UNICEPLAC

Centro Universitário do Planalto Central Apparecido dos Santos - UNICEPLAC
Curso de Sistemas de Informação
Trabalho de Conclusão de Curso

Projeto de desenvolvimento de API para registro e controle de horas complementares

Gama-DF
2020

UNICEPLAC

ANDERSON DA COSTA ANACLETO
ARTHUR MEIRELES DOS SANTOS
EDUARDO JOHNSON CARVALHO MARTINS

Projeto de desenvolvimento de API para registro e controle de horas complementares

Artigo apresentado como requisito para conclusão do curso de Bacharelado em sistemas de informação pelo Centro Universitário do Planalto Central Apparecido dos Santos – Uniceplac.

Orientador: Prof. Esp. Hélder Line Oliveira

Gama-DF
2020

UNICEPLAC

ANDERSON DA COSTA ANACLETO
ARTHUR MEIRELES DOS SANTOS
EDUARDO JOHNSON CARVALHO MARTINS

Projeto de desenvolvimento de API para registro e controle de horas complementares

Artigo apresentado como requisito para conclusão do curso de Bacharelado em sistemas de informação pelo Centro Universitário do Planalto Central Apparecido dos Santos – Uniceplac.

Gama, 3 de dezembro de 2020.

Banca Examinadora

Prof. Esp. Hélder Line Oliveira
Orientador

Prof. André Félix Freitas
Examinador

Prof. Sebastião Ivaldo Carneiro Portela
Examinador

API de registro de horas complementares

ANDERSON DA COSTA ANACLETO
ARTHUR MEIRELES DOS SANTOS
EDUARDO JOHNSON CARVALHO MARTINS

Resumo:

Os sistemas de informação buscam simplificar os processos e trazer mais agilidade, praticidade e eficácia no dia a dia da população. A partir dessa ideia, foi considerado a necessidade de tornar mais fácil e rápido a contabilização das horas complementares dos alunos de ensino superior. Portanto esse projeto visa construir uma API (Application Programming Interface) de código aberto que auxilia no controle de horas relativas às atividades extracurriculares executadas, onde será possível registrar, gerenciar, consultar e acompanhar as horas complementares que foram enviadas pelos estudantes. Um front end responsivo em REACT foi criado para apresentação visual do projeto e contribuição com a comunidade acadêmica.

Palavras-chave: Atividade, Complementar, API.

Abstract:

The information systems seek to simplify the processes and bring more agility, practicality and efficiency in the daily life of the population. Based on this idea, it was considered the need to make it easier and faster to count the complementary hours of higher education students. Therefore, this project aims to build an open source API (Application Programming Interface) that assists in the control of hours related to the extracurricular activities performed, where it will be possible to register, manage, consult and monitor as complementary hours that were analyzed by the students. A responsive REACT front end was created to visually present the project and contribute to the academic community.

Keywords: Activity, Complementary, API.

1. INTRODUÇÃO

As demandas atuais exigem o aprimoramento de todos os processos para ter maior integridade e controle dos dados, melhorando o gerenciamento deles. De acordo com Garcia (2005), é importante automatizar o acesso aos dados, pois a cada dia que passa as organizações recebem mais informações para serem armazenadas. As empresas que se recusam automatizar os dados estão mais propícias ao fracasso, necessitando de um controle eficiente e eficaz, isso vale também para as instituições de ensino superior que lidam com a grande quantidade de informações de seus alunos. Observa-se que é um direito e necessidade de os estudantes conferirem e contabilizarem as horas cumpridas, porém, muitas vezes a forma de registro não é adequada, o que sugere a viabilização uma solução tecnológica que possa trazer amplo acesso às informações, tanto por parte dos discentes como parte das instituições.

Buscando evitar divergências no registro ou ausência de informações, esse trabalho apresenta o processo de construção de uma API simples e de qualidade para contabilizar as horas complementares dos discentes dos mais diversos cursos e instituições, e um front end mobile-first em REACT (biblioteca JavaScript de código aberto para criar interfaces) para a apresentação gráfica deste projeto.

