Ericky Moreno Mamede

Desenvolvimento de parte da API para plataforma de cursos do IFRO

Vilhena - RO

2022

Ericky Moreno Mamede

Desenvolvimento de parte da API para plataforma de cursos do IFRO

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – campus Vilhena, realizado em cumprimento de requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Gilberto Pereira da Silva

Vilhena - RO
2022

FICHA CATALOGRÁFICA

Biblioteca IFRO – Campus Vilhena

M264d

MAMEDE, Ericky Moreno
Desenvolvimento de parte da API para plataforma de cursos do IFRO / Ericky Moreno Mamede – Vilhena, Rondônia, 2021.

67f. ; il.
Orientador Prof. Esp. Gilberto Pereira da Silva

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

1. API 2. EAD 3. FSLab 4. Kanban 5. Plataforma de cursos I. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia – IFRO II. Título

006.7882

Bibliotecária responsável Rosilene Maria do Couto Marques CRB 11/321

Este trabalho é dedicado à toda minha família, amigos, docentes e demais que ajudaram efetivamente na realização do mesmo.

Agradecimentos

Em primeiro lugar, a Deus, que fez com que meus objetivos fossem alcançados, durante todos os anos já vividos.

Agradeço aos meus pais, Eduardo e Zenaide, meus heróis que me deram apoio, me incentivaram nas horas difíceis, de desânimo, cansaço, dentre outros. A minha irmã, Emilly, por todo carinho, companheirismo e incentivo durante toda a vida. A minha tia Célia que não mediu esforços para a minha formação. Também a minha tia Lília que durante os últimos anos tornou-se uma segunda mãe. Ao meu tio Gilberto, por todo apoio, paciência, incentivo e conselhos passados durante esse processo. A toda minha família faço os meus agradecimentos.

Agradeço ao professor orientador Gilberto Pereira da Silva, por toda dedicação, confiança, paciência e incentivo nesta, trajetória de estudos. E aos professores Juliano Fischer Naves e Marco Antonio Augusto de Andrade deixo os meus mais sinceros agradecimentos. De forma geral, agradeço a todos os professores por me proporcionarem o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender. A vocês os meus eternos agradecimentos.

Ao meu amigo, Wandreus, por todo companheirismo nas horas difíceis e nos momentos de diversão, bem como pelo incentivo dedicado e colaboração, durante todo o curso. Aos demais colegas, em especial meu amigo Jackson, por toda colaboração e parceria, ao longo do curso.

O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.

José de Alencar

Resumo

Esta monografia apresenta o processo de desenvolvimento de parte de uma API (*Application Programming Interface* - Interface de Programação de Aplicações) para plataforma de cursos do IFRO (*Instituto Federal de Educação, Ciência e Tecnologia de Rondônia*). Tem como objetivo desenvolver uma parte da API para apresentar informações referentes aos cursos criados pelo instituto. Durante o desenvolvimento deste projeto, foram usadas tecnologias recorrentes no mercado atual, como JavaScript, Node.js, Docker e MongoDB. Para o gerenciamento das atividades, foi escolhido a utilização do método Kanban, sendo gerado métricas para controle do desenvolvimento. Portanto, ao fim deste trabalho espera-se que o desenvolvimento da API possa continuar, com o propósito de alimentar o site e o aplicativo a serem desenvolvidos.

Palavras-chave: Plataforma de Cursos. API. *Kanban*. FSLab. EAD.

Abstract

This monograph presents the process of developing part of an API (*Application Programming Interface*) for the IFRO (*Instituto Federal de Educação, Ciência e Tecnologia de Rondônia*) course platform. It aims to develop a part of the API to present information regarding the courses created by the institute. During the development of this project, recurrent technologies in the current market were used, such as JavaScript, Node.js, Docker and MongoDB. For the management of activities, the use of the Kanban method was chosen, generating metrics to control the development. Therefore, at the end of this work, it is expected that the development of the API can continue, with the purpose of feeding the website and the application to be developed.

Keywords: Plataforma de Cursos. API. *Kanban*. FSLab. EAD.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
EAD	Ensino a distância
FSLab	Laboratório de Fabrica de Software
HTTP	HyperText Transfer Protocol
ID	Identificador
IFRO	Instituto Federal de Educação, Ciência e Tecnologia de Rondônia
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
JS	JavaScript
JSON	JavaScript Object Notation
MEC	Ministério da Educação
RF	Requisitos Funcionais
RQ	Requisitos Não Funcionais
UML	Unified Modeling Language
URL	Uniform Resource Locator
UUID	Universally Unique IDentifier
VM	Virtual Machine
YAML	YAML Ain’t Markup Language

Sumário

1 Introdução

1.1 Contexto e problema

O Ensino a distância (EAD) não é apenas uma realidade atual, seus primeiros registros datam de 1728, quando um professor norte-americano chamado Caleb Phillips enviava conteúdos sobre Taquigrafia através de cartas para seus alunos. Atualmente, com a internet, as distâncias geográficas foram derrubadas, permitindo o ensino a distância (EAD) chegar a todos os lugares.

O EAD tem se mostrado uma crescente tendência na capacitação das pessoas, o Censo da Educação Superior 2019, divulgado pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep) e pelo Ministério da Educação (MEC), mostra que houve um aumento de 378,9% no número de matrículas entre os anos de 2009 e 2019. Isso pode ter ocorrido pelos benefícios que o EAD trás, como, por exemplo a possibilidade do aluno em adaptar o ritmo de aprendizagem de acordo com o seus horários e estudar de qualquer lugar que tenha acesso a internet.

Não apenas no ensino superior tem se destacado o ensino digital, mas também em cursos de formação complementares. Esses cursos se sobressaem por serem objetivos e de curta duração, auxiliando ou adicionando novas habilidades na formação profissional das pessoas.

Os cursos complementares, normalmente, são ofertados por instituições de ensino superior ou empresas privadas especializadas na área. Essas empresas cobram por assinaturas, por curso, ou ainda apenas o certificado de conclusão do curso. As instituições de ensino, tendem a ofertar gratuitamente os cursos e seus devidos certificados de conclusão.

Quando desenvolvidos com um objetivo pré-definido, os cursos complementares auxiliam não só na formação dos alunos de nível superior, mas também de nível básico. Sendo assim, o Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO), por meio do Laboratório de Fábrica de Software (FSLab), espera desenvolver uma nova plataforma de cursos gratuitos para auxiliar a formação dos discentes. Inicialmente, a plataforma será disponibilizada apenas no ambiente interno do instituto, no entanto, presume-se que futuramente possa ser aberta a toda população.

Dessa forma, pretende-se atender de forma mais efetiva os interesses e necessidade dos alunos e profissionais do instituto. Visa-se acolher também a necessidade das horas complementares obrigatórias para a conclusão dos cursos superiores.

A partir desse contexto nota-se um problema: Como melhorar o acesso dos alunos a cursos complementares gratuitos, com boa qualidade e certificação válida?

1.2 Objetivos

1.2.1 Objetivo geral

Este trabalho tem como objetivo geral desenvolver uma API para a plataforma de cursos do IFRO, permitindo a melhora na oferta de cursos on-line por parte do instituto.

1.2.2 Objetivos específicos

Os objetivos específicos são os passos para se atingir o objetivo geral:

- Efetuar levantamento de requisitos ;
- Entender o problema em questão ;
- Elaborar diagrama de classe;
- Elaborar diagrama de arquitetura de software;
- Desenvolver um protótipo visual ;
- Criar o banco de dados;
- Documentar e desenvolver a API;
- Realizar testes automatizados;
- Disponibilizar API em ambiente de produção para que seu comportamento possa ser verificado.

1.3 Justificativa

O ensino digital tem ganhado cada vez mais espaço na formação das pessoas. Acredita-se que o aumento em cursos a distância foi influenciado pela pandemia do Covid-19, que teve início a partir de março de 2019. De acordo com o Censo do Inep, houve-se um aumento de mais de 30% na oferta de cursos a distância no período de 2019 para 2020, chegando a ser ofertado 13,5 milhões de vagas. Neste mesmo período, o aumento de vagas presenciais foi de apenas 1,3%.

Levando em conta a crescente influência do ensino digital, o Instituto Federal de Educação, Ciência e Tecnologia de Rondônia (IFRO) vê a necessidade dos alunos em acessar cursos on-line e emitir certificados para horas complementares. Assim, uma nova plataforma de cursos do próprio instituto, possibilitará aos alunos acesso fácil e rápido a cursos de confiança. Tal plataforma possibilitará também aos docentes o reaproveitamento de conteúdo para reforçar o aprendizado dos seus discentes. Dessa forma, este trabalho desenvolve uma parte da API da plataforma de cursos do IFRO, espera-se que, a partir dele, outros trabalhos possam dar continuidade para a finalização e implantação da API.

2 Fundamentação teórica

2.1 Educação a Distância

O EAD (Ensino a Distância) tem se apresentado como uma eficiente ferramenta de aprendizagem, pois possibilita o acesso à ensino-aprendizagem de qualquer lugar que tenha acesso a internet, de casa, do parque, zona rural, etc. Sem que seja necessário o deslocamento do tutor ou aluno. Deste modo pode-se realizar cursos técnicos de graduação, especialização, e outros mais. Tudo isso por meio da TICs (Tecnologias da Informação e Comunicação). (COSTA, 2017).

Ademais, as TICs têm proporcionado mais autonomia aos alunos para que desenvolvam pesquisas independentes, e possam aperfeiçoar cada vez mais os seus estudos sem que seja necessário a presença física de um professor. Houve uma melhora e agilidade considerável às mudanças, juntamente ao acesso aos cursos EAD. Tudo isso em virtude da criação e os avanços da internet, e as facilidades propostas pela mesma. (ANDRADE et al., 2017).

Com o EaD o aluno tem mais flexibilidade, facilidade e autonomia para administrar o acesso aos cursos, e também qual é melhor ritmo para sua aprendizagem. Não obstante que seja um aprendizado a distância, todavia o aluno pode interagir com os demais colegas através de fóruns, chats, e etc. e tem o auxílio constante de um dos seus professores.

O Ensino a distância pode ser utilizado também para cursos de graduação ou extensão. A união entre ensino, pesquisa e universidade, permitem a extensão universitária efetiva e contribuem para transformar a sociedade, através do conhecimento; Por envolver alunos, docentes e comunidades. Contudo a extensão no ensino facilita o processo pedagógico, ampliando a aprendizagem, somando na politização do ensino com viés social na produção do conhecimento. Sendo assim, extensão deve ser considerada, uma construtora de autonomia, autodesenvolvimento, autoaprendizagem e espaço de vivências, em que se há relações com o outro e com toda a conjuntura social (FRANÇA et al., 2022).

Entretanto apesar do EAD ser um ensino a distância o mesmo deve estar dentro dos parâmetros exigidos pelo MEC (Ministério da Educação), das quais devem estar dentro do conceito de qualidade no país sendo constituído a partir de três funções: avaliação, regulação e supervisão das instituições e dos respectivos cursos superiores (NOBRE; NAVES, 2015).

Conquanto, no Ensino a Distância se faz fundamental que os tutores concedam toda atenção possível às atividades propostas, corrigindo as tarefas e exercícios cuidadosamente o mais rápido possível, para que assim, o mesmo faça o acompanhamento necessário e tenha possibilidade de intervir na aprendizagem dos seus alunos. Ao avaliar o ensino-aprendizagem o professor pode avaliar o grau de satisfação dos seus alunos referente ao curso por meio de

métodos estatísticos, fichas de avaliação e de observação. (MACHADO; MACHADO, 2004).

Observa-se que os alunos têm sua satisfação influenciada por alguns fatores relacionados a planejamento, implantação e manutenção em um curso a distância.

Faz-se perceptível que o AVA (Ambiente Virtual de Aprendizagem) necessita ter um design prático e flexível de fácil acesso. Que tenha um bom suporte atendendo às necessidades e mantendo alunos e professores com uma interação satisfatória, mas também que instigue o aluno a despertar seu interesse pela aprendizagem e deixá-lo mais próximo da sua instituição de ensino (SILVA; MELO; MUYLDER, 2015).

FRANÇA et al. (2022) destaca a importância dos Ambientes Virtuais de Aprendizagem:

> Os Ambientes Virtuais de Aprendizagem (AVAs) são plataformas constituídas por estruturas de funções que possibilitam o desenvolvimento dos cursos à distância. Tal plataforma foi desenvolvida pela evolução tecnológica e expansão da internet. Um exemplo dessa plataforma éo Moodle (Modular Object-Oriented Dynamics Learning Environment), um ambiente gratuito construído para auxiliar no processo de EAD (FRANÇA et al., 2022 apud KUHN; HOFLER; SILVA, 2017, p.86-114). As plataformas virtuais auxiliam os alunos dos cursos à distância, com fornecimento de material didático, como vídeo aulas, slides, referências bibliográficas, chats interativos entre professores e alunos para se sanar dúvidas relacionadas ao conteúdo das disciplinas dos cursos, entre outras funções, sendo assim, muito útil nessa modalidade de ensino (FRANÇA et al., 2022, p.3).

