
RNF05	O sistema deve possuir tratamento de erros	O front-end deve exibir mensagens amigáveis para erros de rede ou de validação, mantendo a experiência do usuário.
RNF06	O sistema deve possuir integração e entrega contínua	O front-end deve estar configurado com CI/CD no GitLab, automatizando build e deploy.
RNF07	O sistema deve ter desempenho otimizado	As páginas devem carregar rapidamente, utilizando técnicas de lazy loading, code splitting e cache.

3.4 Arquitetura do software

Quando se trata de desenvolver aplicações, a escolha de uma boa arquitetura é essencial para garantir o bom funcionamento do software. Considerando que tanto o *back-end* quanto o *front-end* foram desenvolvidos, torna-se necessário definir uma arquitetura adequada para cada parte. Para o *back-end*, utilizou-se uma adaptação do padrão Model-View-Controller (MVC), enquanto o *front-end*, desenvolvido em Next.js, segue uma arquitetura baseada em componentes, aproveitando recursos como Server-Side Rendering (SSR) e Static Site Generation (SSG) para otimizar desempenho e organização do código.

3.4.1 Arquitetura do Back-end: Model-View-Controller

A arquitetura do *back-end* foi desenvolvida seguindo os princípios do padrão *Model-View-Controller* (MVC). A escolha por este padrão se deu por sua capacidade de promover uma clara separação de responsabilidades, dividindo a aplicação em três camadas interconectadas, o que facilita a manutenção e a organização.

Neste projeto, as camadas do MVC foram interpretadas da seguinte forma:

- **Model (Modelo):** Representa a camada de dados e a lógica de negócio da aplicação. É a única camada que interage diretamente com o banco de dados (SANTOS et al., 2010). Neste projeto, o *Model* foi implementado com o auxílio do Prisma ORM, que abstrai as consultas ao banco de dados MySQL e define os modelos de dados.
- **View (Visão):** Em uma API, a *View* não é uma interface visual (HTML), mas sim a representação dos dados que são enviados como resposta ao cliente. Para este

projeto, a *View* consiste nos dados formatados em JSON, que são retornados pelos *endpoints* da API.

- **Controller (Controlador):** Atua como o intermediário entre as requisições do cliente e a camada de *Model* (SANTOS et al., 2010). O *Controller* é responsável por receber as requisições HTTP das rotas, validar os dados de entrada, acionar os métodos apropriados no *Model* para manipular os dados e, por fim, selecionar e retornar a *View* (a resposta JSON) adequada.

3.4.2 Arquitetura do *Front-end*: *Component-Based* com Next.js

O *front-end* da aplicação foi construído com o *framework* Next.js, que adota a arquitetura baseada em componentes. A ideia principal é dividir a tela em blocos menores e reutilizáveis, que chamamos de componentes. Cada um desses blocos funciona de forma independente e agrupa seu próprio código de lógica, aparência e estrutura. Fazer isso torna a criação e a manutenção de telas complexas muito mais simples.

A interface da plataforma foi construída como uma árvore de componentes. Componentes menores e genéricos são combinados para formar componentes maiores e mais específicos, que por sua vez compõem as páginas completas da aplicação.

Além da componentização, o Next.js enriquece a arquitetura com as estratégias de renderização que você mencionou:

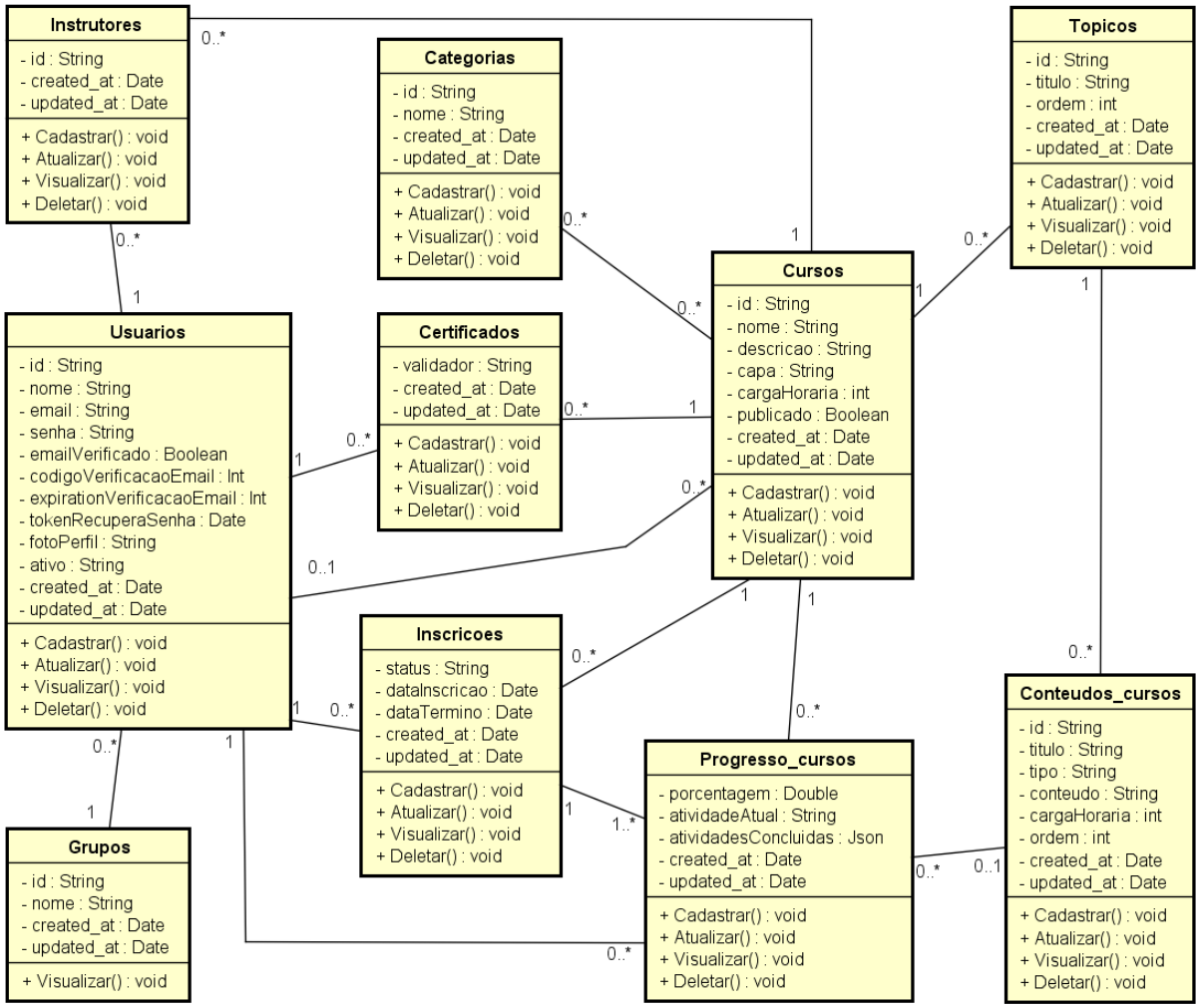
- **Server-Side Rendering (SSR):** Páginas que exigem dados atualizados a cada acesso, como a tela de perfil do usuário, são renderizadas no servidor a cada requisição. Isso garante que o usuário sempre veja as informações mais recentes (VERCEL, 2025b).
- **Static Site Generation (SSG):** Páginas cujo conteúdo muda com menos frequência, como a página de detalhes de um curso já publicado, podem ser pré-renderizadas no momento do *build*. Isso resulta em um carregamento quase instantâneo, melhorando a performance e a experiência do usuário (VERCEL, 2025a).

3.5 Modelagem

Na fase de iniciação do projeto, foi criado o Diagrama de Classes da UML (Unified Modeling Language). O diagrama de classes é importante para o processo de modelagem do sistema, pois representa a estrutura estática, definindo as principais entidades (classes), seus atributos, métodos e os relacionamentos entre elas (FOWLER, 2014). Dependendo da complexidade do sistema, um único diagrama de classe é capaz de modelar um sistema inteiro (IBM, 2021a). Ele facilitou muito o desenvolvimento, pois estabeleceu uma base sólida para o início da codificação.

A estrutura do sistema foi organizada em torno de três eixos principais: gestão de usuários ("Usuarios", "Grupos"), estrutura de cursos ("Cursos", "Categorias", "Topicos", "Conteudos_cursos") e a interação entre eles ("Instrutores","Inscricoes", "Progresso_cursos", "Certificados"). Cada classe no diagrama define um molde para os objetos que o sistema irá manipular, servindo como o principal guia para a arquitetura da API e para a posterior criação da estrutura de persistência de dados.

Figura 6 – Diagrama de classes



Fonte: elaborado pelo autor (2025).

3.6 Persistência de Dados

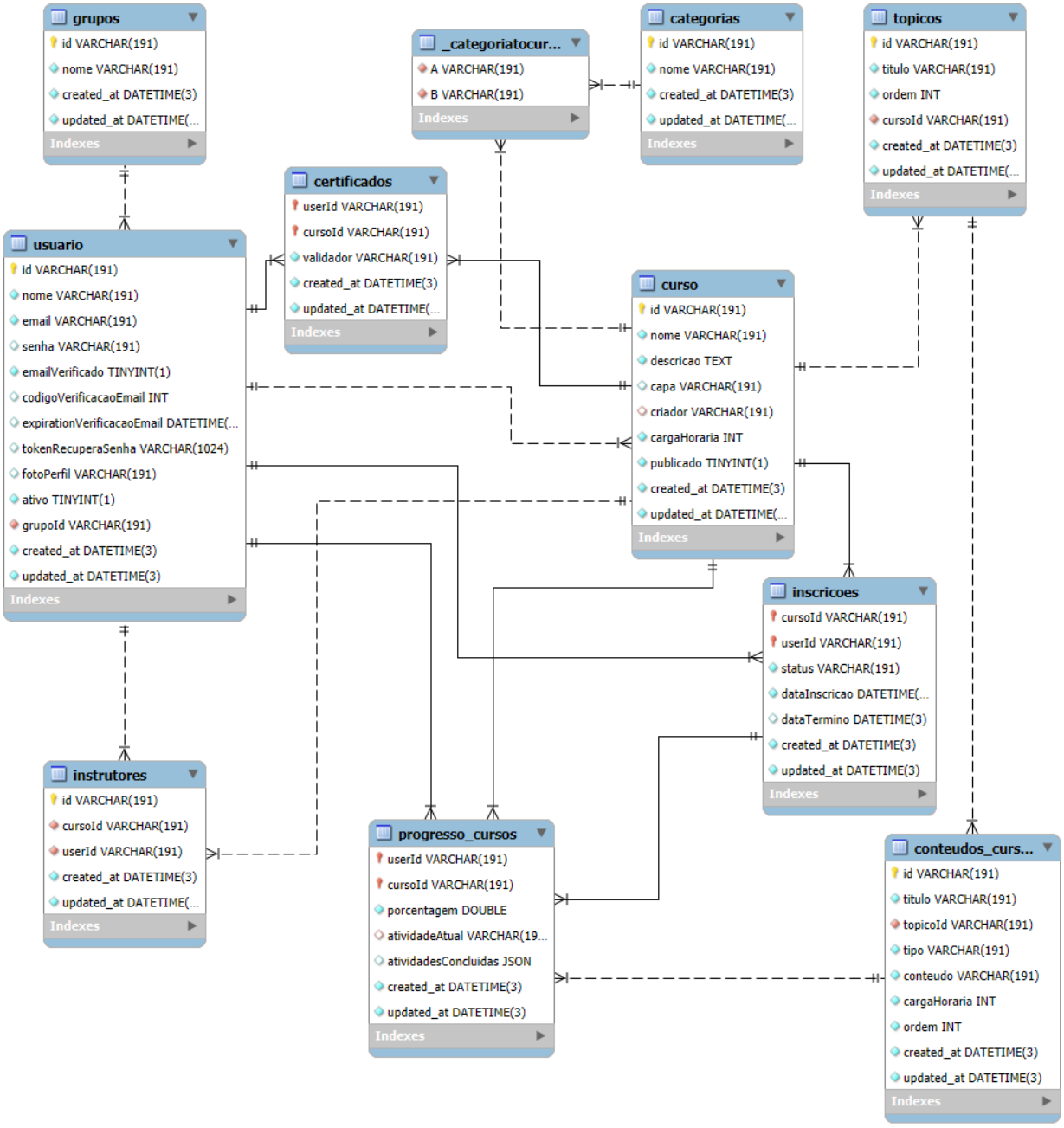
A persistência de dados descreve como as informações, modeladas conceitualmente na seção anterior, são efetivamente armazenadas e gerenciadas. Para este projeto, optou-se por um banco de dados relacional, o MySQL. O modelo de classes serviu como base

para a criação do modelo físico, detalhado no Diagrama de Entidade-Relacionamento (DER) da Figura 7.