Foi desenvolvido uma API WEB que possibilita armazenar o certificado do curso e as horas de atividades efetuadas pelo aluno, permitindo a gerência dessas informações e facilitando o registro das horas e as consultas desses dados. O aluno terá maior controle sobre seus envios dos certificados e visualização de suas horas. O discente terá uma tela de visualização dos seus certificados que já foram validados, tendo maior integridade dos dados armazenados e ganhando controle sobre o somatório total das horas enviadas. O usuário docente poderá visualizar o certificado e realizar a validação das horas complementares de cada atividade por aluno, aprovando ou reprovando.

1.1. OBJETIVO GERAL

Criação de uma API open source com um front end responsivo para sua apresentação.

1.2. OBJETIVOS ESPECÍFICOS

- Construir um front end mobile-first para o consumo futuro desta API utilizando a biblioteca de código aberto REACT.
- Validar as informações armazenadas na API, mantendo a integridade dos dados.
- Possibilitar a criação de uma camada visual em qualquer linguagem ou dispositivo.

2. REFERENCIAL TEÓRICO

Neste cenário de transformações advindas na “sociedade da informação”, surge a necessidade de conhecimentos diversos que permitam aos indivíduos desenvolverem atividades variadas, como também a resolução de problemas relacionados com seu cotidiano social, educacional e profissional. (DUDZIAK, 2001).

3. PROCEDIMENTOS METODOLÓGICOS

Segundo PIRES (2016), o funcionamento de uma API se baseia em um ponto de acesso entre a aplicação e seu cliente, podendo ser ele um usuário ou uma aplicação. Nesse contexto, foi construído um ponto de acesso contendo todas as regras negociais funcionais, acessível via requisições HTTP (Hypertext Transfer Protocol), permitindo o consumo por parte de um usuário ou outro sistema, com as funcionalidades de registro das novas atividades complementares, de gerência das atividades por parte docente e de visualização das atividades e extratos pelos discentes. O sistema permite unificar o acesso, o controle e a gerência de forma segura, inclusive gerando os comprovantes de horas complementares apresentados pelos alunos. Do ponto de vista do coordenador, contribui com a melhoria do processo de validação. A ideia se concretizou após o desenvolvimento de um protótipo de um sistema com essas funcionalidades

3.1. REQUISITOS

Este conjunto de requisitos de software define os comportamentos do sistema. Divididos em requisitos funcionais e não funcionais, em que os requisitos não funcionais dizem a respeito das características dos padrões de qualidade e os funcionais se referem às funções, aos recursos implementados no sistema e ao que o sistema deve fazer para cada usuário ou outras operações do sistema. Estes estão listados abaixo e figurado no caso de uso.

[RF001] Cadastro de Usuário

Permitir o cadastro de usuário do sistema, sendo necessário informar os dados pessoais do usuário a ser cadastrado.

[RF002] Realização da Autenticação

Permitir que todos os usuários cadastrados realizem acesso ao sistema após a inserção de suas credenciais.

[RF003] Gerência de Atividade (Discente)

Permitir que todos os usuários cadastrados realizem acesso ao sistema após a inserção de suas

credenciais.

[RF004] Gerência de atividade (Docente)

Possibilitar ao docente visualizar as atividades complementares por aluno, filtrar as atividades por situação e matrícula, buscar uma atividade complementar, visualizar extrato por aluno e validar as atividades complementares por aluno.