Projetado por Martin Dougiamas, em 2001 o MOODLE, é um ambiente virtual que permite a comunidades on-line a administração de atividades educacionais com foco na aprendizagem colaborativa, que permite de uma forma simples e prática a interação de alunos e professores. (VASCONCELOS; JESUS; SANTOS, 2020).

Com as dificuldades impostas pela Covid-19, referentes às aulas lecionados fisicamente em sala, o aprendizado a distância ganhou um espaço emergencial quantioso, havendo assim uma considerável adesão ao modelo EAD, com isso há uma maior expectativa de que o os cursos a distância conquistem ainda mais espaço no auxílio da formação de cada vez mais indivíduos

2.2 MOOCs

Os MOOCs (massive open online courses - curso on-line aberto e massivo) são cursos pensados para alcançar um grande número de pessoas. Diferentemente do ensino tradicional, onde o número de matrículas é limitado para garantir que os tutores consigam interagir de forma efetiva com os alunos, os Moocs não tendem a limitar as vagas e alcançam o maior número de pessoas possíveis. A construção de um Mooc é pensada para que essa interação se faça desnecessária e o aluno avance de forma autônoma (PAPPANO, 2012).

De forma geral, os Mooc baseiam-se em modelos pedagógicos e em teorias de aprendizagem bem conhecidas e estabelecidas como, por exemplo, o comportamentalismo, o cognitivismo,

o conectivismo etc.(AGONÁCS; MATOS, 2020). Os Moocs caracterizam-se por serem massivos, portanto, qualquer pessoa com acesso a internet pode cursá-los. Essa abordagem é uma das formas mais efetivas de tornar a educação gratuita e democrática.

2.3 Application Programming Interface

Uma API (*Application Programming Interface*) é uma série de regras que discriminam como dispositivos ou aplicativos podem se comunicar uns com os outros. As APIs permitem que aplicativos ou serviços acessem recursos dentro de outros serviços ou aplicativos. O aplicativo ou serviço que realiza o acesso é chamado de cliente, e o aplicativo ou serviço que contém o recurso é chamado de servidor (IBM, 2021).

FIELDING (2000) descreve o que é uma API REST e quais os seus propósitos:

> REST é um conjunto coordenado de restrições arquitetônicas que tenta minimizar a latência e a comunicação de rede ao mesmo tempo que maximiza a independência e escalabilidade de implementações de componentes. Isso é alcançado colocando restrições na semântica do conector onde outros estilos se concentraram na semântica dos componentes. REST permite o armazenamento em cache e a reutilização de interações, dinâmicas substituibilidade de componentes e processamento de ações por intermediários, atendendo assim às necessidades de um sistema de hipermídia distribuído na Internet. (FIELDING, 2000, p.1).

2.4 Linguagem de Programação JavaScript

JavaScript (frequentemente abreviado para JS) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, também conhecida como linguagem de script para páginas web. Além disso, a linguagem JS pode ser usada em vários outros ambientes sem serem navegadores, como por exemplo, *Node.js*. O JavaScript é uma linguagem dinâmica de multi-paradigma, baseada em protótipos que suportam estilos de orientação a objetos imperativos e declarativos (MOZILLA, 2022) .

2.4.1 Ambiente de execução

A linguagem JavaScript, assim como todas as outras, necessita de um ambiente de execução. Sendo esses, plataformas de execução específicas, como um sistema de gerenciamento de banco de dados ou um sistema operacional. Assim, é possível descrever o contexto no qual a execução de um modelo ocorre (IBM, 2015).

Para este projeto foi escolhido o ambiente de execução Node.js. O Node.js é um *runtime*, baseado no núcleo do Google Chrome v8, para JavaScript. Um aplicativo Node.js é executado em apenas um único processo, sem criar um novo *thread* para cada solicitação. É fornecido um conjunto de primitivas de entrada/saída (E/S) assíncronas em sua biblioteca padrão que

impedem o bloqueio do código JavaScript e, geralmente, as bibliotecas no Node.js são escritas usando paradigmas sem bloqueio, tornando o comportamento de bloqueio uma exceção e não uma norma (NODE, 2022).

2.5 Conteinerização

Pensando na flexibilidade e escalabilidade do sistema, o mesmo foi desenvolvido com base na arquitetura de conteinerização, sendo cada parte do sistema executado em um Contêiner diferente.

Contêineres são pacotes de software que agrupam o código de um aplicativo e respectivos arquivos de configuração e bibliotecas necessárias para a execução do aplicativo. Dessa forma, os desenvolvedores e os profissionais de TI podem implantar aplicativos diretamente nos ambientes (AZURE, 2022).

Para o gerenciamento dos contêineres foi escolhido o *software open-source* Docker. O Docker é um sistema para automatizar a implantação de aplicativos como contêineres autossuficientes portáteis que podem ser executados na nuvem ou localmente (MICROSOFT, 2021).

2.6 Banco de dados

Quando falamos em banco de dados é comum pensarmos em bancos relacionais, como MySQL ou Mariadb. No entanto, quando necessitamos de grande escalabilidade bancos relacionais tendem a ter um custo bem elevado. Visando maior flexibilidade e escalabilidade, optou-se por utilizar uma solução com banco de dados não relacional (NoSql) para essa aplicação.

Para essa plataforma, foi implementado um banco de dados NoSql orientado a documentos. Normalmente, os bancos de dados não relacionais armazenam qualquer tipo de arquivo binário enviado, porém, bancos de dados orientados a documentos armazenam arquivos com estruturas pré-definidas. Ainda que essas estruturas sejam semanticamente variáveis, os bancos de dados orientados a documentos se assemelham um pouco com bancos relacionais (TOTH, 2011).

O MongoDB é um servidor *open-source* de banco de dados que permite o armazenamento de dados em documentos do tipo JSON. Essa abordagem permite uma maior flexibilidade, escalabilidade horizontal e também um melhor desempenho (MONGODB, 2021).

2.7 Versionamento

No desenvolvimento de *software* é comum utilizarmos sistemas de gerenciamento de versões que possibilitam o controle de versões dos códigos desenvolvidos. O versionamento de

código-fonte fornece ao analista a possibilidade de controlar os responsáveis por cada mudança e segmentar as entregas/módulos em ramos diferentes do projeto, podendo assim serem testadas separadamente todas as alterações decorridas ao longo do processo (DIAS, 2014).

O Git é um Sistema de Controlo de Versões (SCV), desenvolvido em 2005 por Linus Torvalds. Os SCV são softwares que ajudam a administrar o controle de versão de códigos, principalmente em projetos de software (MURÇA et al., 2020).

O GitLab é construído baseado em git, permitindo que usuários que contribuem para um projeto tenham uma uma cópia do projeto baixada/retirada/clonada em seu computador local. Atuando como uma fonte única de verdade, o GitLab pode evitar conflitos e a dupla manipulação de trabalho, agilizando e facilitando o trabalho de desenvolvimento do código (O'GRADY, 2018).

2.8 Metodologia ágil

As metodologias ágeis ganharam espaço a partir de 2001, quando um grupo de dezessete desenvolvedores se empenharam em encontrar formas mais eficientes para o desenvolvimento de *software*. Foram elaborados quatro valores e doze princípios, que ficaram conhecido como Manifesto Ágil. Sendo os quatro valores:

- **Indivíduos e interações** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação de contratos;
- **Responder a mudanças** mais que seguir um plano.

A utilização do texto em negrito remete a ideia de que, por mais que haja valores nos itens à direita, deve haver uma maior valorização nos itens à esquerda (BECK et al., 2001).

2.8.1 Kanban

Para o desenvolvimento dessa aplicação foi aderido ao método Kanban. Após a segunda guerra mundial, o Japão, passando por uma crise econômica, buscava estratégias para reduzir os custos e aumentar a produtividade. Foi então que a empresa Toyota criou a técnica Kanban, que significa cartão ou sinalização. Na década de 1960, esses cartões coloridos ajudaram a controlar o estoque e a produção(SILVA, 2019).

O Kanban é um método que permite diversas abordagem, mas a maioria dos especialistas concordam que o Kanban é um método de gestão de mudança, que dá ênfase aos seguintes princípios:

- Visualizar o trabalho em andamento;
- Visualizar cada passo em sua cadeira de valor do conceito geral até software que se possa lançar;
- Limitar o Trabalho em Progresso (Work in Progress), restringindo o total de trabalho permitido para cada estágio;
- Tornar explícitas as políticas que são seguidas;
- Medir e gerenciar o fluxo, para poder tomar decisões bem embasadas, além de visualizar as consequências dessa decisões;
- Identificar oportunidades de melhorias, criando uma cultura Kaizen, a qual a melhoria contínua é responsabilidade de todos.

Tudo isso, com a filosofia subjacente de que se deve:

- Começar com o que se está fazendo agora;
- Concordar em buscar mudanças incrementais e evolucionárias;
- Respeitar o processo atual, com papéis, responsabilidades e cargos.

O método Kanban traz uma abordagem de sistema "puxado"(pull), os *cards* são "puxados"em um fluxo constante que foi definido. Dessa forma, o Kanban serve como catalisador para entregas de valor de sistemas de software (BOEG, 2012).

2.9 Trabalhos similares

As seguintes plataformas de cursos, foram encontrados durante esta pesquisa:

- ALURA: a maior plataforma brasileira de cursos de tecnologia. Para ter acesso aos cursos da Alura é necessário a contratação de uma assinatura anual. Após a expiração do plano, o usuário perde acesso à plataforma. Logo, não tem acesso aos cursos. Tendo em vista o valor dos planos ofertados, a Alura se torna pouco viável para instituições como o IFRO.
- COURSERA: é uma empresa de tecnologia educacional norte-americana fundada pelos professores de ciência da computação Andrew Y. Ng e Daphne Koller, da Universidade Stanford. A plataforma conta com ótimos cursos práticos e objetivos, porém, para obter um certificado autenticado é necessário pagar uma taxa que varia de 20 a 30 dólares.
- UDEMY: é um marketplace voltado para o ensino e aprendizado. A Udemy é uma plataforma aberta com cursos gratuitos e pagos. Diferentemente da Alura, a Udemy vende cada curso separadamente. Um ponto negativo dessa abordagem é a insatisfação gerada

no momento de assistir ao curso, isso porque, não há um controle de qualidade dentro da plataforma.

- MOODLE: é um sistema *open-source* de gerenciamento de aprendizado on-line personalizável, flexível, seguro e acessível. Com ele, educadores podem criar e organizar ambientes de ensino com uma ampla gama de recursos, por meio de plugins ou integrações certificadas.

- IFRO: O MOOC IFRO é uma plataforma aberta e massiva, onde se oferece uma pequena diversidade de cursos com a dinâmica de ensino através de vídeos e materiais complementares. Atualmente, a plataforma passa por diversos problemas técnicos e muitos cursos foram removidos do ar.

3 Materiais e métodos

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

3.1 Materiais

Foi escolhida a linguagem de programação JavaScript para o desenvolvimento deste trabalho[1], dado a sua flexibilidade, dinamismo e o aumento da sua popularização nos últimos anos. Outro aspecto que influenciou na escolha da mesma, foi o futuro desenvolvimento do front-end com a biblioteca React.Js[2] que, como o próprio nome sugere, utiliza JavaScript para o desenvolvimento. Dessa forma, garantimos a padronização no nos projetos da FSLab, facilitamos manutenções e atualizações futuras e o reaproveitamento das equipes.

Para o uso de JavaScript em *server-side* é necessário um ambiente de execução. Portanto, foi escolhido o interpretador de código JavaScript Node.JS [3], pois, é o principal *software open-source* do cenário. Ao utilizarmos o Node.JS, temos acesso ao maior repositório de códigos do mundo, o NPM (*Node Package Manager* - Gerenciador de Pacotes do Node)[4]. Dessa forma, podemos reutilizar pacotes de forma gratuita e agilizar o desenvolvimento de sistemas.

Ao adotarmos o Node.JS para o desenvolvimento de aplicativos web, é comum fazermos uso de algum *framework* para estruturar, acelerar e facilitar o processo. Neste caso, utilizamos o *framework* Express[5], sendo esse, um dos maiores e mais utilizados para o Node.JS. Algumas das suas principais características são: gerenciamento de requisições HTTP com os seus diversos verbos, sistema de rotas completo e possibilita o tratamento de exceções dentro da aplicação.

3.1.1 Persistência de dados

Na computação, a persistência de dados pode ser definida como a garantia de que um dado foi salvo e que poderá ser recuperado quando necessário. Essa técnica é importante, pois permite que as informações atuais dos usuários sejam completamente armazenadas com segurança. Dessa forma, as empresas e instituições podem fazer uso dos dados persistentes em seus sistemas para a tomada de decisão nos seus negócios.