A tradução do modelo lógico para o físico seguiu um conjunto de regras e convenções:

- **Classes para Tabelas:** Cada classe do diagrama UML foi mapeada para uma tabela no banco de dados (ex: a classe ‘Usuarios‘ tornou-se a tabela ‘usuario‘).
- **Atributos para Colunas:** Os atributos das classes foram convertidos em colunas, com seus tipos de dados genéricos (como ‘String‘ e ‘Date‘) sendo especificados para os tipos do MySQL (‘VARCHAR(191)‘, ‘DATETIME(3)‘).
- **Relacionamentos 1-para-N:** As associações de um-para-muitos, como entre ‘Usuarios‘ e ‘Inscricoes‘, foram implementadas através de chaves estrangeiras (Foreign Keys). A tabela ‘inscricoes‘, por exemplo, contém as colunas ‘userId‘ e ‘cursoId‘ que referenciam as chaves primárias das respectivas tabelas.
- **Relacionamento N-para-N:** A associação de muitos-para-muitos entre ‘Cursos‘ e ‘Categorias‘ foi materializada por meio de uma tabela de junção, denominada ‘categoriatocurso‘. Esta tabela associativa contém as chaves estrangeiras de ambas as tabelas, permitindo que um curso pertença a várias categorias e uma categoria contenha vários cursos.

Figura 7 – Diagrama de Entidade Relacionamento



Fonte: elaborado pelo autor (2025).

Para realizar a comunicação entre a API e o banco de dados MySQL, foi adotada a ferramenta Prisma ORM (Mapeamento Objeto-Relacional). O Prisma atua como uma camada de abstração que simplifica e torna mais segura a interação com o banco de dados.

O funcionamento do Prisma no projeto se baseou em seu arquivo de *schema* denominado schema.prisma, que define os modelos de dados da aplicação de forma declarativa, espelhando a estrutura das tabelas apresentada no DER (figura 7). A partir deste *schema*, o Prisma gera um cliente de banco de dados totalmente tipado, que foi utilizado na API para executar todas as operações de CRUD.

A escolha pelo Prisma foi estratégica e trouxe os seguintes benefícios ao projeto:

- **Segurança de Tipos:** Garante que as operações com o banco de dados sejam consistentes com os tipos de dados definidos, prevenindo uma classe inteira de erros em tempo de execução.
- **Produtividade:** O auto-complete e as *queries* intuitivas do Prisma Client aceleraram significativamente o desenvolvimento da camada de acesso a dados.
- **Abstração e Segurança:** Ao invés de escrever SQL manualmente, as operações foram feitas através de métodos do Prisma, o que abstrai a complexidade do SQL e ajuda a prevenir vulnerabilidades comuns, como SQL Injection.

Dessa forma, o Prisma atuou como a camada de acesso a dados do sistema, garantindo uma comunicação performática, segura e de fácil manutenção entre a aplicação e o banco de dados.

Quadro 1 – Parte da codificação prisma

```prisma
model Inscricao {
  cursoId       String
  userId        String
  status        String    @default("Em Andamento")
  dataInscricao DateTime  @default(now())
  dataTermino   DateTime?
  created_at    DateTime  @default(now())
  updated_at    DateTime  @updatedAt

  curso     Curso          @relation(fields: [cursoId], references: [id])
  usuario   Usuario        @relation(fields: [userId], references: [id])
  progresso ProgressoCurso?

  @@id([userId, cursoId])
  @@map("inscricoes")
}


// modelo de Progresso do Curso
model ProgressoCurso {
  userId               String
  cursoId              String
  porcentagem          Float
  atividadeAtual       String?
  atividadesConcluidas Json?
  created_at           DateTime @default(now())
  updated_at           DateTime @updatedAt

  curso     Curso          @relation(fields: [cursoId], references: [id])
  usuario   Usuario        @relation(fields: [userId], references: [id])
  conteudo  ConteudoCurso? @relation(fields: [atividadeAtual], references: [id])
  inscricao Inscricao      @relation(fields: [userId, cursoId], references: [userId, cursoId])

  @@id([userId, cursoId])
  @@map("progresso_cursos")
}
```

Fonte: elaborado pelo autor (2025).

O trecho de código apresentado no Quadro 1 traz como exemplo o uso do prisma na modelagem de dados do sistema: o acompanhamento do progresso do aluno. Esta estrutura foi criada para separar de forma clara a matrícula em um curso da evolução do aluno dentro dele, utilizando para isso os modelos Inscricao e ProgressoCurso.

O modelo Inscricao atua como uma tabela de junção explícita entre Usuario e Curso. Sua chave primária é composta pela combinação de userId e cursoId (@@id([userId, cursoId])), o que garante a regra de negócio de que um usuário só pode se inscrever uma única vez no mesmo curso. Esta entidade armazena dados estáticos sobre a matrícula, como o status e a dataInscricao.

Por sua vez, o modelo ProgressoCurso foi criado para armazenar os dados dinâmicos da jornada do aluno, como a porcentagem de conclusão e as atividadesConcluidas. O que

garante a integridade do progresso do aluno é a forma como os dois modelos se relacionam:

- A linha progresso ProgressoCurso? no modelo Inscricao estabelece que uma inscrição pode ter um registro de progresso associado.
- A linha inscricao Inscricao @relation(...) no modelo ProgressoCurso cria um relacionamento obrigatório de um-para-um com a inscrição correspondente, utilizando a mesma chave primária composta como chave estrangeira.

Essa estrutura garante a integridade dos dados, assegurando que um registro de progresso só possa existir se houver uma inscrição válida correspondente. Adicionalmente, o uso do tipo JSON para o campo atividadesConcluidas confere alta flexibilidade, permitindo armazenar uma lista de IDs de atividades concluídas sem a necessidade de criar tabelas adicionais.

3.7 Licença de Uso

A licença escolhida para este projeto foi a *MIT License*[9]. Essa licença, originalmente desenvolvida pelo *Massachusetts Institute of Technology* (MIT), é amplamente utilizada em projetos de software livre por sua simplicidade e permissividade. Ela permite que qualquer pessoa utilize, copie, modifique, distribua e até comercialize o software, desde que seja mantido o aviso de copyright e a licença original.

[9] Disponível em <https://mit-license.org/>

4 Resultados e discussões

4.1 Gerenciamento de Configuração e Mudanças

O gerenciamento do código-fonte é importante para garantir a integridade e a rastreabilidade do projeto ao longo de seu ciclo de vida (ATLASSIAN, 2025). A adoção de um processo de versionamento bem configurado é crucial mesmo em cenários de desenvolvimento individual, pois previne a perda de código e a introdução de erros decorrentes de um gerenciamento inadequado. Nesse contexto, a utilização de um sistema de controle de versão robusto como o Git[1] foi uma decisão estratégica para a execução bem-sucedida deste trabalho.

4.1.1 Ferramenta Adotada: GitLab

A ferramenta selecionada para apoiar todo o processo de Gerenciamento de Configuração e Mudanças foi a plataforma GitLab[2]. A escolha se deu por sua natureza de solução integrada, que vai além de um simples repositório de código Git, oferecendo um ecossistema completo para o ciclo de vida de desenvolvimento de software. Os principais recursos utilizados foram:

- **Repositórios Git:** Para o controle de versão distribuído do código-fonte da API e do front-end.
- **Issue Tracking:** Para o cadastro, discussão e acompanhamento de tarefas, novas funcionalidades e correções de *bugs*.
- **Merge Requests (MRs):** Para a revisão de código e integração controlada das alterações no ramo principal de desenvolvimento.
- **CI/CD Pipelines:** Para a automação dos processos de *build* da aplicação a cada nova alteração.

4.1.2 Processo de Controle de Versão e Mudanças

O projeto foi estruturado em dois repositórios independentes na plataforma GitLab, sendo um dedicado ao *back-end* e outro ao *front-end*. Os repositórios podem ser acessados através dos seguintes links:

[1] Disponível em: <https://git-scm.com/>
[2] Disponível em: <https://about.gitlab.com/>

- ***Back-end*:** <https://gitlab.fslab.dev/academia-fslab/academia-fslab-back-end>
- ***Front-end*:** <https://gitlab.fslab.dev/academia-fslab/academia-fslab-front-end>

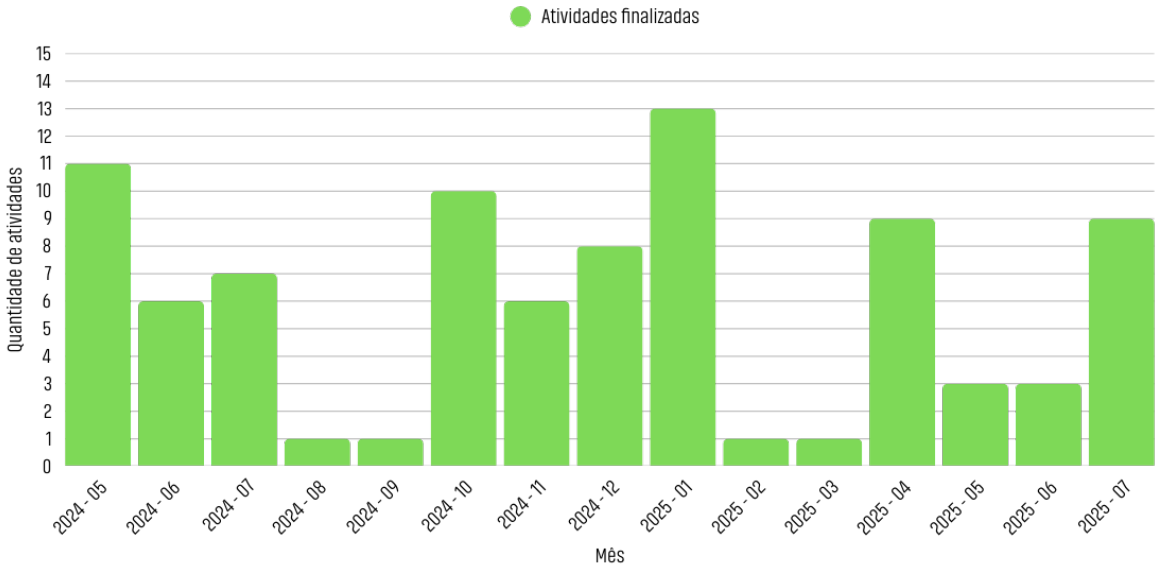
Apesar da separação, a mesma estratégia de ramificação foi adotada, garantindo a consistência do fluxo de trabalho. A estrutura dos ramos (*branches*) foi organizada da seguinte forma:

- ***main*:** Ramo protegido que representa a versão estável e de produção do software. Nenhuma alteração é feita diretamente neste ramo.
- ***development*:** Ramo principal de desenvolvimento. Todas as novas funcionalidades são integradas aqui antes de serem promovidas para o ramo main.
- ***feature*/nome-da-funcionalidade:** Ramos temporários criados a partir do *development*. Todo o desenvolvimento de uma nova funcionalidade ou tarefa ocorre de forma isolada em um ramo de *feature* próprio.