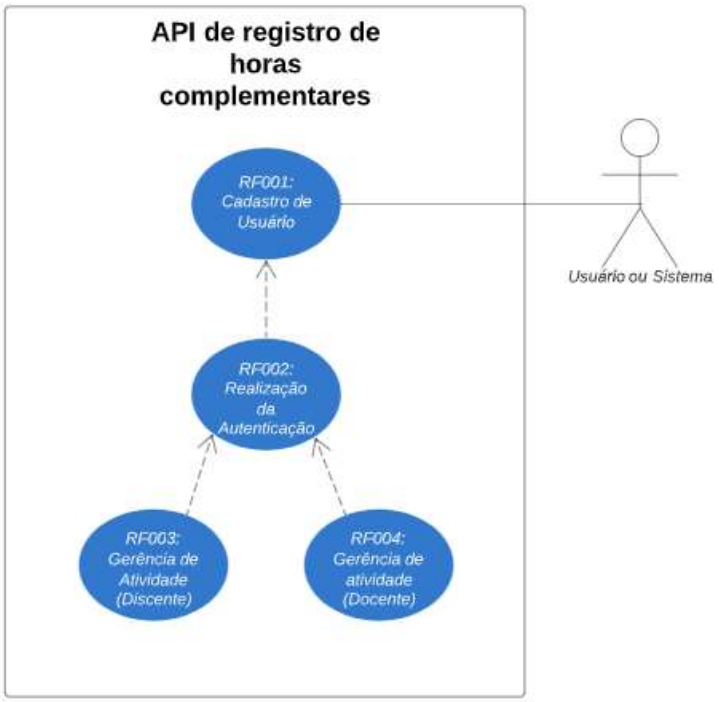
[RNF001] Compatibilidade de navegadores

Sistema deverá ser compatível com a versão mais recente dos navegadores *Google Chrome*, *Mozilla Firefox* e o *Microsoft Edge*.

3.1.1. DIAGRAMA DE CASO DE USO

O sistema terá na tela principal as opções de login e criar conta. Na realização do cadastro, o novo usuário será titulado como discente, ao finalizar o cadastro a tela será redirecionada através do sistema. Após esse trâmite discente ou docente fará o login e terá suas permissões o que permite gerenciar suas atividades de acordo com o seu perfil. Abaixo segue o diagrama de caso de uso (Figura 1).

Figura 1 – Diagrama de casos de uso que representa o sistema.

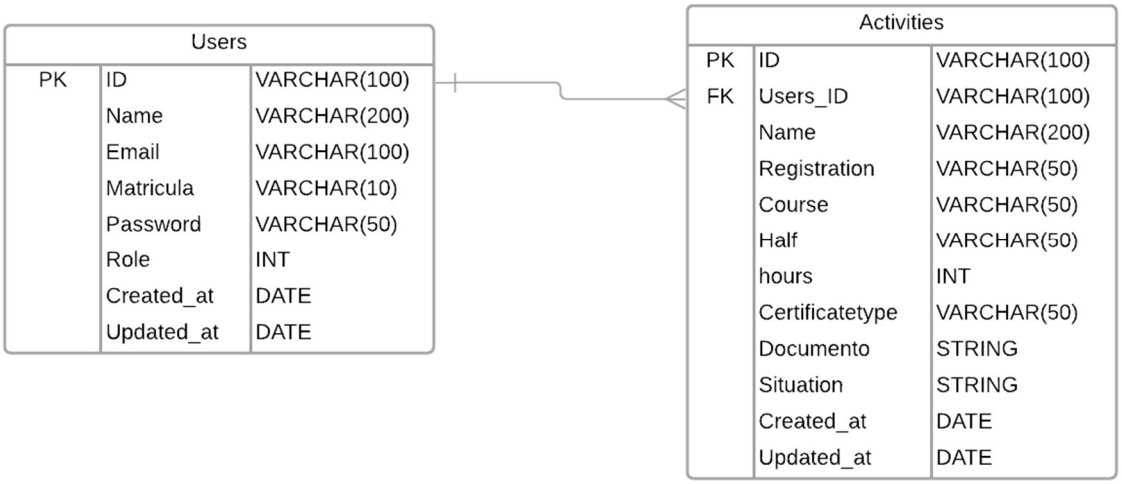


Fonte: Dos autores, 2020.

3.2. MODELO DE DADOS

Ao desenvolver o sistema, foi necessário criar um modelo de dados relacional que está representado na figura 2.

Figura 2 – Diagrama Relacional.