Atualmente, a técnica mais utilizada para a persistência de dados é o uso de um sistema de banco de dados, entretanto, pode se usar outras ferramentas, como um arquivo de texto,

1 Disponível em https://developer.mozilla.org/pt-BR/docs/Web/JavaScript
2 Disponível em https://pt-br.reactjs.org/
3 Disponível em https://nodejs.org/pt-br/
4 Disponível em https://www.npmjs.com
5 Disponível em https://expressjs.com

planilhas, etc. Visto que, o sistema trabalha de forma automatizada e com grande volumes de dados, optou-se por utilizar um servidor de banco de dados nesse trabalho.

Foi utilizada a técnica de *conteinerização* do banco de dados. A conteinerização possibilita maior flexibilidade, escalabilidade e agilidade para as aplicações.

Para esse projeto foi escolhido o *software open-source* Docker[6]. Ele é responsável pela construção e gerenciamento dos contêineres. Em conjunto com o Docker, foi utilizado a ferramenta Docker Compose, possibilitando a inicialização de multi-contêineres a partir de um arquivo YAML (*YAML Ain't Markup Language* - YAML não é linguagem de marcação). A figura 1 apresenta o arquivo *docker-compose.yaml*, responsável pelas informação do contêiner do banco de dados que é utilizado.

Figura 1 – Docker Compose

Para prover o serviço de armazenamento de dados, foi escolhido o MongoDB [7], sendo esse um dos principais serviços de banco de dados orientado a documentos. Sua escolha se deu

6 Disponível em https://www.docker.com
7 Disponível em https://www.mongodb.com/pt-br

pelo fato da aplicação não necessitar de muitos relacionamentos e por conta do desempenho alcançado com o MongoDB.

Para manipulação dos dados, utilizou-se a biblioteca do Node Mongoose [8]. O Mongoose é um ODM (*Object Data Mapping* - Mapeamento de dados de objeto) que permite a criação de soluções para modelagem dos dados por meios de esquemas. Ele possui ainda sistemas de conversão de tipo, criação de consultas, validação e hooks para lógica de negócios. Com Mongoose, os dados do banco de dados são transformados em objetos para JavaScripts, podendo assim, serem utilizados na aplicação.

No Mongoose, cada Schema mapeia uma coleção no MongoDB e define a forma dos documentos dentro dessa coleção (KARPOV et al., 2022). Sendo assim, criou-se dois Schemas para esse sistema, um para usuário e outro para curso.

A figura 2 demonstra como o Schema de Usuário está estruturado. Nesse Schema é indicado o tipo do campo, se são requeridos e se são únicos dentro da base de dados. Nota-se que o campo *cursa* é um array contendo objetos, logo, pode ser considerado como um documento embutido. Em caso do usuário ser um administrador ou um instrutor, os campos administrador, instrutor e leciona são adicionados no Schema do banco, sendo os dois primeiros do tipo booleano e o último um array contendo quais cursos o instrutor leciona.

[8] Disponível em https://mongoosejs.com/

```
const usuarioSchema = new Schema({
  nome: {
    type: String,
    required: true
  },
  dataDeNascimento: {
    type: Date,
    required: true
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  senha: {
    type: String,
    required: true,
    select: false
  },
  fotoDePerfil: {
    type: String,
    required: false
  },
  cursa: [{
    id_curso: {
      type: mongoose.ObjectId,
      required: true
    },
    status: {
      type: String,
      required: true
    }
    ,
    aulas: [{
      id_aula: {
        type: mongoose.ObjectId,
        required: true
      },
      status: {
        type: String,
        required: true
      },
      topicosConcluidos: {
        type: Array,
        required: true
      }
    }],
    certificado: {
      type: String,
      required: true
    },
    dataDeinicio: {
      type: Date,
      required: true

}   }
```

Figura 2 – Schema Usuário

Assim como no Schema de usuário, o Schema de curso modela os dados dentro da coleção, portanto, repete-se o processo de indicar a tipagem do campo, se são requeridos e se são únicos dentro do banco de dados. A figura 3 apresenta as devidas propriedades dos campos.

```
const cursoSchema = new Schema({

  nome: {
    type: String,
    required: true
  },
  duracao: {
    type: Number,
    required: true
  },
  dataDeCriacao: {
    type: Date,
    require: true
  },
  ultimaAtualizacao: {
    type: Date,
    required: false
  },
  descricao: {
    type: String,
    required: true
  },
  tag: {
    type: Array,
    required: false
  },
  aulas: [{
    nome: {
      type: String,
      required: true
    },
    descricao: {
      type: String,
      required: true
    },
    topicos: [{
      titulo: {
        type: String,
        required: true
      },
      conteudo:{
        type: String,
        required: true
      }
    }]
  }]
})
```

Figura 3 – Schema Curso

Vale lembrar que, os Schemas para banco de dados NoSql não são uma estrutura rígida, são flexíveis permitindo que novos campos sejam adicionados ou removidos em um documento. Dessa forma, os desenvolvedores devem cuidar para que não haja perda ou corrompimento dos dados.

Inicialmente, a aplicação estava sendo desenvolvida com Flask[9] - um framework para desenvolvimento web escrito em Python - e utilizaria o banco de dados MySQL [10].Entretanto, após uma mudança nos requisitos sda aplicação, a mesma teve que ser reescrita na linguagem JavaScript. Aproveitando-se dessa mudança, foi decidido, juntamente com o *product owner*, que a utilização de um banco de dados não relacional seria mais viável.

3.2 Métodos

3.2.1 Versionamento

Para o controle de versão do código-fonte foi utilizado um repositório no GitLab da FSLab[11]. Nesse repositório, além da ramificação (branch) padrão master, foram criadas diversas outras branchs, uma para cada funcionalidade descrita no quadro *kanban*. Ao término do desenvolvimento de cada funcionalidade, o código era enviado para o repositório remoto, feito um code review e mesclado na *branch* 'development', sendo essa utilizada para armazenar o código durante o estado de produção do sistema.

A utilização de um repositório remoto possibilitou rastrear o andamento de todo o desenvolvimento do código, além de facilitar a colaboração entre os membros da equipe, já que todos tinham acesso ao mesmo código atualizado.

3.2.2 Modelagem

Ao longo da documentação do projeto, alguns diagramas UML (Unified Modeling Language - Linguagem Unificada de Modelagem) foram criados. Dentre eles o diagrama de classes, demonstrado na figura 4, é responsável por exemplificar a modelagem e estrutura das classes, demostrando como os dados se relacionam dentro da aplicação.

[9] Disponível em https://pypi.org/project/Flask/
[10] Disponível em https://www.mysql.com/
[11] Disponível em https://gitlab.fslab.dev/fslab/api-plataforma-de-cursos-ifro

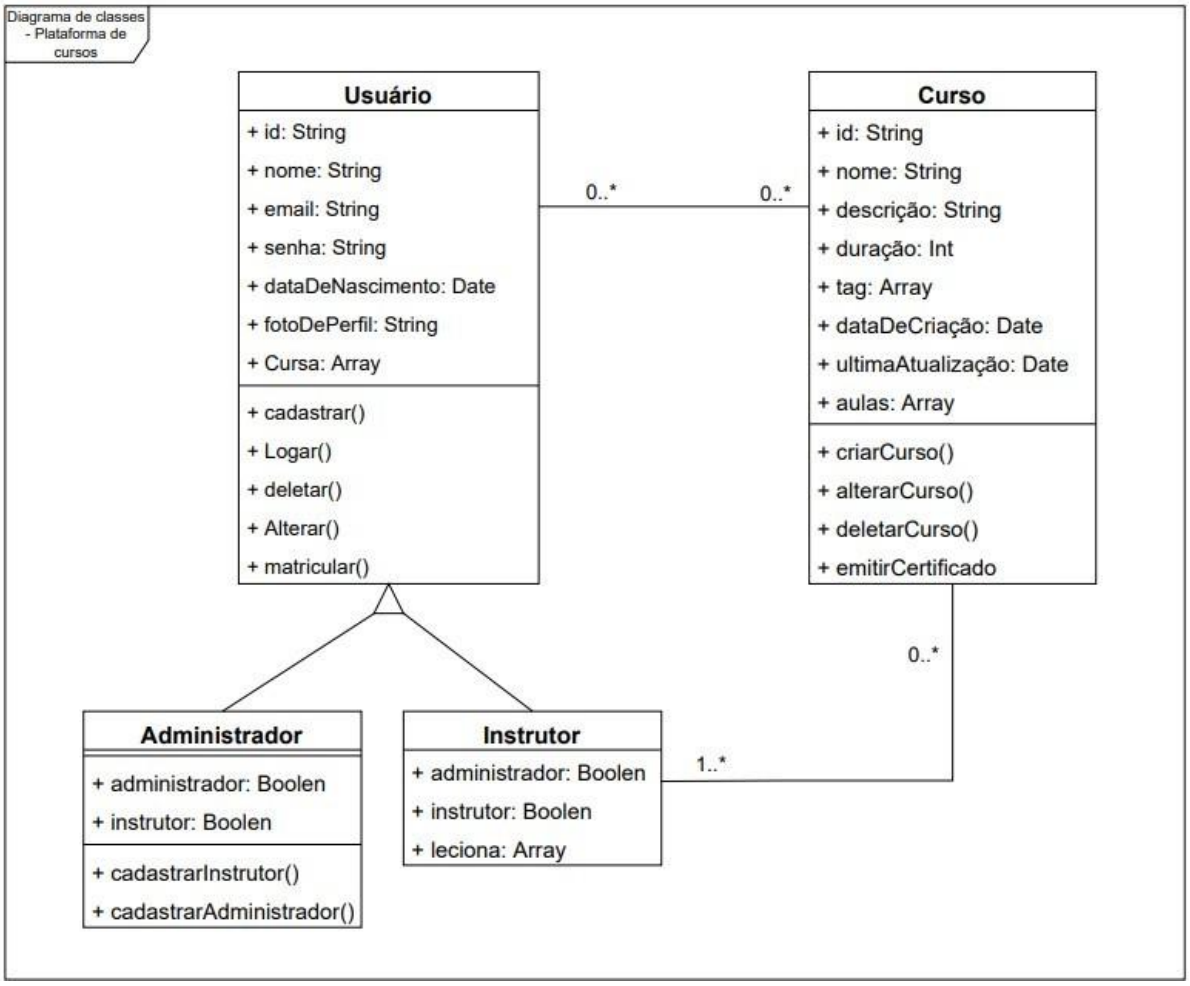


Figura 4 – Diagrama de classes

A figura 5 demonstra o diagrama de atividade, nele é mostrado o fluxo de execução do sistema para a matrícula em um curso.

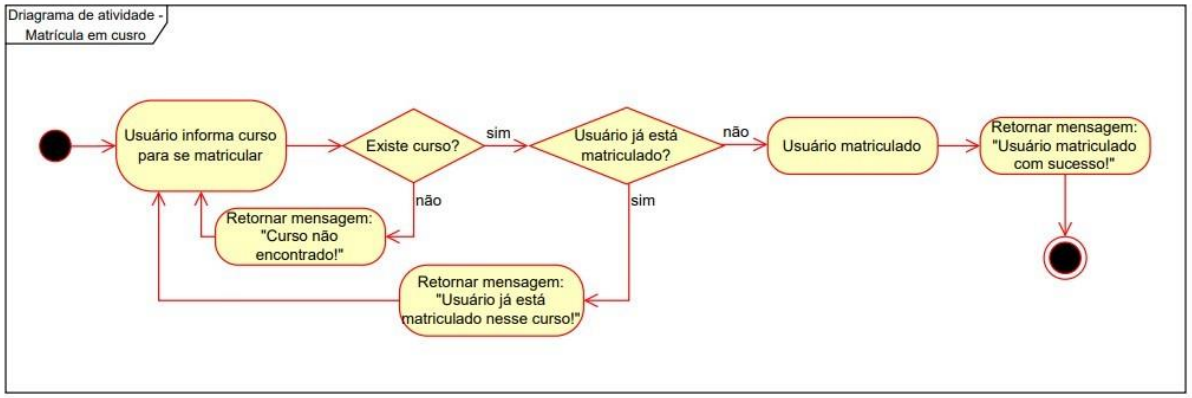


Figura 5 – Diagrama de atividade - Matrícula em curso

3.3 Requisitos

Nos métodos ágeis, o product-owner (PO) é a pessoa ou representante das partes interessadas no produto. Seu papel é responsável por apresentar requisitos, validar entregas de valor e garantir a integridade conceitual das funcionalidades. Dessa forma, este projeto teve como product-owner o Prof. Marco Antônio.