4.2 Processo de desenvolvimento

Para um melhor entendimento do desenvolvimento do trabalho, foram extraídos alguns dados importantes em forma de gráficos. Os gráficos analisados são: *Lead Time* (Tempo Total de Entrega), *Cycle Time* (Tempo de Ciclo de Desenvolvimento) e *Throughput* (Taxa de Entrega). Tais gráficos permitem visualizar o tempo total para a entrega de uma tarefa, o tempo total de desenvolvimento de uma atividade e a quantidade de entregas realizadas em um mês. A seguir, será apresentada a análise dessas métricas geradas.

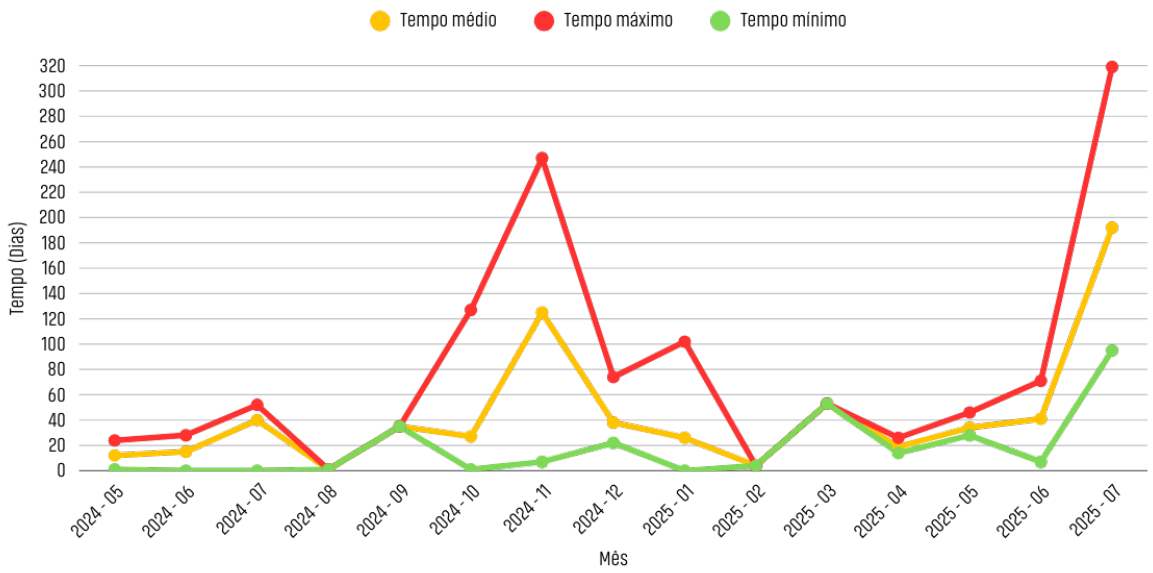
Figura 8 – Throughput



Fonte: elaborado pelo autor (2025).

O gráfico de *Throughput*, apresentado na Figura 8, mostra a quantidade de atividades finalizadas ao longo dos meses, evidenciando picos de entrega e meses com baixa ou nenhuma conclusão de tarefas. Do ponto de vista do processo, essa irregularidade não ocorreu por falta de desenvolvimento, mas sim por uma dificuldade em gerenciar o projeto seguindo a metodologia ágil proposta. Embora houvesse a tentativa de adaptação, em muitos momentos o formalismo da metodologia era deixado de lado para que o desenvolvimento continuasse. Os gráficos a seguir demonstrarão o impacto disso, como em atividades que, apesar de concluídas, não tiveram o devido acompanhamento na ferramenta de controle e, por isso, constam como se tivessem ficado por muito tempo em aberto.

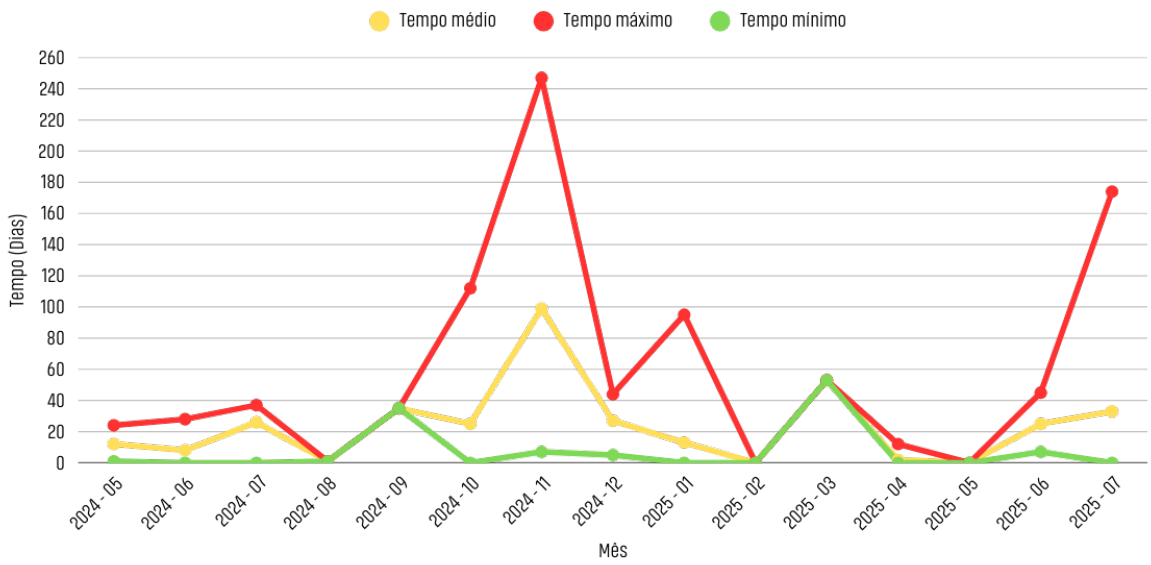
Figura 9 – Lead time



Fonte: elaborado pelo autor (2025).

O gráfico apresentado na Figura 9 é o de *Lead Time*, que demonstra a quantidade de tempo que as atividades finalizadas ficaram em aberto, agrupadas por data de finalização. Nota-se que algumas atividades foram concluídas em tempo hábil, mas observa-se que em duas ocasiões as atividades levaram muito tempo para serem concluídas. Os picos no mês de novembro de 2024 e no mês de julho de 2025 acabaram elevando a média de tempo para conclusão das atividades. No geral, a média ficou em 49 dias para a conclusão de uma tarefa, o que, na prática, não foi bem assim. O que elevou a taxa média foi o uso incorreto da metodologia ágil.

Figura 10 – Cycle Time



Fonte: elaborado pelo autor (2025).

Por fim, o gráfico demonstrado na Figura 10 mostra o tempo real empregado no desenvolvimento, com base em quando a atividade foi iniciada e concluída. Seguindo a tendência do gráfico anterior, os picos de tempo permanecem nos mesmos meses, o que apenas reforça a premissa de que a metodologia ágil não foi usada corretamente. Isso também trouxe um aumento expressivo na média de tempo de desenvolvimento, que foi de 23 dias, embora, na realidade, a maioria das entregas tenha demandado menos tempo para ser concluída.

4.3 Testes e Validação da Aplicação

Para garantir que o código-fonte cumpra os requisitos propostos, a criação de testes é uma etapa fundamental do ciclo de desenvolvimento. Neste projeto, foi adotada uma estratégia de validação abrangente para assegurar a qualidade, a confiabilidade e o correto funcionamento da solução.

A abordagem de testes combinou verificação manual e automatizada para cobrir diferentes aspectos do sistema, desde componentes isolados até a jornada completa do usuário.

4.3.1 Estratégia e Tipos de Testes

Foram aplicados diferentes tipos de testes, cada um com um objetivo específico e utilizando as ferramentas adequadas, conforme detalhado a seguir.

4.3.1.1 Testes de Unidade

Os testes de unidade focaram em validar as menores partes lógicas do sistema de forma isolada, como funções-chave e regras de negócio específicas. O objetivo era verificar se cada componente individualmente se comportava como o esperado.

- **Ferramentas (API):** Framework *Jest*.

4.3.1.2 Testes de Integração

Realizados na API, os testes de integração verificaram se os diferentes módulos do *back-end* (como *Controllers*, *Routers* e *Middlewares*) funcionavam corretamente em conjunto. Eles validaram o fluxo de dados e a comunicação entre as camadas da aplicação.

- **Ferramentas (API):** *Jest* em conjunto com o *Supertest* para simular requisições HTTP.

4.3.1.3 Testes de Ponta a Ponta (E2E)

Os testes de ponta a ponta (End-to-End) foram aplicados no *front-end* para analisar a funcionalidade completa do sistema sob a perspectiva do usuário. Eles simularam a jornada do usuário na interface, garantindo que a integração com o *back-end* estava funcionando de forma adequada.

- **Ferramentas (Front-end):** Framework Cypress.

4.3.1.4 Testes Manuais

Os testes manuais foram aplicados de forma exploratória em ambas as partes da aplicação. O objetivo era navegar livremente pelo sistema em busca de *bugs*, falhas de usabilidade ou comportamentos inesperados que os testes automatizados poderiam não prever.

- **Ferramentas:** Postman para o *back-end* e o navegador Google Chrome para o *front-end*.

4.3.2 Resultados e Conclusões

A execução da estratégia de testes descrita permitiu alcançar resultados importantes para a validação do projeto. Nos testes manuais, foi possível obter *feedback* imediato, confirmar as respostas esperadas nos *endpoints* e validar as regras de negócio. Nos testes automatizados, os testes unitários confirmaram o comportamento isolado das funções, enquanto os testes de integração validaram a comunicação entre os módulos da API.

A realização de um ciclo de testes completo garantiu um alto nível de confiabilidade e qualidade para a aplicação. A validação de todas as funcionalidades de forma sistemática foi crucial para assegurar que a versão do código a ser colocada em produção estivesse estável, livre de *bugs* conhecidos e pronta para proporcionar uma boa experiência ao usuário final.

A seguir, são apresentados os relatórios de execução dos testes unitários, de integração e de ponta a ponta (E2E), oferecendo uma visão detalhada da cobertura de código, do comportamento da API e da validação da jornada do usuário no sistema.