Fonte: Dos autores, 2020.

USERS:

A entidade do *Users* no sistema. Cada registro feito nessa tabela representa um aluno que terá um documento enviado.

ACTIVITIES:

Activities é utilizada para o registro de todos os envios dos certificados, cada certificado enviado para validação será verificado e contabilizado para as horas de atividades.

3.3. PADRÃO MVC

De acordo com MASSARI (2017), o padrão MVC (*Model View Controller* – Modelo, Visão e Controladora) é um padrão de arquitetura de software definido por camadas, tendo boas práticas, além de, deixar o código reutilizável caso seja necessário utilizar alguma função sem recriar o script.

3.4. OUTRAS TECNOLOGIAS APLICADAS

- **JAVASCRIPT:**

Uma linguagem de programação interpretada e estruturada de script em alto nível. Juntamente com *HTML* e *CSS*, o *JavaScript* é uma das três principais tecnologias da internet.

- **REACT:**

Uma biblioteca do *JavaScript* de código aberto que permite criar interfaces de usuário em páginas WEB.

- **NODEJS:**

É uma poderosa Engine, que possibilita rodar *JavaScript* a nível de servidor, permitindo criar robustas aplicações nessa linguagem.

- **POSTGRES:**

É um poderoso sistema de banco de dados objeto-relacional de código aberto com mais de 30 anos de desenvolvimento ativo, que lhe rendeu uma forte reputação de confiabilidade, robustez de recursos e desempenho.

- **REST:**

Conforme o dito por SOUZA (2020), o *Representational State Transfer* (REST) é um grupo de regras aplicadas para que as requisições HTTP respeitem as regras definidas na arquitetura.

- **MOBILE-FIRST:**

É uma ideia usada em aplicativos da WEB onde o foco principal da arquitetura e do desenvolvimento é os dispositivos móveis e, posteriormente, adaptada para o *desktop*.

4. APRESENTAÇÃO DE DADOS E ANÁLISE

A API REST atende as partes principais da necessidade de um sistema, sendo entregue com *endpoints* prontos para o consumo, com as funcionalidades principais como *login*, cadastro de uma atividade complementar, listagem de atividades complementares por aluno, listagem de solicitações para validação docente, visualização de uma atividade complementar, funcionalidade para validação docente, funcionalidade para recuperação de senha e alteração de dados pessoais.

4.1. API

A API foi desenvolvida como um código aberto, com sua comunicação toda em *JavaScript Object Notation* (JSON), tem como seu propósito a computação das horas complementares de qualquer instituição de ensino, necessitando apenas da construção de um front end para o consumo desta API. Construída para ser facilmente entendida, qualquer resultado contrário ao esperado resulta em um erro tratado com mensagem auto explicativa. A API será entregue com os seguintes *endpoints*:

- **/LOGIN**

Esta rota espera uma requisição do tipo *post*, com os dados de matrícula e senha em formato JSON. Ao receber os dados, a rota valida se existe um usuário cadastrado com a matrícula fornecida, e se a senha, armazenada de forma criptografada, confere com a senha informada, fornecendo como resposta um *Json Web Token* (JWT), realizando o *login*.

- **/USERS**

Esta rota espera uma requisição do tipo *post* ou *get*. Ao receber uma requisição *post*, com os dados: nome, e-mail, matrícula e senha. Com isso, a API valida se já existe algum usuário com estes dados para evitar duplicidades, e logo após cria o usuário no banco como discente, sempre armazenando somente a senha encriptada e retorna um JSON com os dados do usuário cadastrado.

Ao receber uma requisição *get*, sem corpo, a API retorna uma listagem em JSON de todos os usuários cadastrados com o status ativo somente, nunca exibindo suas senhas na listagem.

- **/USERS/:USER_ID**

Esta rota espera uma requisição do tipo *get*, *put* ou *delete*. Ao receber uma requisição *get*, com os identificados (*id)* do usuário passado como parâmetro retorna um JSON contendo o usuário vinculado ao id informado. Ao receber uma requisição *put*, com o id do usuário passado como parâmetro, e o nome, e-mail e matrícula como corpo da requisição, a API verifica se existe um usuário com este id e altera os dados para os novos informados no corpo, retornando o usuário já alterado.