Em conjunto com PO e com base no protótipo da aplicação[12], diversos requisitos foram definidos. A seguir são listados os requisitos funcionais:

- Como aluno, eu quero me matricular em um curso, para que possa acessar seu conteúdo.
- Como aluno, eu quero poder buscar cursos pelo nome, para que eu possa encontrar cursos do meu interesse.
- Como administrador, eu quero poder cadastrar novos instrutores, para que mais cursos possam ser criados.
- Como administrador, eu quero cadastrar novos administradores, para que ajudem na moderação da plataforma.
- Como instrutor, eu quero cadastrar mais de um instrutor para um cursos, para que colegas possam participar dos cursos.
- Como instrutor, eu quero colocar mais de uma tag para um curso, para que meus cursos tenham mais relevância.
- Como aluno, eu quero ter acesso ao meu certificado logo após a conclusão do curso, para que eu possa comprovar minhas atividades.
- Como usuário, eu quero verificar a autenticidade dos certificados, para que eu tenha a segurança de serem válidos.

Foram levantados também alguns requisitos não funcionais:

- O sistema deve ser desenvolvido na linguagem de programação JavaScript.
- O sistema deve utilizar o ambiente de execução Node.js.
- O sistema deve utilizar banco de dados MongoDB.

Vale ressaltar que os requisitos aqui listados referem-se a apenas uma parte da aplicação. Alguns outros foram levantados, entretanto, não eram abrangidos pelo escopo deste projeto.

[12] Disponível em https://www.figma.com/file/61Rl6m2Fw8f7bzcYVjJgWx/Plataforma-de-cursos

3.4 Arquitetura do software

A arquitetura da plataforma foi construída pensando na API, ela servirá como um controlador entre o *front-end* e o banco de dados. A partir da API o *front-end web* e o aplicativo mobile poderão consumir e registar dados no banco de dados. A figura 6 exemplifica a arquitetura da plataforma a ser desenvolvida.

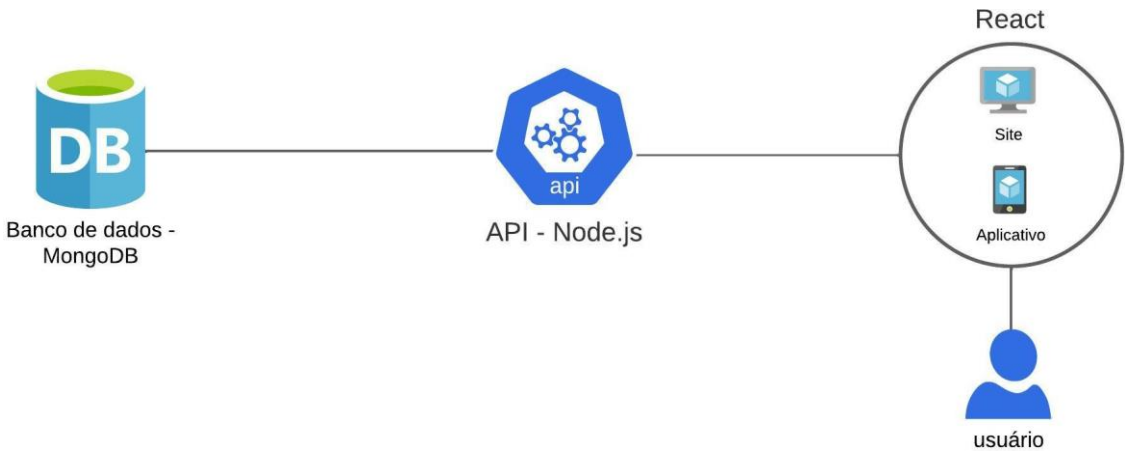
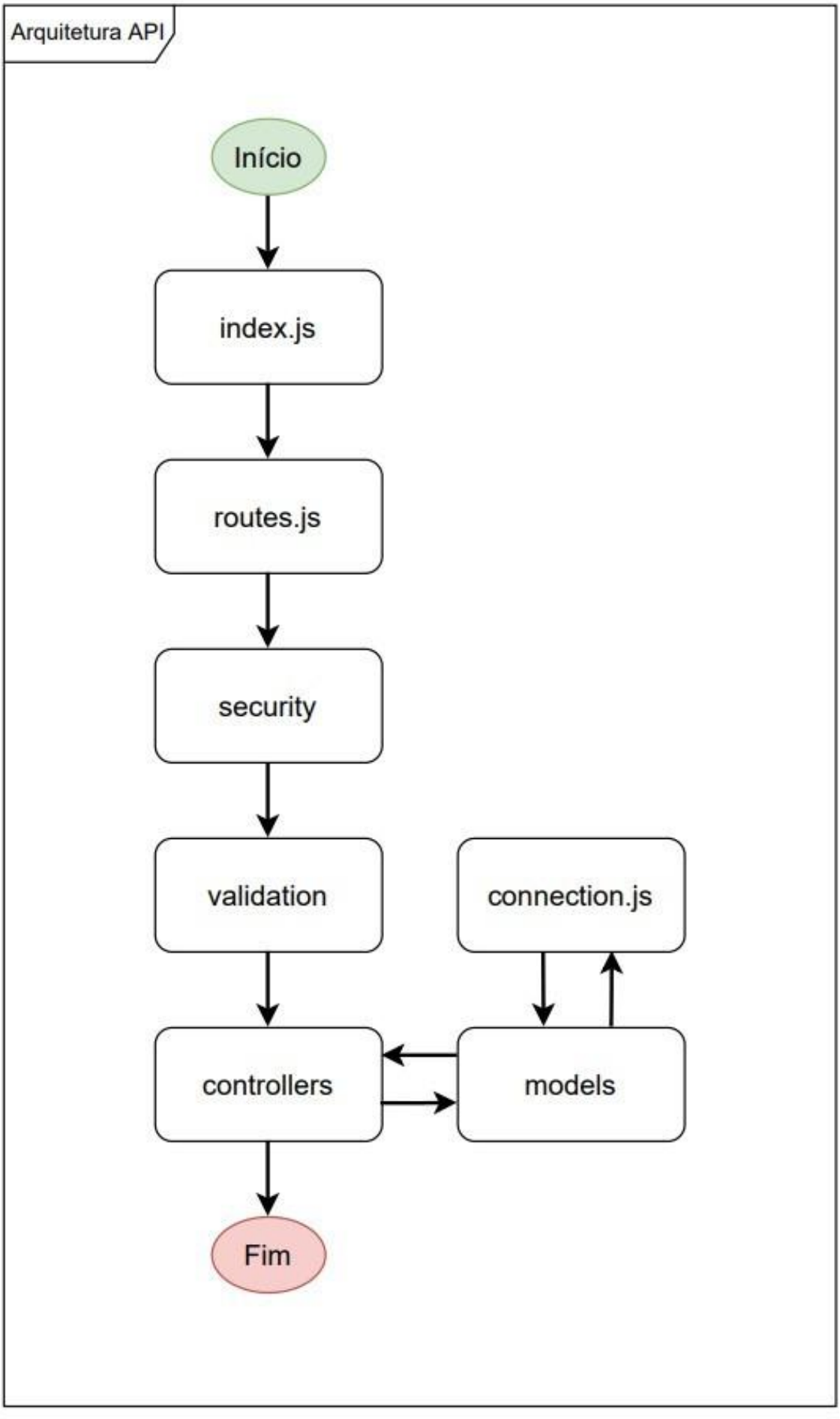


Figura 6 – Arquitetura da Plataforma

A API foi construída de forma monolítica, ou seja, todos os serviços da api estão no mesmo bloco. Inicialmente, a requisição é escutada pelo arquivo *index.js*, passando depois para arquivo *routes.js*, que chama as funcionalidades *security* e *validation*, sendo essas responsável pela validação dos dados de entrada. Em seguida, a requisição é enviada para os *controllers*, responsável por fazer o processamento da requisição e envia-lá para os *models*, esse então, conecta-se com o banco de dados e retorna o resultado das ações realizadas para os *controllers*. Por fim, os *controllers* fazem o processamento e enviam as respostas para o usuário. A figura 7 ilustra a arquitetura da API.


Arquitetura API
Início
index.js
routes.js
security
validation
connection.js
controllers
models
Fim

3.5 Licença de uso

A licença utilizada no desenvolvimento desse *software* foi a licença "MIT"[13]. Uma licença para *software* livre desenvolvida pelo Instituto de Tecnologia de Massachusetts. A escolha dessa licença se deu por conta da clareza dos seus termos, indicando o que pode ou não ser feito com o código-fonte do projeto. Uma de suas vantagens é permitir que qualquer pessoa possa usar, modificar, copiar, mesclar, publicar, distribuir e vender cópias do software sem restrições dos direitos, apenas deve-se manter o aviso de copyright e uma cópia da licença em todas as cópias do software.

[13] Disponível em https://opensource.org/licenses/MIT

4 Resultados e discussões

Neste capítulo são apresentados os resultados e discussões acerca da aplicação proposta.

4.1 Gerenciamento de tarefas - Métricas

4.1.1 Throughput

O throughput é uma relação entre as atividades desenvolvidas e uma unidade de tempo - no caso deste trabalho, um mês. A figura 8 expõe o throughput durante o processo de desenvolvimento

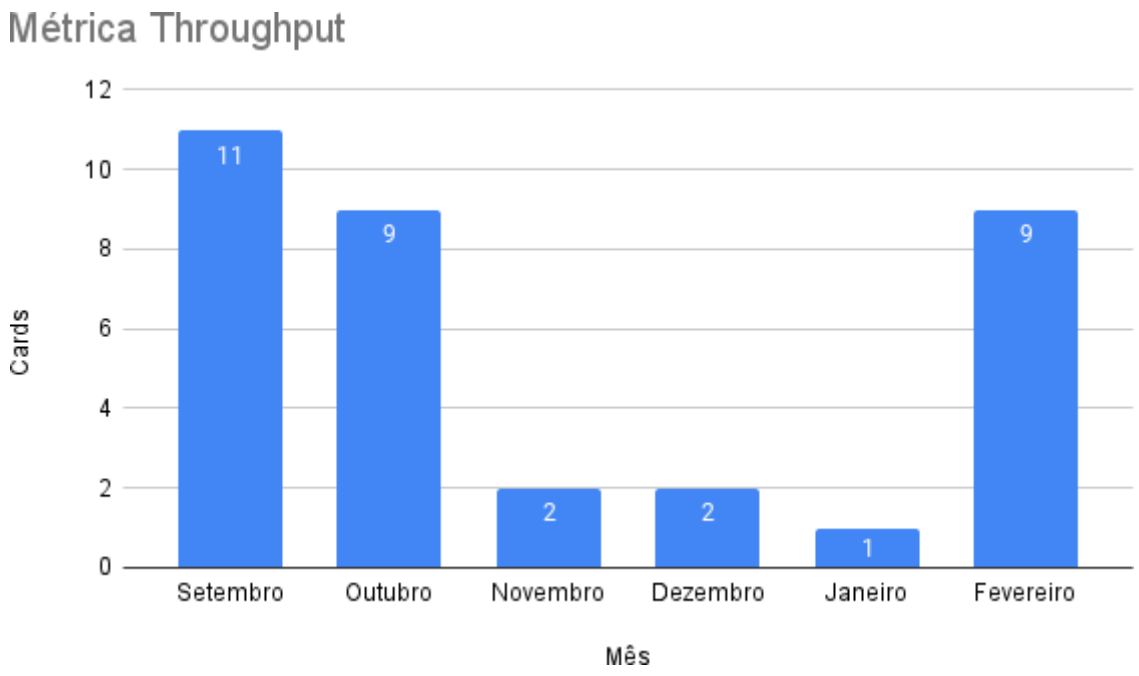


Figura 8 – *Throughput*

Analisando a figura 8 percebe-se que durante o mês de setembro houve o maior número de atividades concluídas. Isso se dá pois, nesse mês, foram realizadas as tarefas iniciais do projeto e também contava com a ajuda da colega Heloísa no desenvolvimento. No mês de outubro o número de atividades foi menor devido à complexidade das mesmas. No mês de novembro, devido à quantidade de atividades acadêmicas e passando a desenvolver sozinho, o andamento das atividades relacionadas a esse projeto foi prejudicado. Em dezembro, desenvolvi algumas atividades no início do mês, após isso afastei-me das atividades acadêmicas, fato que se

estendeu até o final do mês de janeiro. Em fevereiro, voltei com as atividades que faltavam ser desenvolvidas.

Em decorrência desse afastamento, durante o recesso, muitas atividades ficaram por muito tempo abertas, isso acarretou em métricas que não correspondiam com a realidade do projeto. Sendo assim, decidi analisar apenas a métrica de *throughput*.

4.2 Relatório de testes

Durante o desenvolvimento da aplicação utilizamos duas abordagens de testes: testes automatizados e testes manuais. Para os testes manuais, foi utilizado o *software* Insonia [1]. Para os testes automatizados, os *frameworks* Jest[2] e Supertest[3].