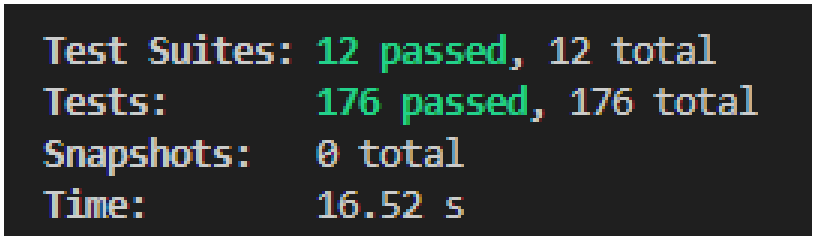
Quadro 2 – Testes de integração de inscrições

```
describe('Testes de inscricões', () => {
  describe('/POST em inscricões', () => {
    it('Deve retornar erro tentar se increver em um curso com id inválido',
      async () => {
        const res = await request(app)
          .post(`/inscricoes`)
          .set('Authorization', `Bearer ${token}`)
          .send({
            cursoId: 'f6efef75-1904-41c2-9ec0-c2cc58b067f3'
          })

        expect(res.statusCode).toEqual(422);
        expect(res.body.errors).toContainEqual(
          "Nenhum curso publicado encontrado com esse ID"
        );
      }
    );

    it('Deve retornar erro tentar se increver em um curso sem tópicos',
      async () => {
        const res = await request(app)
          .post(`/inscricoes`)
          .set('Authorization', `Bearer ${token}`)
          .send({
            cursoId: curso2.id
          })

        expect(res.statusCode).toEqual(422);
        expect(res.body.errors).toContainEqual(
          "Não é possivel se inscrever em um curso sem tópicos"
        );
      }
    );
  });
});
```

Fonte: elaborado pelo autor (2025).

O Quadro 2 mostra dois exemplos de teste de integração na rota de criação de inscrição em um curso. No primeiro caso, testa-se a criação de uma inscrição utilizando um ID ao qual nenhum curso está atribuído, o que causa um erro que é validado pelo teste. No outro caso, o ID do curso é válido, porém ele foi publicado incorretamente, sem nenhum tópico de aulas cadastrado, retornando também um erro. Esse tipo de teste assegura que o usuário que está tentando se inscrever realmente tenha acesso a um curso válido, evitando erros futuros.

Figura 11 – Resultados da Execução dos Testes de Integração



Fonte: elaborado pelo autor (2025).

A Figura 11 apresenta a quantidade de testes de integração executados, totalizando 176 testes divididos em 12 suítes, todos finalizados com sucesso em 16,52s. Isso demonstra que as rotas da API estão funcionando como o esperado e evidencia o nível de confiabilidade da API.

Figura 12 – Relatório de Cobertura dos Testes de Integração

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	87.46	72.87	76	88.27	
src	100	83.33	100	100	
app.js	100	83.33	100	100	21
src/config	68.08	50	88.88	68.88	
initialConfig.js	66.66	50	85.71	65.51	10-11,18-25,64-65
minioConfig.js	100	50	100	100	6
prismaClient.js	60	100	100	60	28-31
zodConfig.js	83.33	50	100	100	7
src/controller	98.18	91.85	89.47	100	
authController.js	94.73	87.5	100	100	14
categoriaController.js	100	95	100	100	73
certificadoController.js	100	85.71	100	100	14-15
conteudoController.js	97.11	92.18	75	100	42,81,190,259,339
cursosController.js	97.28	89.58	88	100	97,112,161,212,246,535-536,599,751,813,867,946-951
gruposController.js	100	100	100	100	
inscricoesController.js	100	100	100	100	
progressoController.js	100	92.85	100	100	13-14
recuperaSenhaController.js	96.66	92.85	100	100	29
topicoController.js	96.47	90.38	70	100	35,142,230-235,285
usuarioController.js	100	93.93	100	100	168-198,376
verificarEmailController.js	100	100	100	100	
src/middleware	81.39	59.25	100	85	
EmailVerificadoMiddleware.js	100	100	100	100	
authMiddleware.js	71.42	50	100	83.33	29,33
logRoutesMiddleware.js	85.71	66.66	100	85.71	12
multerMiddleware.js	75	50	100	75	9-10
permissaoMiddleware.js	90	62.5	100	88.88	21
src/routes	96.66	50	50	96.66	
TopicoRouter.js	100	100	100	100	
authRouter.js	100	100	100	100	
categoriaRouter.js	100	100	100	100	
certificadoRouter.js	100	100	100	100	

Fonte: elaborado pelo autor (2025).

A Figura 12 traz o relatório de cobertura dos testes de integração. Nele, são apresentadas informações importantes, detalhando, no geral e por arquivo, as coberturas de

linhas executáveis de código (%Stmts), condições de decisão (%Branch), funções testadas (%Funcs) e linhas de código cobertas (%Lines). Analisando o relatório, nota-se que os testes de integração já resultam em uma alta cobertura de toda a API, sendo ela de 87,46%.

Quadro 3 – Teste unitário da função de upload para o MinIO

```
describe('Teste de minio Function', () => {
  describe('Testes da função upload', () => {
    beforeEach(() => {
      fPutObject.mockReset();
      unlinkSync.mockReset();
      uuidv4.mockReset();
      extname.mockReset();
    });

    it('Envia o arquivo com nome uuid se nome original não for um uuid',
      async () => {
        const fakeFile = {
          originalname: 'imagem.png',
          path: '/tmp/teste.png'
        };

        uuidv4.mockReturnValue('123e4567-e89b-12d3-a456-426614174000');
        extname.mockReturnValue('.png');

        const bucket = 'meu-bucket';
        const nomeEsperado = '123e4567-e89b-12d3-a456-426614174000.png';

        const resultado = await minioFunctions.upload(
          fakeFile,
          bucket
        );

        expect(fPutObject).toHaveBeenCalledWith(
          bucket,
          nomeEsperado,
          fakeFile.path
        );
        expect(unlinkSync)
          .toHaveBeenCalledWith(fakeFile.path);
        expect(resultado).toBe(nomeEsperado);
      }
    );
  });
});
```

Fonte: elaborado pelo autor (2025).

O Quadro 3 mostra um teste de unidade realizado na função criada para facilitar e padronizar a inserção de arquivos no MinIO. Através do uso de *mocks*, o teste garante que a lógica interna desta função opere conforme o esperado, validando seu comportamento de forma isolada.

Figura 13 – Métricas dos testes da API

```
Test Suites: 19 passed, 19 total
Tests:       306 passed, 306 total
Snapshots:   0 total
Time:        28.761 s
```

Fonte: elaborado pelo autor (2025).

Figura 14 – Relatório final de cobertura dos testes da API

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	96.72	87.12	93	98.08	
src	100	83.33	100	100	
app.js	100	83.33	100	100	21
src/config	68.08	50	88.88	68.88	
initialConfig.js	66.66	50	85.71	65.51	10-11,18-25,64-65
minioConfig.js	100	50	100	100	6
prismaClient.js	60	100	100	60	28-31
zodConfig.js	83.33	50	100	100	7
src/controller	98.18	91.85	89.47	100	
authController.js	94.73	87.5	100	100	14
categoriaController.js	100	95	100	100	73
certificadoController.js	100	85.71	100	100	14-15
conteudoController.js	97.11	92.18	75	100	42,81,190,259,339
cursosController.js	97.28	89.58	88	100	97,112,161,212,246,535-536,599,751,813,867,946-951
gruposController.js	100	100	100	100	
inscricoesController.js	100	100	100	100	
progressoController.js	100	92.85	100	100	13-14
recuperaSenhaController.js	96.66	92.85	100	100	29
topicoController.js	96.47	90.38	70	100	35,142,230-235,285
usuarioController.js	100	93.93	100	100	168-198,376
verificarEmailController.js	100	100	100	100	
src/middleware	81.39	59.25	100	85	
EmailVerificadoMiddleware.js	100	100	100	100	
authMiddleware.js	71.42	50	100	83.33	29,33
logRoutesMiddleware.js	85.71	66.66	100	85.71	12
multerMiddleware.js	75	50	100	75	9-10
permissaoMiddleware.js	90	62.5	100	88.88	21
src/routes	96.66	50	50	96.66	
TopicoRouter.js	100	100	100	100	
authRouter.js	100	100	100	100	
categoriaRouter.js	100	100	100	100	
certificadoRouter.js	100	100	100	100	
conteudoRouter.js	100	100	100	100	
cursoRouter.js	100	100	100	100	
grupoRouter.js	100	100	100	100	

Fonte: elaborado pelo autor (2025).

As Figuras 13 e 14 trazem as métricas completas dos testes da API, totalizando 306 testes divididos em 19 suítes e resultando em 96,72% de cobertura no código da API. Isso demonstra a alta confiabilidade do código e valida que o sistema se comporta conforme o planejado para os cenários testados.

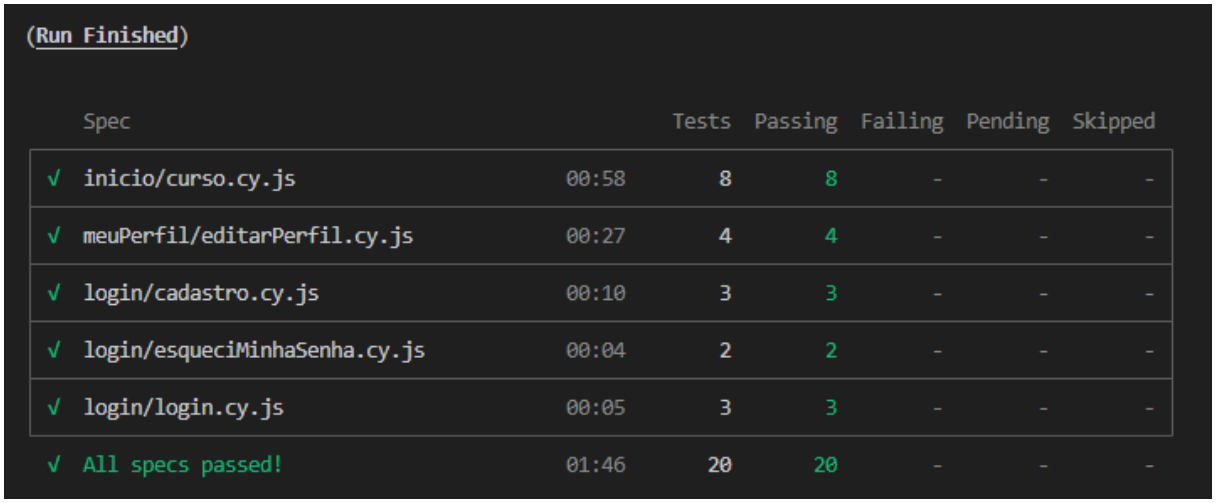
Quadro 4 – Teste E2E de navegação e conclusão de um curso

```
function navegarConteudo() {
    cy.wait(500);

    cy.get('body').then(($body) => {
        const temProximo = $body.find('[data-test="btnProximoConteudo"]').length > 0;
        const temFinalizar = $body.find('[data-test="btnfinalizarConteudo"]').length > 0;

        if (temFinalizar) {
            cy.getByData('btnfinalizarConteudo')
                .should('be.visible')
                .click()
                .then(() => {
                    navegarConteudo();
                });
        } else if (temProximo) {
            cy.getByData('btnProximoConteudo')
                .should('be.visible')
                .click()
                .then(() => {
                    navegarConteudo();
                });
        }
    });
}

it('deve navegar para o player do curso e finalizar o curso', () => {
    cy.logar();

    cy.getByData('button-proxima-pagina').click();
    cy.getByData('linkCurso3').click();

    cy.url().should('match', /\/curso\/[0-9a-fA-F-]{36}$/);

    cy.getByData("btnAcessarCurso").click();

    cy.url().should('match', /\/curso\/[0-9a-fA-F-]{36}\/player$/);

    navegarConteudo();

    cy.getByData('linkSidebarCertificado').should('exist');
})
```

Fonte: elaborado pelo autor (2025).