Por fim, ao receber uma requisição delete, com o id do usuário passado como parâmetro, altera a coluna *active* de 1 para 0, excluindo logicamente e impedindo o registro de ser exibido na listagem comum, passando a ser exibido somente na listagem dos usuários excluídos.

- **/USERS/DELETED**

Esta rota espera uma requisição do tipo *get* e retorna um JSON contendo os usuários excluídos.

- **/USERS/:USER_ID/ACTIVITIES:**

Esta rota espera uma requisição do tipo *get* ou *post*. Ao receber uma requisição *post*, com o id do usuário passado como parâmetro, e os dados: nome, matrícula, curso, semestre, total de horas, tipo do certificado, anexo e situação informados no corpo da requisição. Diante disso, a API encontra o usuário dono do id fornecido, cadastra uma nova atividade vinculada ao usuário e retorna como resposta um JSON com a atividade cadastrada. Ao receber uma

requisição *get*, com o id do usuário passado como parâmetro a API retorna um JSON com os dados do usuário e suas atividades vinculadas.

- **/FORGOT_PASSWORD:**

Esta rota espera uma requisição do tipo *post* com o e-mail do usuário passado no corpo, a API encontra o usuário dono do e-mail informado, soma uma hora ao horário atual e armazena este horário novo como um token de expiração de prazo junto ao usuário e envia um e-mail contendo o *link* para troca de senha.

- **/RESET_PASSWORD:**

Esta rota espera uma requisição do tipo *post* com os dados de e-mail, *token* (gerado na rota "/forgot_password") e a nova senha passados no corpo. A API encontra o usuário dono do e-mail informado, verifica se o horário atual é menor ou igual ao armazenado no *token* de expiração de prazo e então altera a senha do usuário retornando uma mensagem de sucesso.

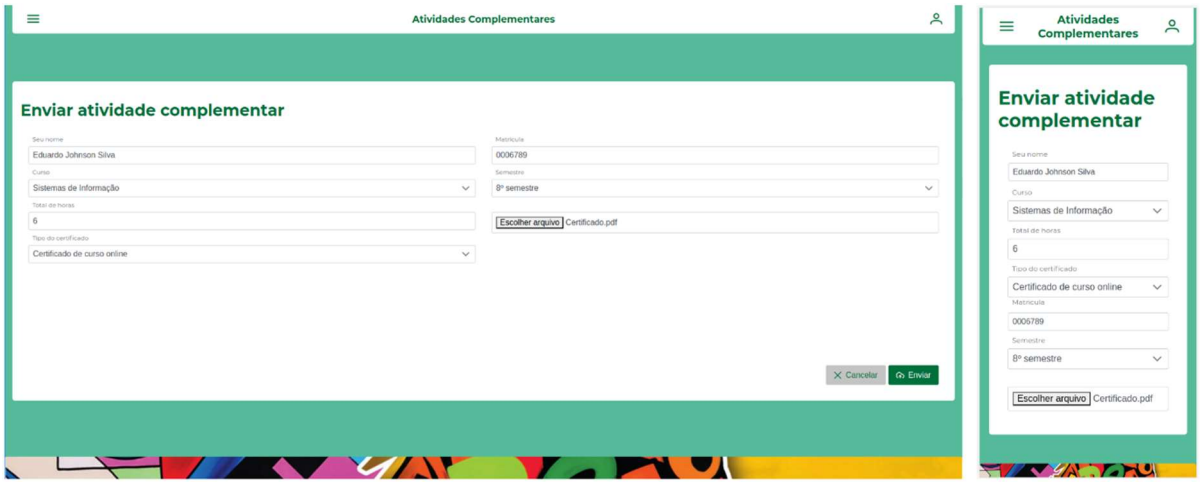
4.2. FRONTEND MODELO

Foi desenvolvido um *frontend mobile-first* responsivo com o objetivo futuro de consumir a API desenvolvida, contando com telas de login, criação de conta, pesquisa de atividades, envio de atividade, visualização de atividade, pesquisa de solicitações, validação de atividade, pesquisa de discentes e visualização de discente. O *frontend* será entregue em REACT, totalmente dividido em componentes de fácil entendimento, facilitando assim a futura integração com a API desenvolvida. Abaixo algumas das telas:

- **TELA PARA ENVIO DE ATIVIDADE:**

O aluno usará a guia "enviar atividade complementar" (Figura 3) para realizar seus certificados. Assim, cada aluno vai possuir controle de seus envios de certificados, podendo visualizar o estado dos seus envios. Caso seja aprovado, esse certificado será somado com os outros envios já validados.

Figura 3 – Tela de envio

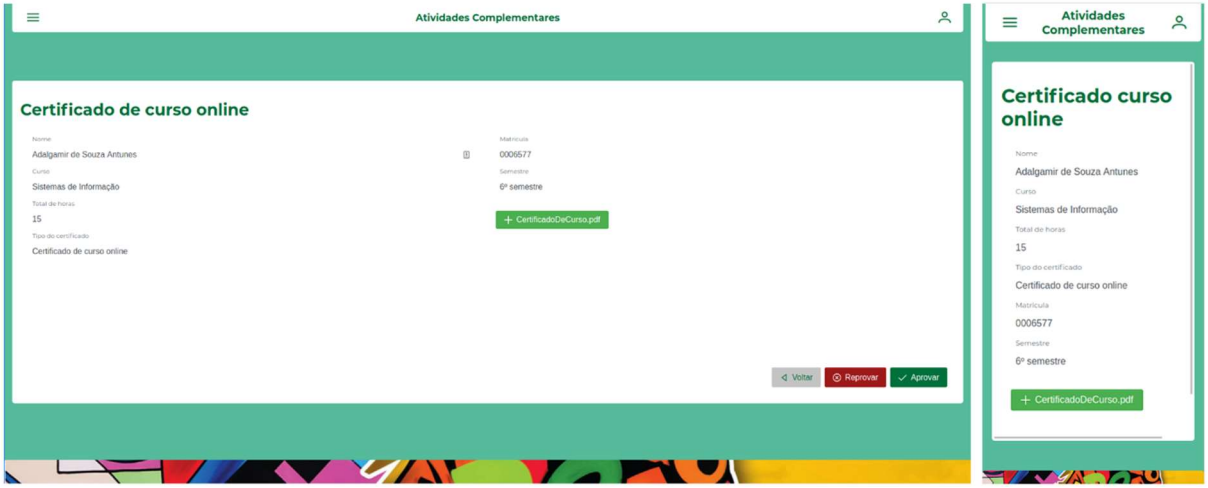


Fonte: Dos autores, 2020.

- **TELA PARA VALIDAÇÃO DE ATIVIDADE:**

O docente usará a guia abaixo (Figura 4) para realizar a validação dos certificados enviados pelos alunos, permitindo o *download* do anexo para a conferência das informações, podendo validar ou não uma atividade complementar institucional.