4.2.1 Testes manuais

Para testar as funcionalidades, durante e após o desenvolvimento, primeiramente realizava-se o teste manual. Esse método de teste permitia que as alterações fossem testadas com diversas entradas de dados diferentes. Os teste se deram por meio do *software* Insonia, um cliente usado para consumir APIs. A escolha desse software se deu mediante a sua facilidade, desempenho e a praticidade na organização das rotas.

4.2.2 Testes automatizados

Para os testes automatizados, foi desenvolvido arquivos responsáveis pela declaração dos testes. Os testes foram desenvolvidos de acordo com a sintaxe padrão do *framework* Jest.

A sintaxe consiste em:

- Declaração da descrição com o método “describe”, referindo-se a descrição dos testes a serem realizados, de uma forma geral. Esse método também invoca uma função de *callback*, a qual contém todos os testes que serão executados;
- Declaração de testes com o método “it”, em que, também, discrimina a descrição do comportamento esperado do teste a ser realizado, porém, dessa vez, de forma mais especificamente. Assim como o “describe”, o método também invoca uma função de *callback*, a qual contém a lógica do teste.

O *framework* Supertest é utilizado na criação de uma constante, a qual é responsável por realizar uma requisição a uma rota específica e armazenar a resposta da mesma. Após a resposta, o valor é utilizado para a lógica do teste.

1 Disponível em https://insomnia.rest/
2 Disponível em https://jestjs.io/pt-BR/
3 Disponível em https://www.npmjs.com/package/supertest

A figura 9 demonstra a sintaxe e lógica dos testes para usuário. Nele é testado a rota *POST /cadastro* e *GET /usuario.*

```
require('dotenv').config()
const app = require('../src/index')

let request = require('supertest')
request = request(app)

describe('Teste POST /cadastro - Criação de usuários', () => {
  it('Dado um corpo de requisição válido, a rota deve retornar um status 201', async ()
    const response = await request.post('/api/v1/cadastro').send({// Lembre-se de mudar
      nome: 'Márcia Sousa Velasco',
      dataDeNascimento: '1994-11-11',
      email: 'marcia.velasco@geradornv.com.br',
      senha: '837UKFf#q3rf',
      fotoDePerfil: '8da8sdha9d3'
    }).expect(201)
  })

  it('Dado uma senha com menos de 8 caracteres no corpo de requisição, a rota deve retor
    const response = await request.post('/api/v1/cadastro').send({
        nome: 'Márcia Sousa Velasco',
        dataDeNascimento: '18/11/1994',
        email: 'marcia.velasco@geradornv.com.br',
        senha: '837',
        fotoDePerfil: '8da8sdha9d3'
    }).expect(400)
  })
})


describe('Teste GET /usuario - Listagem dos dados do usário logado', () => {
  // Pode ser necessário alterar os tokens
  it('Dado um token de autenticação válido, a rota deve retornar um status 200', async (
    const response = await request.get('/api/v1/usuario').set('x-api-token', 'eyJhbGciOi
      .expect(200)
  })

  it('Dado um token de autenticação inválido, a rota deve retornar um status 500', async
    const response = await request.get('/api/v1/usuario').set('x-api-token', 'EyJhbGciOi
      .expect(500)
  })

  it('Caso não seja informado o token de autenticação, a rota deve retornar um status 40
    const response = await request.get('/api/v1/usuario').expect(403)
  })
})
```

Figura 9 – Teste automatizado - *Cadastro e listagem*

A figura 10 apresenta a sintaxe e lógica dos testes para a rota *POST login.*

```
require('dotenv').config()
const app = require('../src/index')

let request = require('supertest')
request = request(app)

describe('Testando a funcionalidade de autenticação', () => {
  it('Informados os dados de login de um usuário válido, a rota deve retornar um status
    const response = await request.post('/api/v1/login').send({
      email: 'marcia.40@geradornv.com.br',
      senha: '837UKFf#q3rf'
    }).expect(200)
  })/

  it('Informados os dados inválidos de login, a rota deve retornar um status 400', async
    const response = await request.post('/api/v1/login').send({
      email: 'emanuela.furtunato@geradornv.com.br',
      senha: 'Oef88@z5$Vk3'
    }).expect(400)
  })
})
```

Figura 10 – Teste automatizado - *login*

Por fim, a figura 11 expõe os testes automatizados desenvolvidos para a rota *GET /curso/:idcurso*.

```
require('dotenv').config()
const app = require('../src/index')

let request = require('supertest')
request = request(app)

describe('Testando a listagem de cursos', () => {
  it('Dado um id do curso valido como parâmetros, a rota deve retornar um status 200', async () => {
    const response = await request.get('/api/v1/curso/620aac1865d9a91efff30d5f')
      .expect(200)
  })// Lembre-se de passar um id referente a um registro existente no banco

  it('Dado um id do curso invalido como parâmetros, a rota deve retornar um status 404', async () => {
    const response = await request.get('/api/v1/curso/620aac1865d9a91efff3043k')
      .expect(404)
  })
})
```

Figura 11 – Teste automatizado - *curso*

Na tabela 1 estão expostos os testes desenvolvidos, bem como os *status* HTTP esperados em caso de sucesso ou erro.

Rota	Status - Sucesso	Status - Erro
GET /curso - Listagem de curso	200	404
POST /login - Funcionalidade de login	200	400
POST /cadastro - Função de cadastro de usuários	201	400
GET /usuario - Listagem dos dados do usuário logado	200	500, 403

Tabela 1 – Testes e retornos esperados

Os testes automatizados cobrem cerca de 20% das funcionalidades da API, tendo 100% de êxito. Os testes manuais cobriram 100% das funcionalidades e, assim como nos testes automatizados, houve 100% de sucesso nos resultados.

4.3 Documentação

Pensando no desenvolvimento de trabalhos futuros, a partir desse projeto, foi elaborada a documentação da API para auxiliar o trabalho de prováveis desenvolvedores. Nessa fase, utilizou-se do *software open-source* Swagger[4]. Contando com o auxílio da biblioteca Swagger UI Express[5], é possível fornecer uma documentação gerada por Swagger UI, baseada em um arquivo que pode ser YAML ou JSON. Assim, foi gerada uma documentação da API hospedada no servidor da aplicação, sendo acessível pela rota [6].

Conforme mostrado na figura 12, as rotas documentadas são separadas por *tags* definidas no código da documentação. Nesse caso, as rotas estão divididas em:

- Usuário: categoria que contém todas as rotas relacionadas com usuários.
- Curso: categoria que contém todas as rotas relacionadas com cursos;
- Aula: categoria que contém todas as rotas relacionadas com aulas;
- Certificado: categoria que contém todas as rotas relacionadas com certificados;

[4] Disponível em https://swagger.io
[5] Disponível em https://www.npmjs.com/package/swagger-ui-express
[6] Disponível em https://cursos.fslab.dev/docs

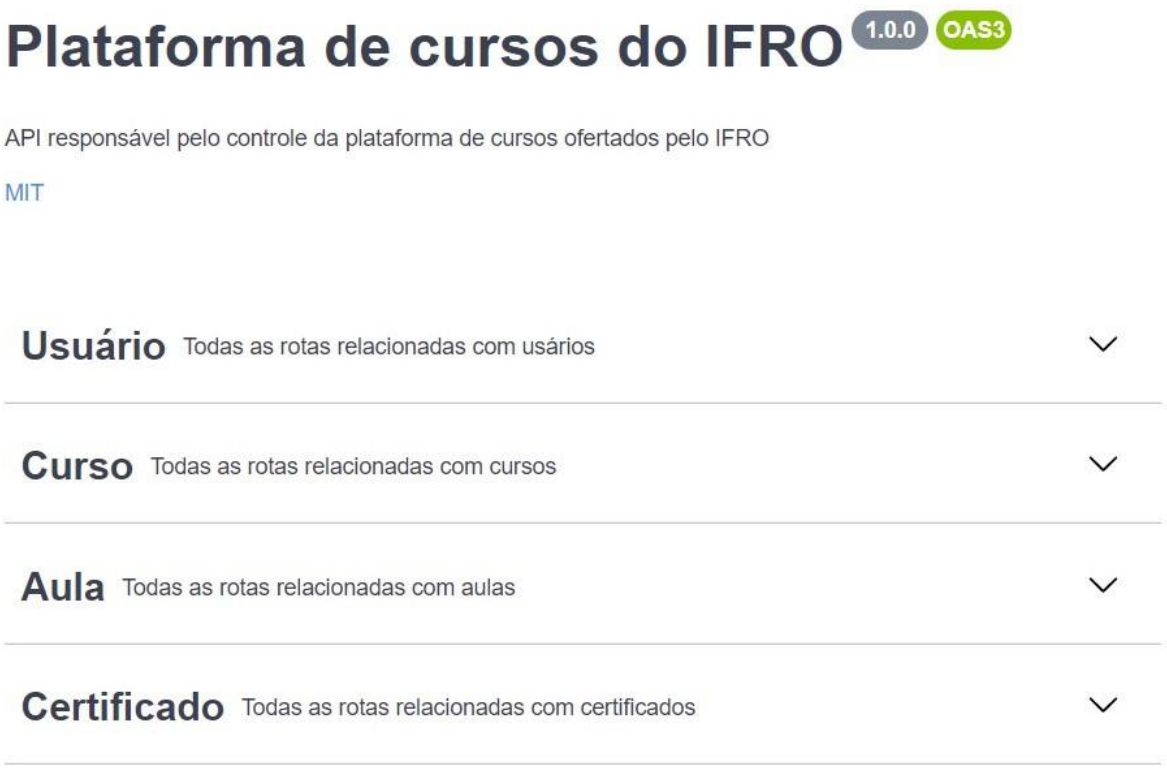


Figura 12 – Tags Swagger

Cada rota contém, além da separação por tags, um *summary* (resumo), o qual também é definido no código da documentação. Esse resumo serve para descrever a funcionalidade de determinada rota.

As rotas também possuem métodos HTTP de requisição, para essa aplicação foram utilizados apenas quatro: Post; Get; Put; Delete.

- Post: Essas rotas são responsáveis por enviarem informações do *front-end* ao *back-end*, foram utilizadas, principalmente, para a criação de registros. Nessas rotas as informações são passadas no *body* (corpo) da requisição. A figura 13 demonstra um exemplo dessas rotas para criação de um novo curso;

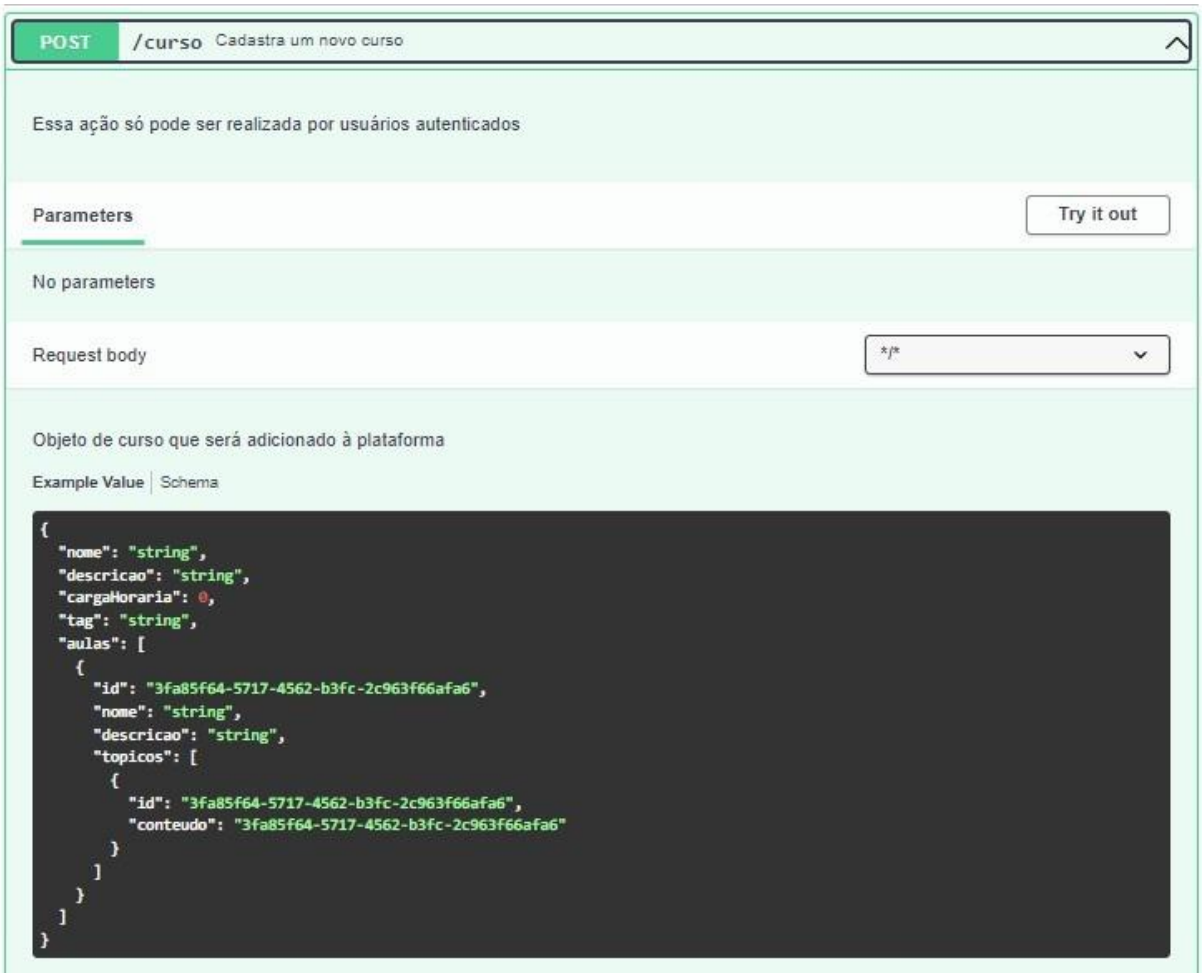


Figura 13 – Rota POST

- Get: rotas que estão encarregadas de retornam informações ao *front-end*). A figura 14 demonstra um exemplo dessas rotas para listagem de curso;