O teste do Quadro 4 é um teste de ponta a ponta (E2E) que valida o fluxo de navegação entre os conteúdos de um curso. Nele, é simulado o processo de abrir um curso, finalizar cada conteúdo sequencialmente até chegar ao último e, por fim, obter o certificado. Esse tipo de teste assegura que a jornada do usuário no sistema funcione corretamente, da mesma forma que ocorreria em um uso real.

Figura 15 – Relatório de Execução dos Testes de Ponta a Ponta



Fonte: elaborado pelo autor (2025).

A Figura 15 mostra os testes de ponta a ponta (E2E) das principais funcionalidades do sistema, que representam os fluxos de maior utilização pelos usuários. A execução bem-sucedida destes testes garante que os principais casos de uso do sistema funcionem corretamente sob a perspectiva do usuário final.

4.4 Documentação

A documentação do projeto está dividida em duas partes principais. Como o *front-end* e o *back-end* estão em repositórios diferentes, cada um conta com um arquivo README.md próprio. Estes arquivos contêm informações essenciais, como: funcionalidades, tecnologias utilizadas, estrutura de pastas, variáveis de ambiente e os comandos para rodar o projeto localmente, conforme mostrado nas Figuras 17 e 16.

Figura 16 – *Readme back-end*

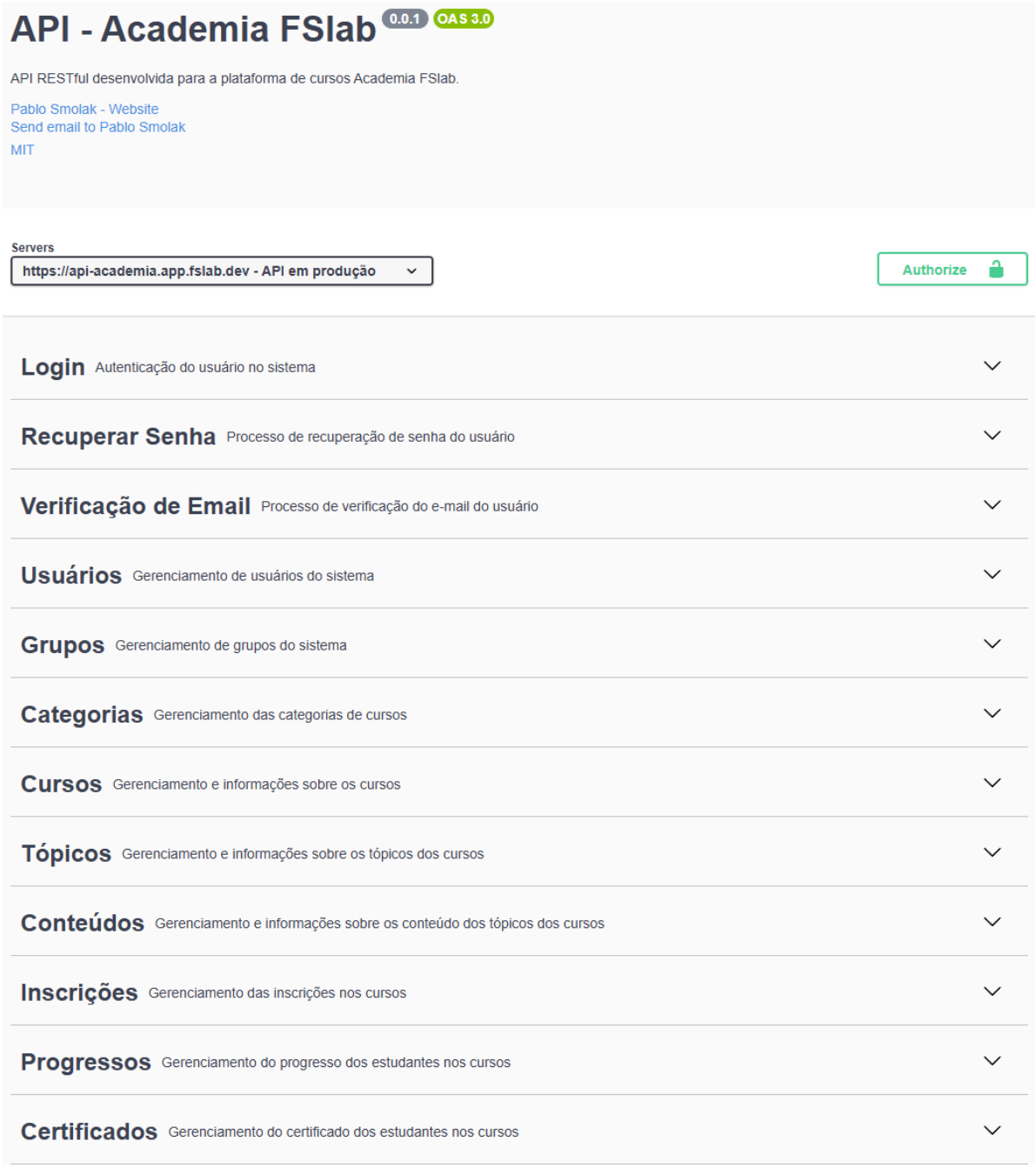
Fonte: elaborado pelo autor (2025).

Figura 17 – *Readme front-end*

Fonte: elaborado pelo autor (2025).

O segundo componente da documentação do projeto é a ferramenta Swagger[3], voltada para a documentação de APIs. Além de explicar as funcionalidades de cada rota, a ferramenta permite que se façam requisições diretamente na API, conforme ilustra a Figura 18, que mostra a organização das rotas do sistema em categorias.

Figura 18 – Documentação Swagger



Fonte: elaborado pelo autor (2025).

Conforme apresentado na Figura 18, as rotas da API foram organizadas por categorias, separando-as por suas funcionalidades específicas. Essa abordagem trouxe uma

[3] Disponível em <https://swagger.io/>

maior organização para a documentação, refletindo a estrutura do código-fonte. As rotas foram divididas nas seguintes categorias:

- **Login:** Autenticação do usuário no sistema.
- **Recuperar Senha:** Rotas para o processo de recuperação de senha
- **Verificação de E-mail:** Rotas para o usuário conseguir verificar o seu e-mail.
- **Usuários:** Gerenciamento de usuários do sistema
- **Grupos:** Gerenciamento de grupos do sistema
- **Categorias:** Gerenciamento das categorias de cursos
- **Cursos:** Gerenciamento e informações sobre os cursos
- **Tópicos:** Gerenciamento e informações sobre os tópicos dos cursos
- **Conteúdos:** Gerenciamento e informações sobre os conteúdo dos tópicos dos cursos
- **Inscrições:** Gerenciamento das inscrições nos cursos
- **Progressos:** Gerenciamento do progresso dos estudantes nos cursos
- **Certificados:** Gerenciamento do certificado dos estudantes nos cursos

Embora a API possua 59 rotas ao todo e todas tenham seu papel essencial, optou-se por mostrar apenas a documentação das rotas referentes à inscrição de usuários nos cursos, que é responsável por gerenciar as matrículas dos alunos. A seguir, são apresentadas capturas de tela da documentação das rotas de inscrição, apresentando as operações de GET, POST e DELETE. As figuras mostram o resumo da funcionalidade da rota, os requisitos obrigatórios para sua utilização, as permissões necessárias e os parâmetros de entrada de cada rota.

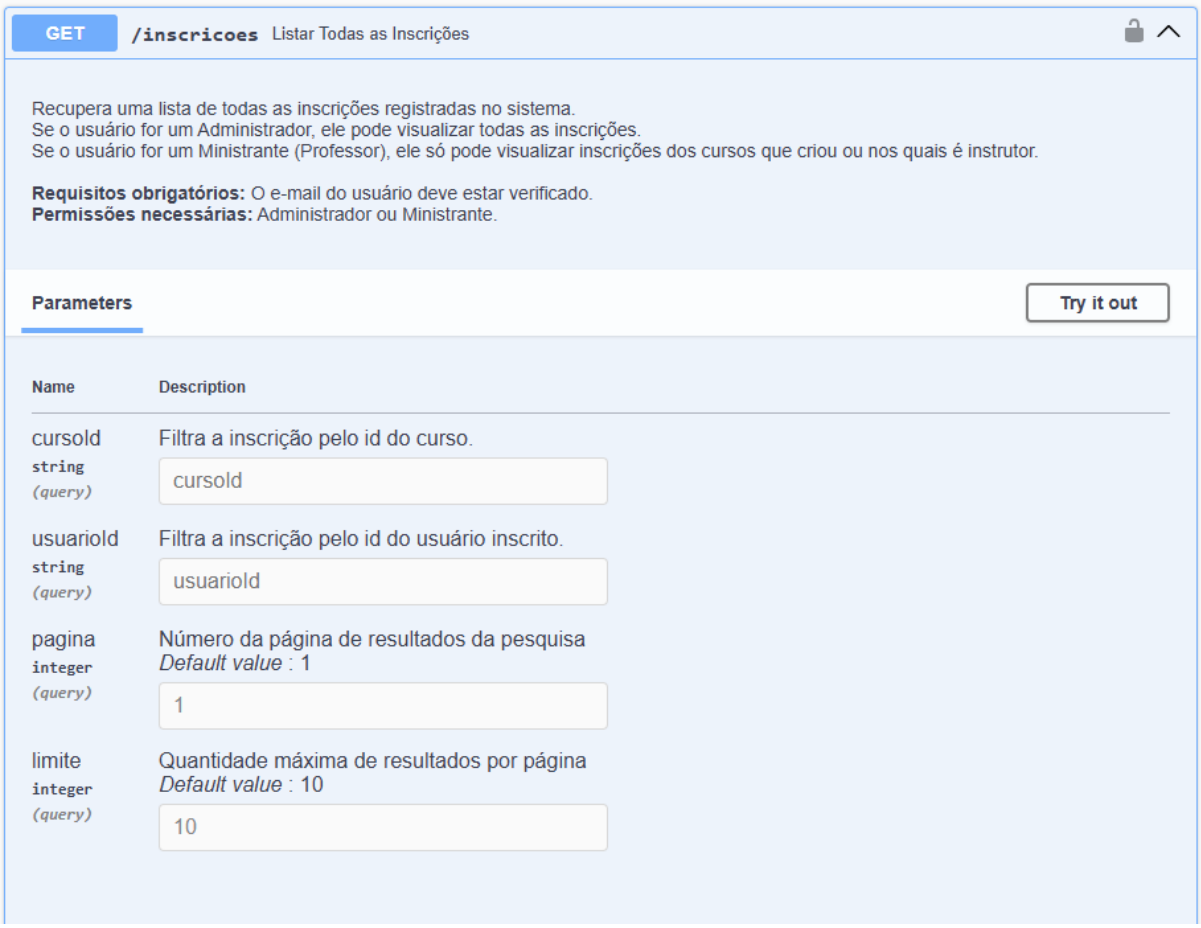
- **POST /inscricoes:** Rota responsável por criar uma nova inscrição.

Figura 19 – Requisição POST para Criação de Inscrição

Fonte: elaborado pelo autor (2025).

- **GET /inscricoes:** Rota responsável por listar todas as inscrições no sistema.

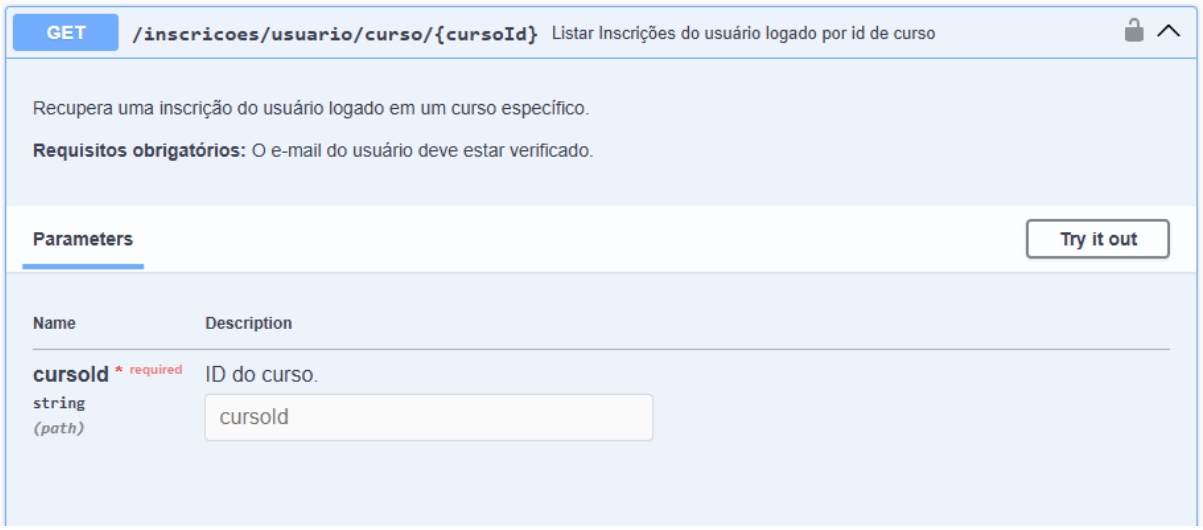
Figura 20 – Requisição GET para Listar Todas as Inscrições



Fonte: elaborado pelo autor (2025).

- **GET /inscricoes/usuario/curso/{cursoId}:** Rota responsável por retornar a inscrição do usuário logado em um curso específico.

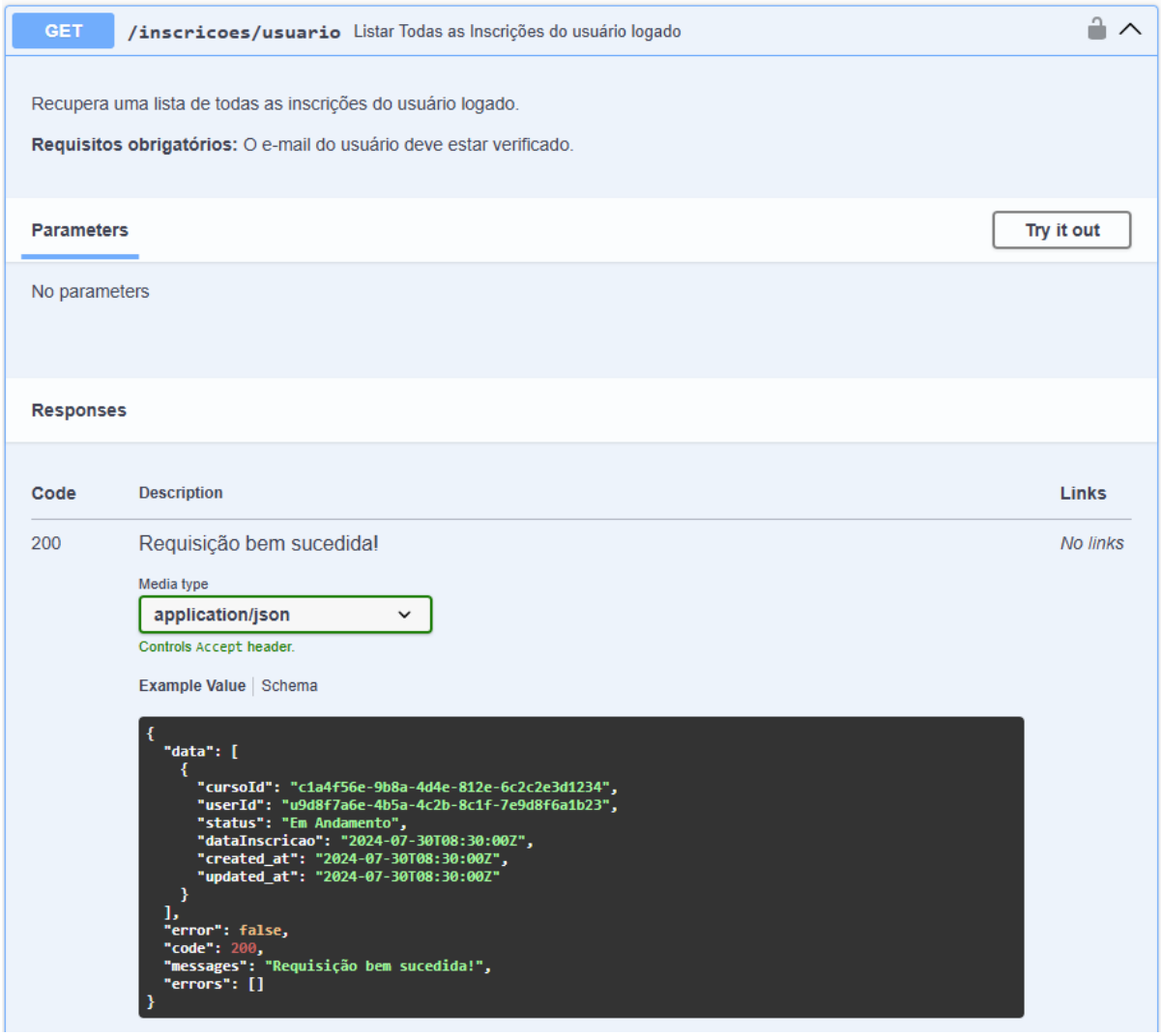
Figura 21 – Requisição GET de Inscrição do Usuário por ID do Curso



Fonte: elaborado pelo autor (2025)

- **GET /inscricoes/usuario:** Rota responsável por retornar todas as inscrições do usuário logado.

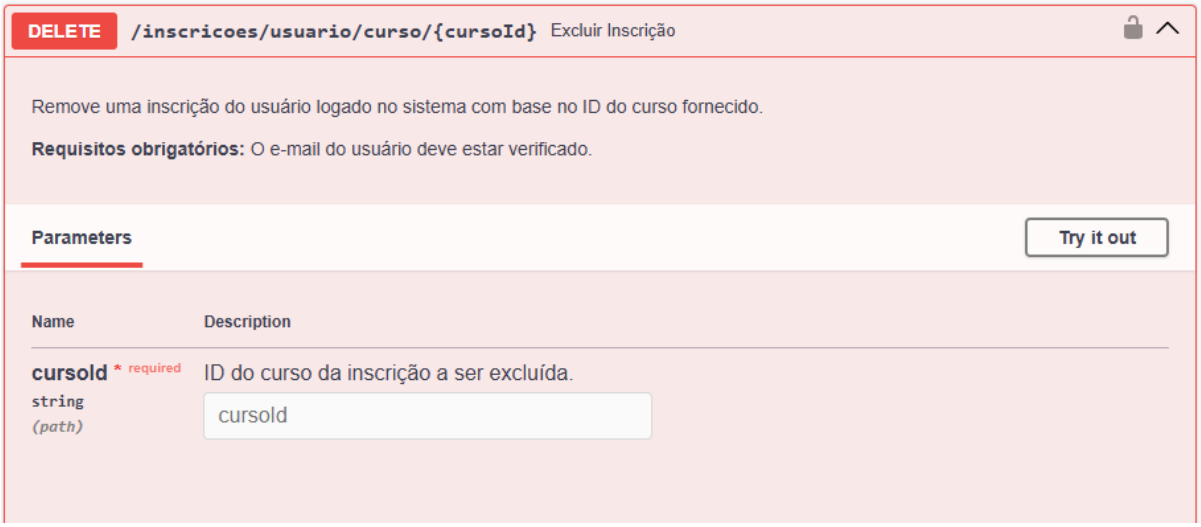
Figura 22 – Requisição GET para Listar Todas as Inscrições do Usuário Logado



Fonte: elaborado pelo autor (2025).

- **DELETE /inscricoes/usuario/curso/{cursoId}:** Rota responsável por deletar a inscrição do usuário logado em um curso específico.

Figura 23 – Requisição DELETE de Inscrição do usuário logado por ID do Curso



Fonte: elaborado pelo autor (2025).

4.5 Implantação

O processo de implantação da solução desenvolvida foi estruturado para garantir disponibilidade, escalabilidade e facilidade de manutenção, utilizando práticas modernas de Integração Contínua/Entrega Contínua e orquestração de contêineres.

O projeto está hospedado em um servidor próprio e é orquestrado por meio do Kubernetes. O fluxo de implantação é automatizado através de um pipeline de CI/CD configurado no GitLab.

Cada *commit* ou *merge* na *branch* principal dispara o pipeline, que automaticamente realiza as etapas de *build* e *deploy* da aplicação nos *clusters* do Kubernetes. Esta abordagem garante que o ambiente de produção esteja sempre sincronizado com o código-fonte mais recente do projeto.

4.5.1 URLs da Aplicação:

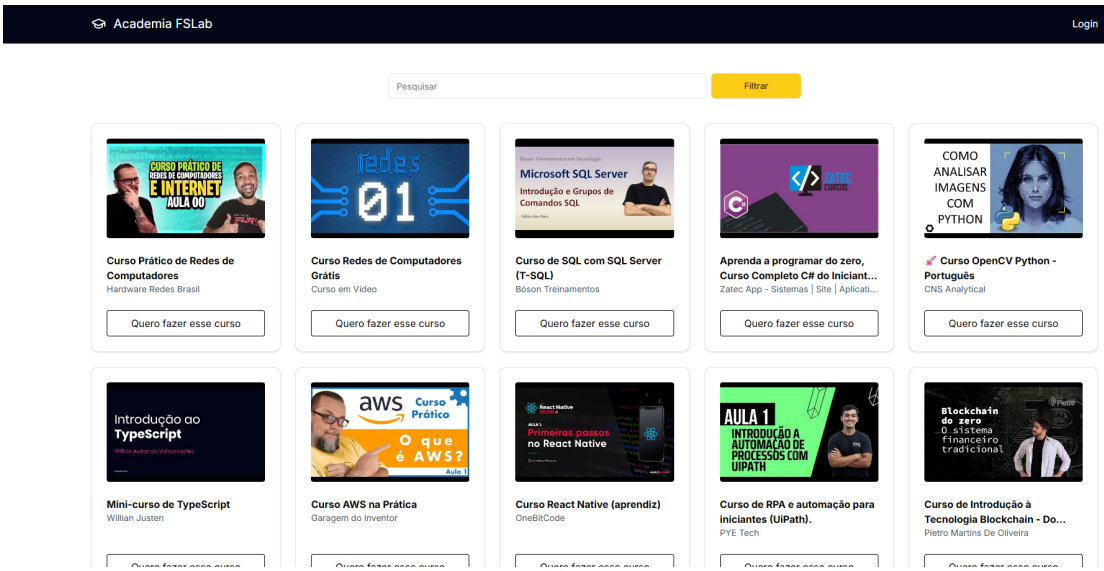
- **API:** <https://api-academia.app.fslab.dev/>.
- **Front-end:** <https://academia.app.fslab.dev/>.