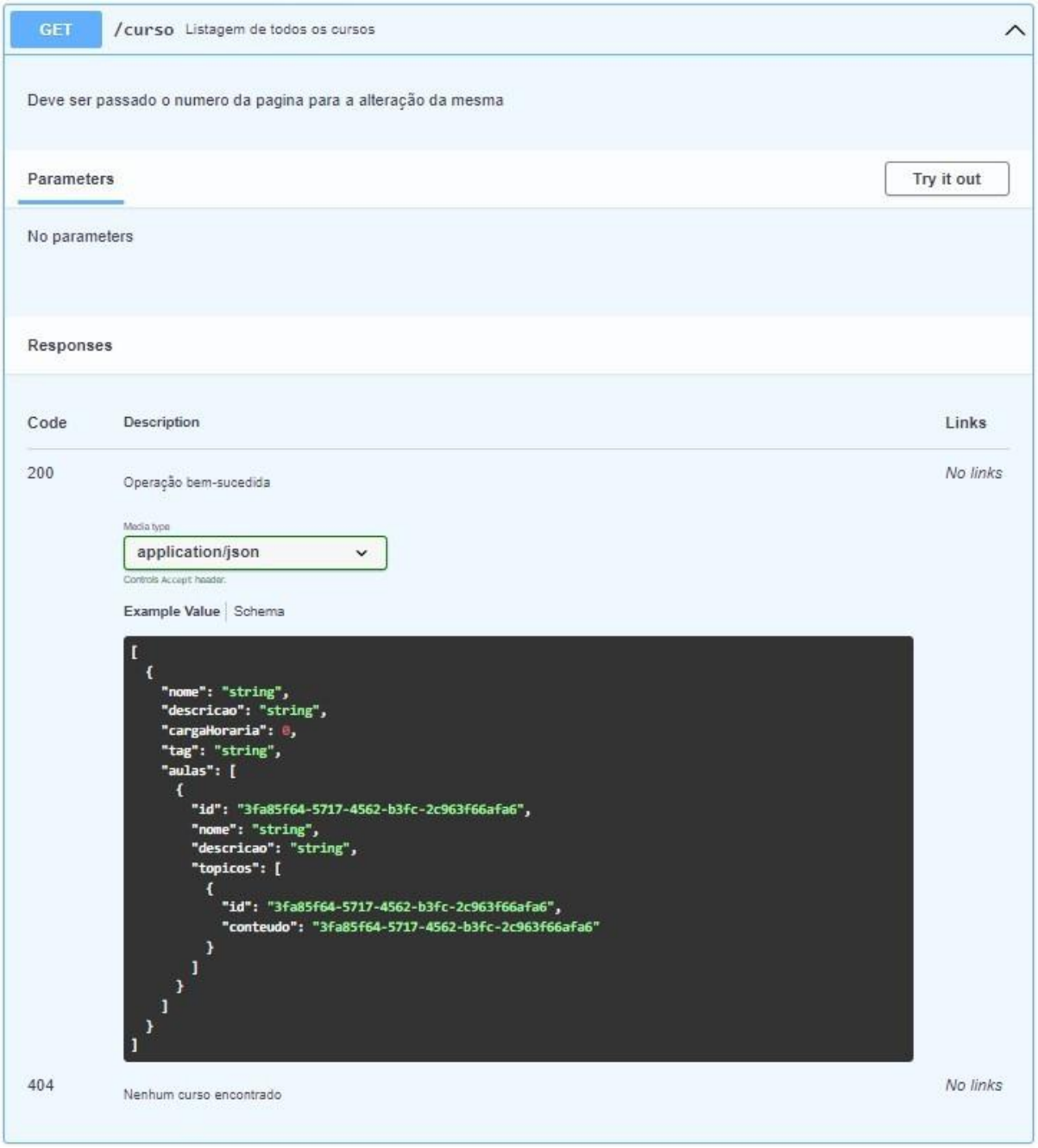


Figura 14 – Rota GET

- Put: rotas responsáveis por atualizar registros já existentes. Nessas rotas informa-se no corpo da requisição as informações a serem atualizadas. Na maioria dos casos, é informado na URL o identificador do registro a ser alterado. A figura 15 demonstra um exemplo dessas rotas para a alteração de cursos.

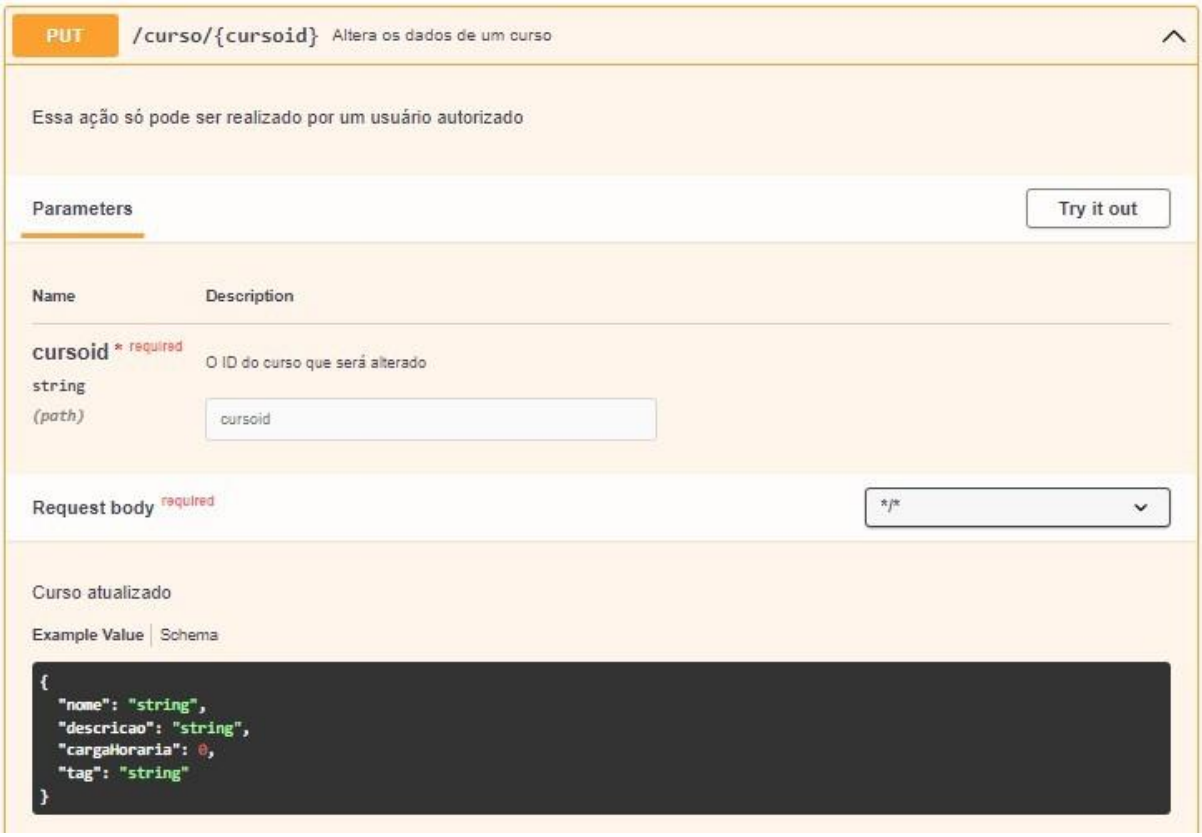


Figura 15 – Rota PUT

- Delete: rotas incumbidas de excluir algum registro, é informado na URL o identificador do registro a ser excluído. A figura 16 demonstra um exemplo dessas rotas para exclusão de curso;

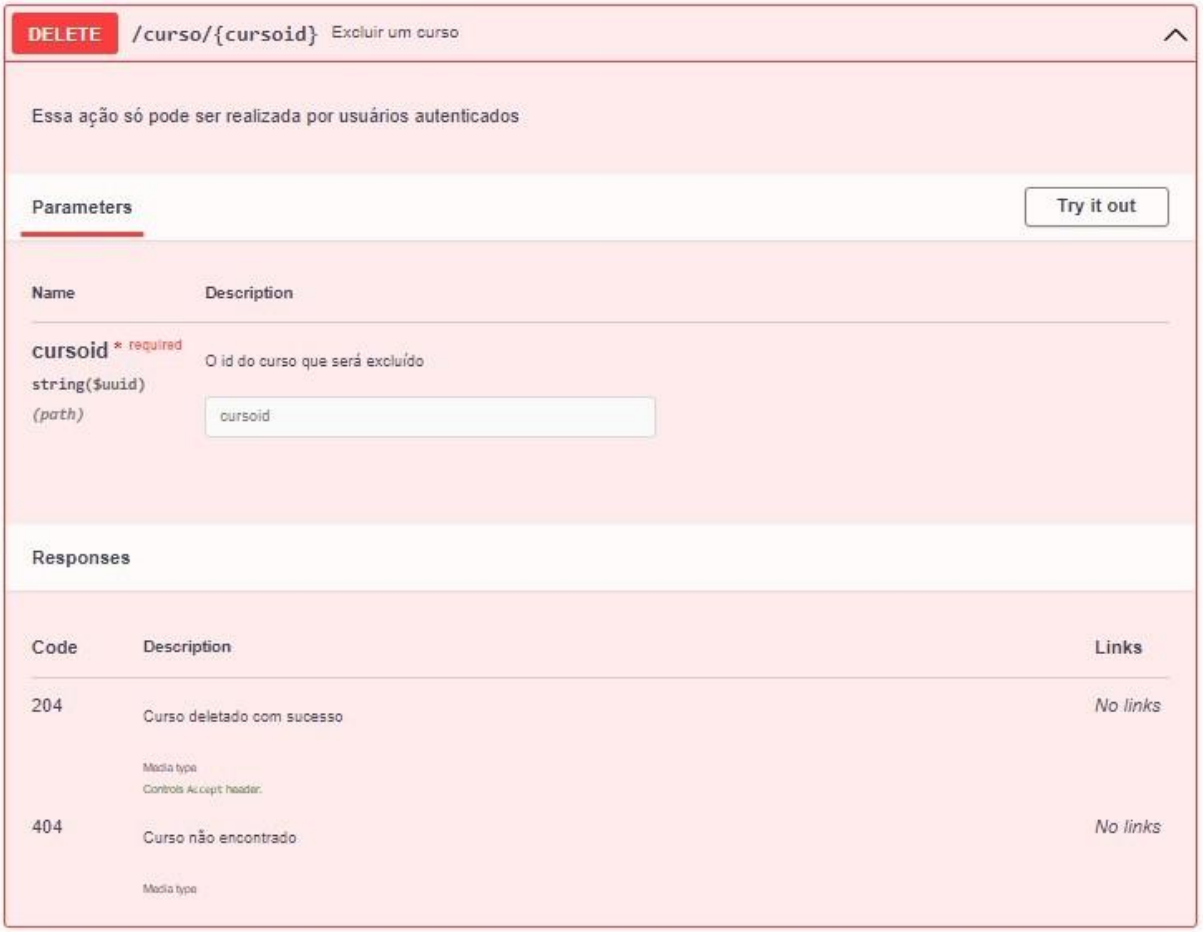


Figura 16 – Rota DELETE

4.4 Populando o banco

Para acelerar e facilitar os testes das rotas, realizou-se o desenvolvimento de um arquivo para semear o banco de dados. Com o uso da biblioteca mongo-seed[7], foi possível inserir registros com informações falsas, utilizadas para a verificação do retorno das rotas. Executando corretamente o comando, a aplicação inicia o processo, inserindo os registros para todas as coleções. A figura 17 demonstra o arquivo de geração de registros.