4.6 Demonstração do software

Para a demonstração do sistema, são apresentadas capturas de tela do *front-end*. Ao acessar a aplicação, o usuário é direcionado para a tela inicial (Figura 24), na qual é

possível visualizar e filtrar os cursos disponibilizados. Também é possível realizar a filtragem por nome do curso, nome do instrutor e tag relacionada ao curso.

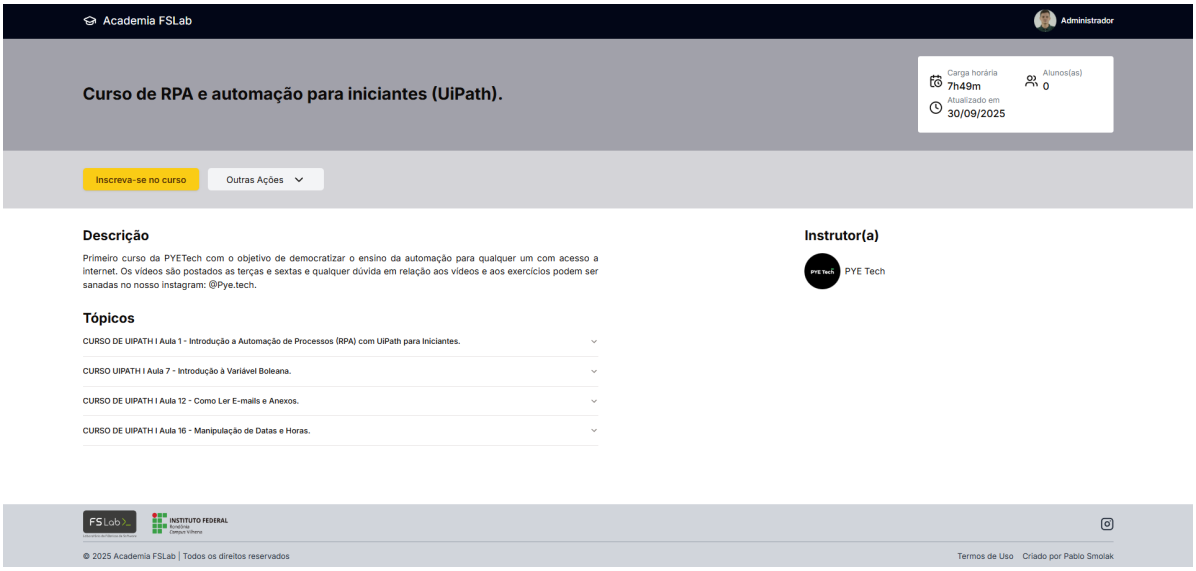
Figura 24 – Página inicial da plataforma com a vitrine de cursos



Fonte: elaborado pelo autor (2025).

Após encontrar um curso de interesse, o usuário pode acessar a página de informações detalhadas, mostrada na Figura 25. Nesta tela, são encontradas diversas informações sobre o curso, como a descrição, os tópicos e atividades que o aluno encontrará, os instrutores, a carga horária, a quantidade de inscritos e a data da última atualização. Com base nesses dados, o usuário pode decidir se inscrever no curso ou retornar à busca.

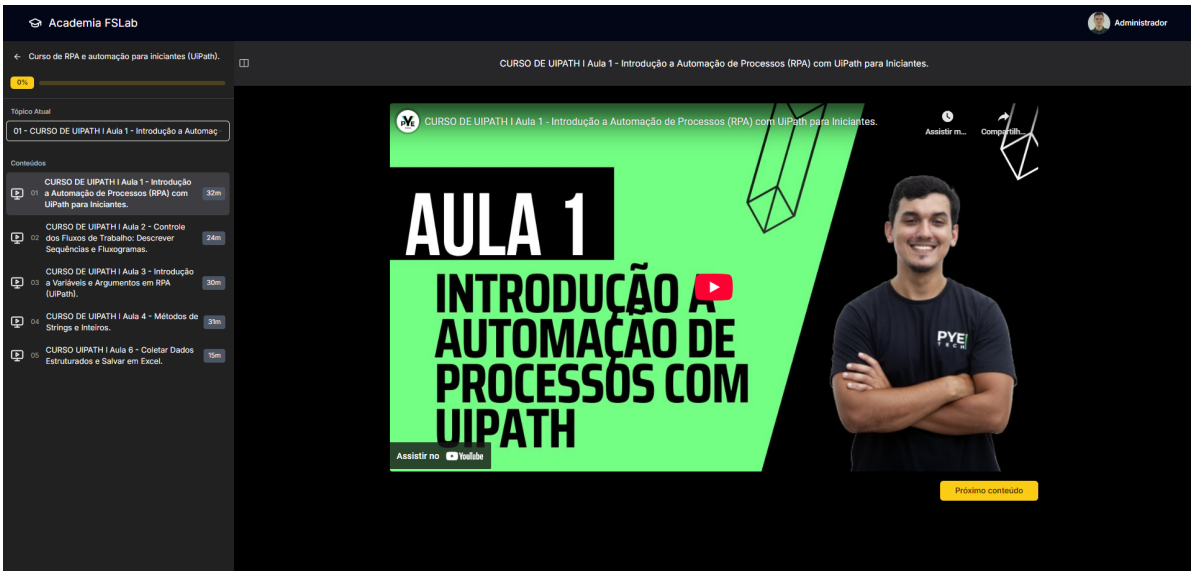
Figura 25 – Página de Detalhes do Curso



Fonte: elaborado pelo autor (2025).

Após a inscrição, o usuário ganha acesso à tela do *player* do curso (Figura 26), o principal ambiente de aprendizado, onde é possível visualizar os conteúdos e marcar as atividades como finalizadas.

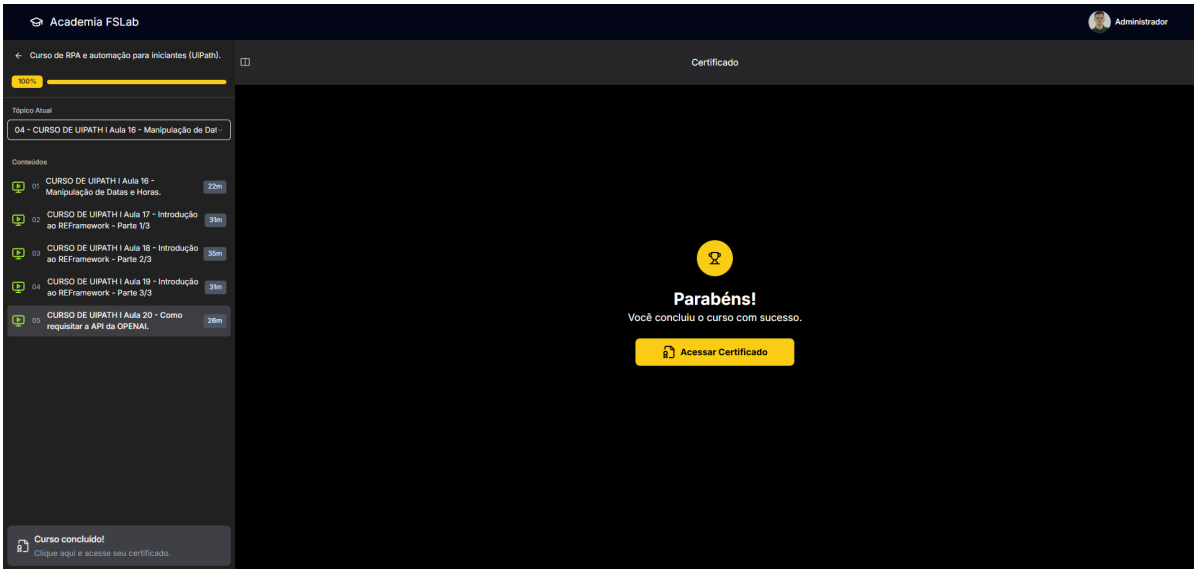
Figura 26 – Página do Player do Curso



Fonte: elaborado pelo autor (2025).

Após finalizar todos os conteúdos disponíveis de um curso, é exibida ao usuário uma tela de parabéns pela conclusão (Figura 27), na qual ele já tem acesso imediato ao seu certificado.

Figura 27 – Página do Player com o curso finalizado



Fonte: elaborado pelo autor (2025).

Na página de certificado (Figura 28), o usuário visualiza o certificado emitido pela

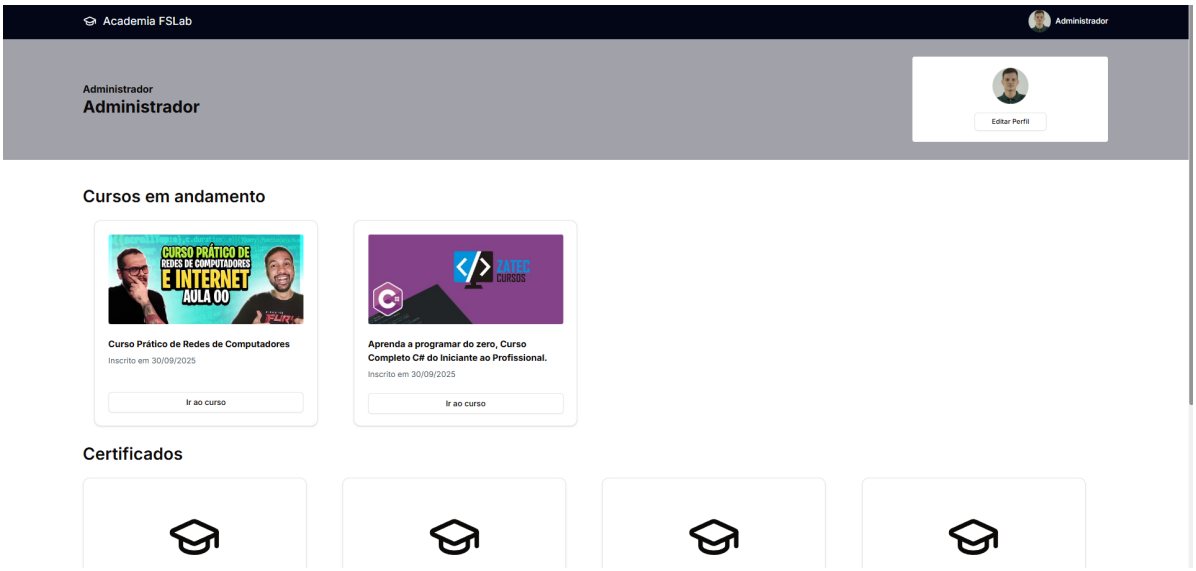
plataforma. O documento contém o nome do aluno, o curso que ele concluiu, a data de conclusão, o nome do emissor e a carga horária do curso finalizado. Nesta página, também é possível baixar o certificado e compartilhá-lo no perfil pessoal do LinkedIn.