[7] Disponível em: https://www.npmjs.com/package/mongoose-seed

Figura 17 – Seeder

4.5 Implantação

Para o *deploy* da aplicação utilizou-se a técnica de conteinerização. Todos os serviços da aplicação estão sendo executados em uma VM, a qual está rodando no servidor da FSLab[8]. Essa técnica foi escolhida tendo em vista a facilidade de escalonamento da aplicação, além da redução de recursos consumidos, em razão da utilização de contêineres.

Para a execução da aplicação e gerenciamento de seus processos, utilizou-se a ferramenta PM2[9]. A escolha dessa ferramenta deu-se em decorrência de sua gama de funcionalidades. Além de geração de relatórios e métricas, o PM2 fornece reinício automático da aplicação. Outro poderoso recurso dessa ferramenta é o *Cluster Mode*, permitindo o dimensionamento dinâmico da aplicação para cada CPU disponível.

4.6 Demonstração do *software*

Fazendo uso de uma plataforma que realize requisições HTTP, é possível realizar testes das funcionalidades API, durante o desenvolvimento utilizou-se *software* Insonia [10], como citado anteriormente.

[8] Disponível em https://cursos.fslab.dev
[9] Disponível em https://pm2.keymetrics.io
[10] Disponível em https://insomnia.rest/

A realização dos testes das rotas devem ser feitos pelo link <https://cursos.fslab.dev/>. Será necessário realizar o login com o e-mail *"admin@ifro.com.edu"* e a senha *"837UKFfq3rf"*. Deve ser realizado o *login* através da rota "*/login*". Vale ressaltar que a aplicação também realiza o *hash* da senha do usuário. Caso o *login* seja bem-sucedido, a resposta da rota possuirá um *token* de autenticação. Nas demais rotas, o *token* deve ser informado no cabeçalho da requisição no parâmetro "*x-api-token*".

Por meio da rota POST - "*/curso*", é possível criar um curso. De acordo com a demonstração na figura 18, para a realização da criação de um curso é necessário informar cinco campos: nome; duração; descrição; tag e aulas, sendo, o último opcional neste momento. Após o envio da requisição, os dados são processados e salvos. Caso haja erros nesses procedimentos, estes então, serão retornados como resposta da requisição. A figura 19 demonstra a resposta da rota em caso de sucesso.

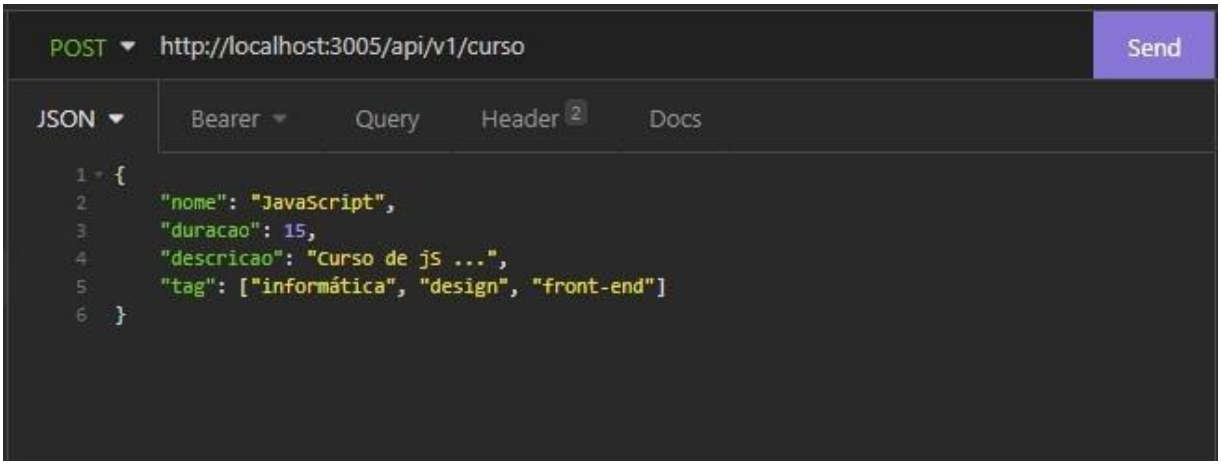


Figura 18 – Requisição - POST "/curso"

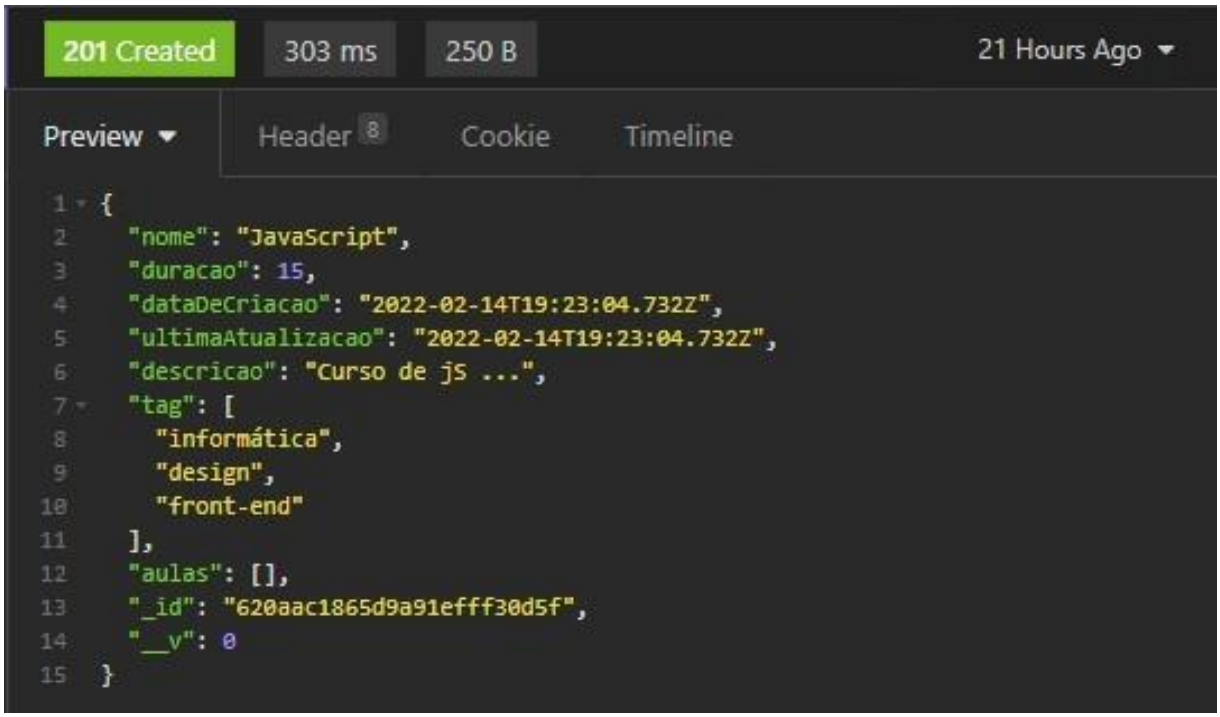


Figura 19 – Resposta - POST "/curso" - sucesso

Através da rota GET - "*/curso*", é retornado todos os cursos criados na plataforma, conforme demonstrado na figura 20, com um sistema de paginação, listando no máximo dez cursos por página. Essa mesma requisição pode ser enviada com parâmetros de *nome*, na qual será retornado os cursos com devidos nomes.

```
{
  "results": [
    {
      "_id": "62070d437e0ab49623fd2764",
      "nome": "JavaScript",
      "duracao": 15,
      "dataDeCriacao": "2022-02-12T01:28:35.037Z",
      "ultimaAtualizacao": "2022-02-12T01:29:38.368Z",
      "descricao": "Curso de jS ...",
      "tag": [
        "informática",
        "design",
        "front-end"
      ],
      "aulas": [
        {
          "nome": "Variaveis",
          "descricao": "",
          "topicos": [
            {
              "conteudo": "uuid",
              "id": "0341d8dd-a49e-4a4f-83c2-39e518a3dde3"
            },
            {
              "conteudo": "uuid",
              "id": "d6434bac-50cf-49c3-bd04-9781d5b346f3"
            }
          ],
          "id": "5e2607d3-abff-4f97-931a-2f03d5b70466"
        },
        { ↔ 4 ↦ }
      ],
      "__v": 0
    },
```


Figura 20 – Resposta - GET "/curso" - Sucesso

O procedimento para adicionar novas aulas a um curso é semelhante a criação do curso. No entanto, deverá ser usado a rota POST - "*/curso/:cursoid/aula*", podendo ser enviado apenas um objeto ou um array de objetos. A figura 21 exemplifica a inserção de aulas em um curso.

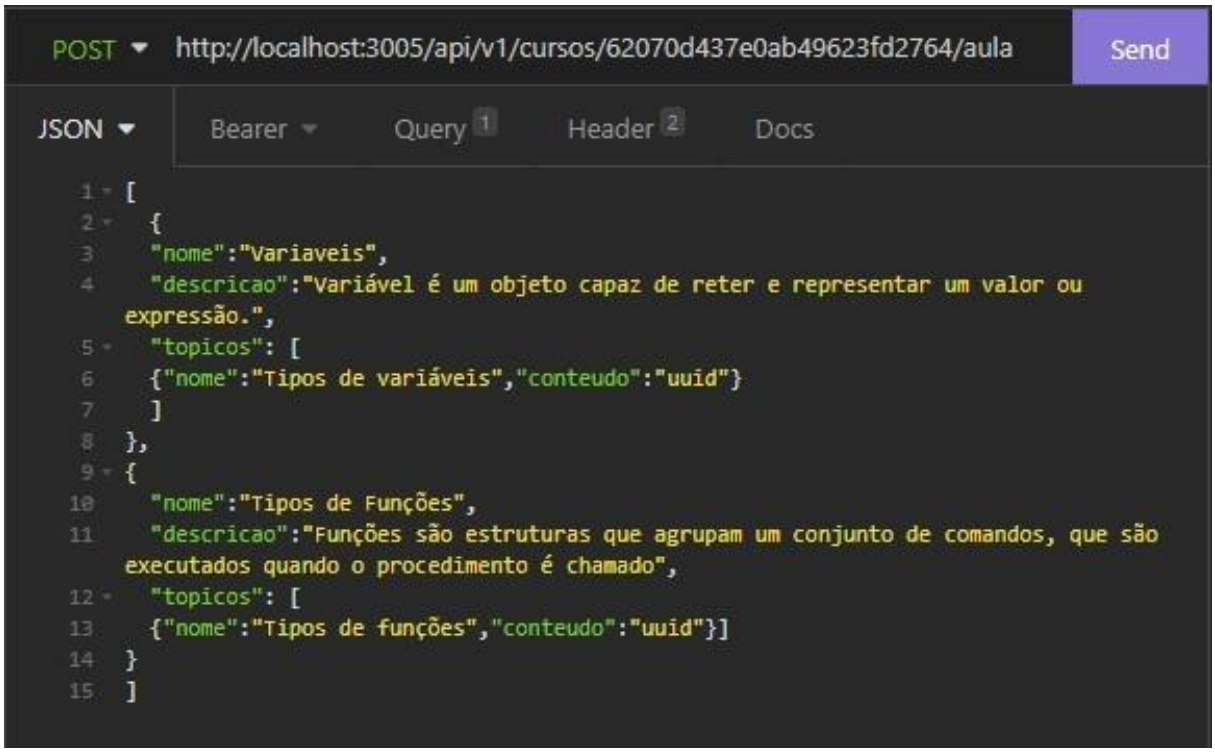


Figura 21 – *aula* - POST "/curso/:cursoid/aula"

Sendo retornado, em caso de sucesso, o objeto criado demostrado na figura 22.

Figura 22 – *aula* - POST "/curso/:cursoid/aula" - sucesso

A partir da rota PUT - "*/curso/:cursoid*", será possível a alteração dos dados de um curso. Os campos que podem ser alterados são os mesmos utilizados na hora de criação do curso. A figura 23 apresenta o procedimento. A resposta esperada, em caso de sucesso, é demonstrada na figura 24.

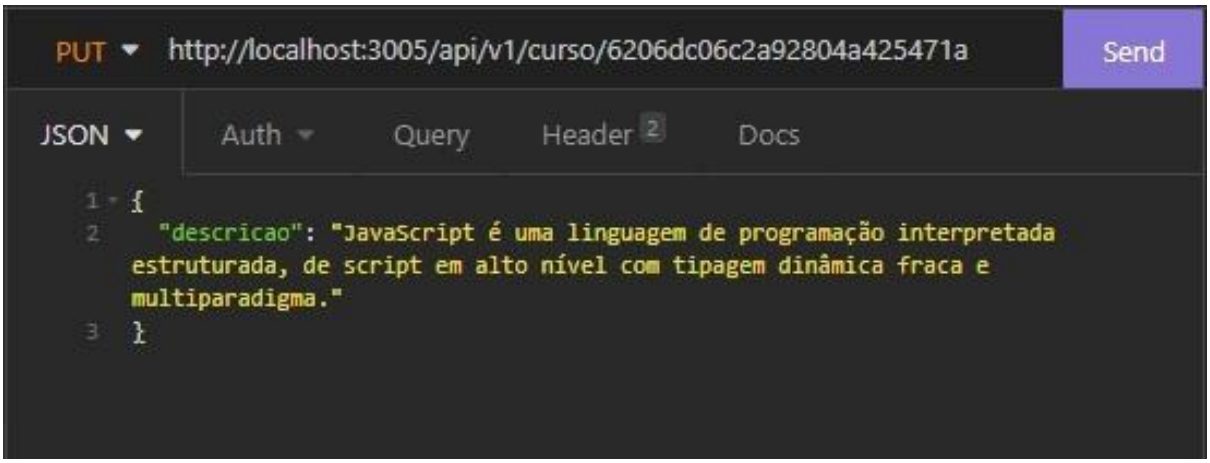


Figura 23 – *aula* - PUT "/curso/:cursoid"

Figura 24 – *aula* - PUT "/curso/:cursoid" - sucesso

Por fim, a rota DELETE - "*/curso/:cursoid*" é responsável pela exclusão de um curso. O id do curso, que será excluído, deve ser passado no parâmetro da requisição. A figura 25 exemplifica o processo. Em caso do curso informado não existir, será retornado o erro demonstrado na figura 26.

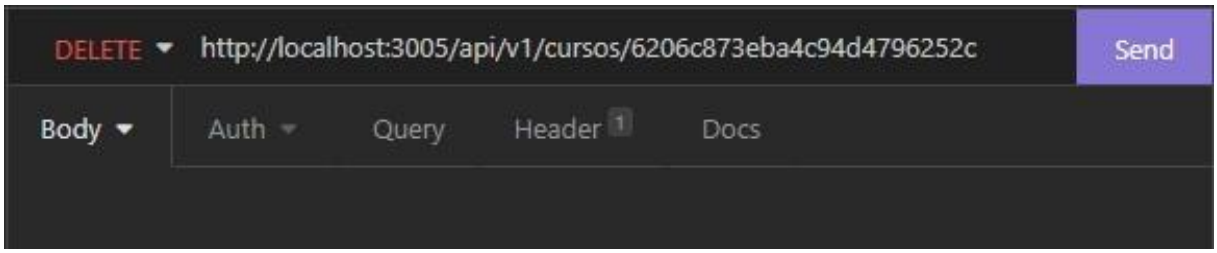


Figura 25 – *curso* - DELETE "/curso/:cursoid"

Figura 26 – *aula* - DELETE "/curso/:cursoid" - erro

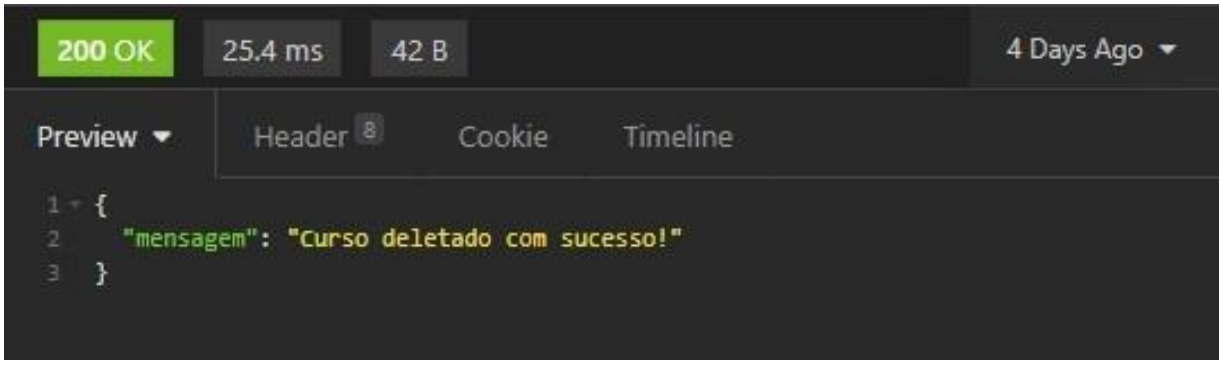


Figura 27 – *aula* - DELETE "/curso/:cursoid" - sucesso

5 Considerações finais

O processo de desenvolvimento desse projeto foi um grande desafio. Logo, um grande aprendizado para mim. Com ele foi necessário aprender novas tecnologias, desenvolver as metodologias utilizadas no mercado de trabalho e colocar em prática todo o aprendizado adquirido durante três anos de curso. Sendo assim, o produto apresentado é apenas uma parte da API da Plataforma de Cursos do IFRO, sendo que todo escopo definido para esse projeto foi atendido.

O projeto visa atender a demanda do IFRO em auxiliar a formação de seus alunos, tanto de nível médio como superior. Ter uma plataforma com cursos desenvolvidos com os próprios docentes da instituição possibilitará um grande controle de qualidade, cursos direcionados para as áreas requeridas e a oferta de certificados autenticados pelo IFRO. Além de tudo, o projeto sendo um *software open-source*, dá a oportunidade que outras instituições ofertarem suas próprias plataformas de cursos.

5.1 Trabalhos futuros

Após o desenvolvimento da solução proposta, foram identificados os seguintes trabalhos futuros que podem ser realizados:

- Adicionar gerenciamento de mídias.
- Permitir o cadastramento de atividades práticas para cursos.
- Implementar filtros na busca de cursos para o usuário final, como filtros por tags e instrutor.

Referências

AGONÁCS, N.; MATOS, J. F. Os cursos on-line abertos e massivos (mooc) como ambientes heutagógicos. RBEP, 2020. Disponível em: <https://www.scielo.br/j/rbeped/a/MKZbj9vMn5SSNzH6wDghZvC/?lang=pt#>. Citado na página 25.

ALURA. *Alura*. 2022. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://www.alura.com.br/sobre>. Citado na página 28.

ANDRADE, S. de et al. Os desafios do ensino à distância e do uso da tecnologia de informação e comunicação. UFRN, 2017. Disponível em: <https://periodicos.ufrn.br/casoseconsultoria/article/view/21836/13406>. Citado na página 23.

AZURE. *O que é um contêiner?* 2022. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://azure.microsoft.com/pt-br/overview/what-is-a-container/#overview>. Citado na página 26.

BECK, K. et al. *Manifesto para Desenvolvimento Ágil de Software*. 2001. [Online; acessado em 14-Fevereiro-2022]. Disponível em: <https://agilemanifesto.org/iso/ptbr/manifesto.html>. Citado na página 27.

BOEG, J. *PRIMING KANBAN: A 10 step guide to optimizing flow in your software delivery system*. InfoQ, 2012. v. 2. Disponível em: <https://www.infoq.com/minibooks/priming-kanban-jesper-boeg/>. Citado na página 28.

COSTA, A. R. da. A educaÇÃo a distÂncia no brasil: Concepções, histórico e bases legais. Unirios, 2017. Disponível em: <https://www.unirios.edu.br/revistarios/media/revistas/2017/12/a_educacao_a_distancia_no_brasil_concepcoes_historico_e_bases_legais.pdf>. Citado na página 23.

COURSERA. *Coursera*. 2022. [Online; acessado 14-Dezembro-2020]. Disponível em: <https://pt.coursera.org/>. Citado na página 28.

DIAS, N. F. e. T. M. é. H. Desenvolvimento colaborativo e integrado de sistemas: Caso de sucesso no cpd-ufrgs. Universidade Federal do Rio Grande do Sul, 2014. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://www.lume.ufrgs.br/bitstream/handle/10183/105098/000943844.pdf?sequence=1>. Citado na página 27.

FIELDING, R. N. T. é. R. T. Principled design of the modern web architecture. ACM, p. 1, 2000. Citado na página 25.

FRANÇA, P. C. S. et al. Satisfação de usuários de cursos de ensino à distância no brasil. 2022. Citado 2 vezes nas páginas 23 e 24.

IBM. *IBM*. 2015. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://www.ibm.com/docs/pt-br/rational-soft-arch/9.7.0?topic=diagrams-execution-environments>. Citado na página 25.

IBM. *APIs de REST*. 2021. [Online; acessado em 13-Fevereiro-2022]. Disponível em: <https://www.ibm.com/br-pt/cloud/learn/rest-apis>. Citado na página 25.

IFRO, M. *SOBRE O MOOC*. 2022. [Online; acessado 26-Março-2022]. Disponível em: <https://mooc.ifro.edu.br/pages/about>. Citado na página 29.

KARPOV, V. et al. *Schemas*. 2022. [Online; acessado em 26-Março-2022]. Disponível em: <https://github.com/Automattic/mongoose/blob/master/docs/guide.md>. Citado na página 33.

KUHN, N.; HOFLER, C. E.; SILVA, S. S. da. Análise da satisfação dos estudantes de um curso técnico ead. 2017. Disponível em: <https://revistas.rcaap.pt/uiips/article/view/21985/16088>. Citado na página 24.

MACHADO, L. D.; MACHADO, E. de C. O papel da tutoria em ambientes de ead. ABED, 2004. Disponível em: <http://www.abed.org.br/congresso2004/por/htm/022-tc-a2.htm>. Citado na página 24.

MICROSOFT. *O que é o Docker?* 2021. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/architecture/microservices/container-docker-introduction/docker-defined>. Citado na página 26.

MONGODB. *O que é o MongoDB?* 2021. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://www.mongodb.com/pt-br/what-is-mongodb>. Citado na página 26.

MOODLE. *O sistema de gerenciamento de aprendizagem de código aberto mais personalizável e confiável do mundo*. 2022. [Online; acessado 26-Março-2022]. Disponível em: <https://moodle.com/pt/lms/recursos-do-moodle/>. Citado na página 29.

MOZILLA. *JavaScript*. 2022. [Online; acessado em 13-Fevereiro-2021]]. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/About_JavaScript>. Citado na página 25.

MURÇA, B. et al. Projeto “mooc: Introdução ao git”, soluções para formações profissionais em tempo de pandemia. Revista UI*PSantarm*, 2020.*Disponvelem* : <>. Citado na página 27.

NOBRE, J. C. de A.; NAVES, A. M. A produção da educação superior no brasil: analisando controvérsias acerca da ead. UERJ, 2015. Disponível em: <https://www.e-publicacoes.uerj.br/index.php/revispsi/article/view/20276/14609>. Citado na página 23.

NODE. *introdução ao Node.js*. 2022. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://nodejs.dev/learn>. Citado na página 26.

O’GRADY, A. *GitLab Quick Start Guide: Migrate to GitLab for all your repository management solutions*. [S.l.]: Packt Publishing Ltd, 2018. Citado na página 27.

PAPPANO, L. The year of the mooc. The New York Times, 2012. Disponível em: <https://www.lernspielwiese.com/cms/lib07/MN01909547/Centricity/Domain/272/The%20Year%20of%20the%20MOOC%20NY%20Times.pdf>. Citado na página 24.

SILVA, F. A. d. M. A. é. Jessica Belém da. Método kanban como ferramenta de controle de gestão. Id on Line, p. 1018–1027, 2019. Disponível em: <https://idonline.emnuvens.com.br/id/article/download/1575/2325>. Citado na página 27.

SILVA, M. P. D.; MELO, M. C. D. O. L.; MUYLDER, C. F. D. EducaÇÃo a distÂncia em foco: Um estudo sobre a produÇÃo cientÍfica brasileira. RAM, 2015. Disponível em: <https://www.scielo.br/j/ram/a/NBrjWSWJKnnbgfDjTTxbMth/abstract/?lang=pt>. Citado na página 24.

TOTH, R. M. Abordagem nosql – uma real alternativa. Universidade Federal de São Carlos – Campus Sorocaba, 2011. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://dcomp.ufscar.br/verdi/topicosCloud/nosql_artigo.pdf>. Citado na página 26.

UDEMY. *Udemy*. 2022. [Online; acessado 14-Fevereiro-2022]. Disponível em: <https://about.udemy.com/pt-br/>. Citado na página 28.

VASCONCELOS, C. R. D.; JESUS, A. L. P. de; SANTOS, C. de M. Ambiente virtual de aprendizagem (ava) na educação a distância (ead): um estudo sobre o moodle. Brazilian Journals, 2020. Disponível em: <https://www.brazilianjournals.com/index.php/BRJD/article/view/8165/7044>. Citado na página 24.

6 Licença MIT