Figura 28 – Página de Visualização do Certificado de Conclusão

Fonte: elaborado pelo autor (2025).

Na página de perfil do usuário (Figura 29), são exibidas as informações sobre os cursos em andamento e os certificados obtidos, bem como a funcionalidade para editar os dados pessoais.

Figura 29 – Página de Perfil do Usuário

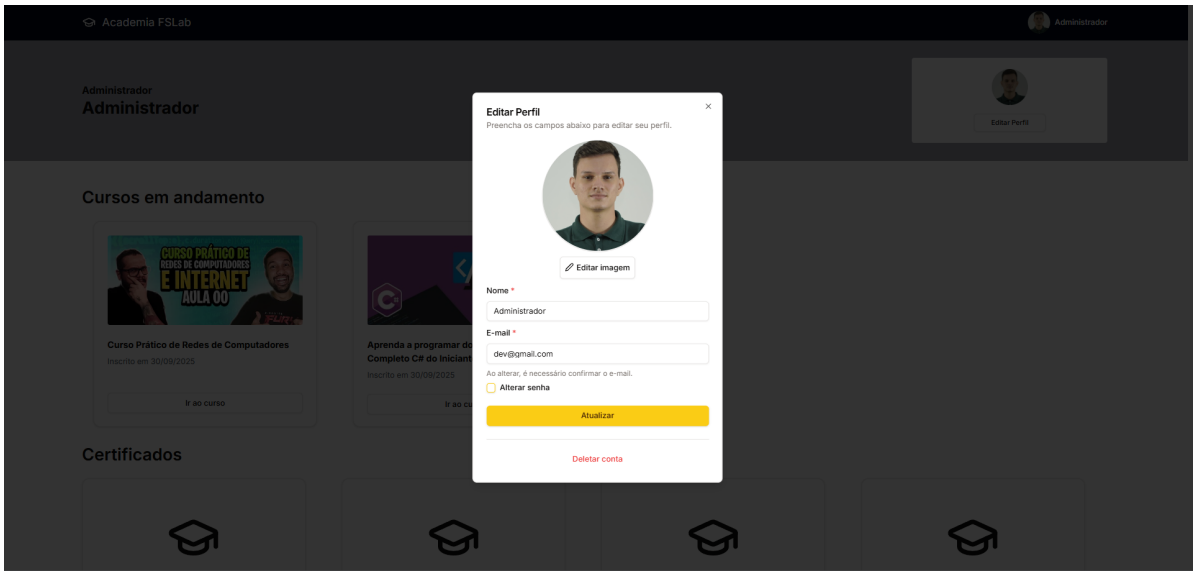


Fonte: elaborado pelo autor (2025).

A funcionalidade de edição de perfil, ilustrada na Figura 30, traz as opções de

editar o nome do usuário, o e-mail e a senha. Adicionalmente, caso o usuário não deseje mais ter uma conta no sistema, a interface oferece a possibilidade de excluir a conta permanentemente.

Figura 30 – Interface para Atualização dos Dados do Perfil



Fonte: elaborado pelo autor (2025).

5 Considerações finais

O desenvolvimento deste software me fez evoluir bastante como profissional, trazendo novos conhecimentos tecnológicos. Antes de iniciar o projeto, meu foco era quase todo no *back-end*, com pouco contato com as tecnologias usadas para o *front-end*, o que trouxe algumas dificuldades no desenvolvimento. Conforme o projeto avançava, essas dificuldades foram sendo resolvidas.

Escolher um projeto completo me mostrou novas formas de pensar no desenvolvimento, percebendo que nem sempre o que planejamos no início é o melhor para o *front-end*. Aventurar-se em novas tecnologias sempre é uma ótima forma de aprender e adquirir novos conhecimentos.

Durante o desenvolvimento, foi possível criar um sistema que inclui autenticação segura, listagem e visualização de cursos, *player* para exibição de conteúdos, emissão de certificados e gerenciamento de perfil. A plataforma foi desenvolvida com tecnologias modernas, como Node.js, Next.js, Prisma e MinIO, e integrada a pipelines de CI/CD no GitLab, garantindo um processo de desenvolvimento automatizado e mais seguro.

Mesmo com o projeto completo, existem algumas limitações. Como o foco do trabalho era nas telas para o aluno, nenhuma tela administrativa foi desenvolvida. No entanto, as rotas necessárias para o gerenciamento foram implementadas e testadas, então é possível usar toda a parte administrativa, embora seja necessário ter conhecimento técnico para acessar diretamente as rotas da API.

Conclui-se, portanto, que o objetivo geral deste trabalho foi alcançado, com a entrega de uma plataforma funcional projetada para ajudar o FSLab na capacitação de novos integrantes. A solução oferece um ambiente moderno e acessível que busca oferecer mais autonomia aos professores e facilitar o aprendizado. O projeto também abre espaço para futuras melhorias e ampliações, podendo servir como referência para outras instituições interessadas em soluções de ensino e treinamento técnico.

5.1 Trabalhos futuros

Após o desenvolvimento da plataforma Academia FSLab, foram identificadas algumas melhorias e expansões que podem ser implementadas em versões futuras do sistema:

- Implementar o painel administrativo completo, permitindo que administradores gerenciem usuários, cursos, categorias e conteúdos.

- Permitir que instrutores criem trilhas ou planos de aprendizado, combinando diferentes cursos já existentes em uma sequência lógica de capacitação.
- Ampliar os tipos de conteúdo dos cursos, permitindo que além de vídeos, os instrutores possam incluir textos, PDFs, quizzes, apresentações e outros recursos multimídia, tornando o aprendizado mais diversificado e dinâmico.
- Desenvolver relatórios detalhados de desempenho e progresso dos alunos, possibilitando o acompanhamento individual e coletivo dos resultados obtidos.
- Adicionar suporte a notificações para lembrar os alunos de concluir os cursos e manter o engajamento.
- Sistema de feedback e avaliações permitindo que alunos avaliem cursos e forneçam feedback aos instrutores, ajudando na melhoria contínua do conteúdo.
- Implementar recomendações inteligentes, sugerindo cursos ou trilhas baseadas no desempenho do aluno ou em seus interesses declarados, tornando o aprendizado mais personalizado e eficiente.

Referências

AGONÁCS, N.; MATOS, J. F. Os cursos on-line abertos e massivos (mooc) como ambientes heutagógicos / massive open online courses (mooc) as heutagogical environments. *Revista Brasileira de Estudos Pedagógicos*, Jun. 2020. Disponível em: <https://www.scielo.br/j/rbeped/a/MKZbj9vMn5SSNzH6wDghZvC/?lang=pt>. Citado na página 16.

ALURA. *Guia de JavaScript: o que é e como aprender a linguagem mais popular do mundo?* 2023. Disponível em: <https://www.alura.com.br/artigos/javascript?srsltid=AfmBOooMR7OrTQO2BaHdnJKYEFR00hp7ByY1KJvEZRo11CQaWMxhSGe9>. Acesso em: 11 set. 2024. Citado na página 20.

ALURA. *Node.JS: o que é, como funciona esse ambiente de execução JavaScript e um Guia para iniciar*. 2023. Alura. Disponível em: <https://www.alura.com.br/artigos/node-js?srsltid=AfmBOoqGsKuojilux-TOmyuJB9h3DquOcnMpmemLCGuDaIyd2zFxTHpI>. Acesso em: 12 set. 2024. Citado na página 20.

ATLASSIAN. *Source Code Management*. 2025. <https://www.atlassian.com/git/tutorials/source-code-management>. Acesso em: 31 out. 2025. Citado na página 33.

FOWLER, M. *UML Essencial: um breve guia para linguagem padrão*. [S.l.]: Bookman editora, 2014. Citado na página 26.

FRANÇA, P. C. S. et al. Satisfação de usuários de cursos de ensino à distância no brasil / user satisfaction of distance learning courses in brazil. *Brazilian Journal of Development*, v. 8, n. 3, p. 16408–16438, Mar. 2022. Disponível em: <https://ojs.brazilianjournals.com.br/ojs/index.php/BRJD/article/view/44853>. Citado na página 15.

IBGE. *Informações atualizadas sobre tecnologias da informação e comunicação*. 2024. Educacao.ibge.gov.br. Disponível em: <https://educa.ibge.gov.br/jovens/materias-especiais/21581-informacoes-atualizadas-sobre-tecnologias-da-informacao-e-comunicacao.html>. Acesso em: 15 dez. 2024. Citado na página 15.

IBM. *Diagrama de Classes*. 2021. Ibm.com. Disponível em: <https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=structure-class-diagrams>. Acesso em: 09 set. 2025. Citado na página 26.

IBM. *Minio*. 2021. Ibm.com. Disponível em: <https://www.ibm.com/docs/pt-br/cloud-private/3.2.x?topic=private-minio>. Acesso em: 13 set. 2024. Citado na página 21.

IBM. *O que é uma API (Interface de Programação de Aplicativos)?* 2024. Ibm.com. Disponível em: <https://www.ibm.com/br-pt/topics/api>. Acesso em: 13 set. 2024. Citado na página 21.

IBM. *O que é uma API REST?* 2025. Ibm.com. Disponível em: <https://www.ibm.com/br-pt/think/topics/rest-apis>. Acesso em: 31 out. 2025. Citado na página 21.

NODE.JS. *Introduction to Node.js*. 2024. Nodejs.org. Disponível em: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs>. Acesso em: 13 set. 2024. Citado na página 21.

OVERFLOW, S. *Developer Survey 2022*. 2022. Disponível em: <https://survey.stackoverflow.co/2022/#technology-most-popular-technologies>. Acesso em: 12 set. 2024. Citado 2 vezes nas páginas 20 e 21.

REDAÇÃO LYCEUM. *MOOC (Massive Online Open Course): entenda o que é e como funciona*. 2019. Blog Lyceum. Disponível em: <https://blog.lyceum.com.br/o-que-e-mooc/>. Acesso em: 15 dez. 2024. Citado na página 16.

SANTOS, I. et al. Possibilidades e limitações da arquitetura mvc (model–view–controller) com ferramenta ide (integrated development environment). *RE3C-Revista Eletrônica Científica de Ciência da Computação*, v. 5, n. 1, 2010. Citado 2 vezes nas páginas 25 e 26.

SERVICES, I. A. W. *O que é uma API? – Explicação sobre interfaces de programação de aplicações*. 2023. AWS. Disponível em: <https://aws.amazon.com/pt/what-is/api/>. Acesso em: 13 set. 2024. Citado na página 21.

VERCEL. *Geração de Site Estático (SSG)*. 2025. <https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation>. Acesso em: 7 out. 2025. Citado na página 26.

VERCEL. *Renderização do lado do servidor (SSR)*. 2025. <https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering>. Acesso em: 7 out. 2025. Citado na página 26.

Anexos

ANEXO A – Licença MIT