Figura 4 – Tela de validação

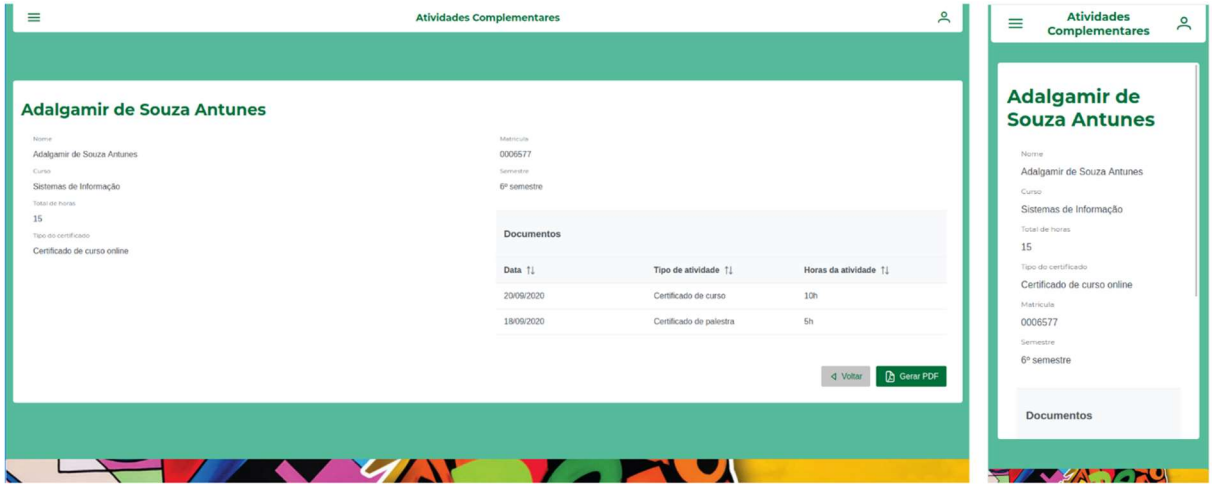


Fonte: Dos autores, 2020.

- **TELA PARA EXTRATO POR ALUNO:**

O docente usará a guia abaixo (Figura 5) para visualizar o extrato por aluno, podendo exportar o resultado em PDF *(Portable Document Format*).

Figura 5 – Tela de extrato



Fonte: Dos autores, 2020.

5. CONSIDERAÇÕES FINAIS

O desenvolvimento da API de atividades complementares visa possibilitar uma melhor eficiência no envio, acompanhamento e avaliação das horas cumpridas pelos discente de qualquer instituição de ensino, tornando prática e uniforme a execução do processo de cadastro das atividades. Constata-se também que se faz necessária a futura integração do *frontend* com a API, necessitando de esforços futuros para o consumo dela. O projeto foi construído para ser de código aberto, permitindo o uso pessoal e comercial. O foco foi possibilitar o uso desta API por qualquer camada de visão construída em qualquer linguagem, podendo ser criado até mesmo outro frontend mantendo os dados e a lógica da aplicação, reduzindo tempo e custos futuros.

Com o encerramento deste projeto, conclui-se que foi proveitoso para a fixação dos conhecimentos nas tecnologias adotadas e na contribuição para a comunidade acadêmica. Portanto, analisado o desenvolvimento das aplicações da API com base nos objetivos propostos para este projeto, conclui-se que os mesmos foram alcançados, resultando em uma API open source para a comunidade e um *frontend mobile-first*.

REFERÊNCIAS

GARCIA, M. **Informática aplicada a negócios**. 1. ed. Rio de Janeiro: Brasport Livros e Multimídia Ltda., 2005. Citado na página 4.

PIRES, Jackson. **O que é API REST e RESTful**. 2016. Disponível em: HTTPs://becode.com.br/o-que-e-API-REST-e-RESTful/. Acesso em: 26 nov. 2020. Citado na página 4.

DUDZIAK, Elizabeth Adriana. **A information literacy e o papel educacional das bibliotecas**. 2001. 187 f. Dissertação (Mestrado em Ciências da Comunicação) - Escola de Comunicação e Artes. Universidade de São Paulo, São Paulo. Acesso em: 26 Out 2020. Citado na página 5.

MASSARI, Jorge. **O que é Model-View-Controller (MVC).** 2017. Disponível em: HTTPs://www.portalgsti.com.br/2017/08/padrao-mvc-arquitetura-model-view-controller.html. Acesso em: 20 nov. 2020. Citado na página 7.

SOUZA, Ivan de. **Entenda o que é REST API:** e a importância dele para o site da sua empresa. e a importância dele para o site da sua empresa. 2020. Disponível em: HTTPs://rockcontent.com/br/blog/REST-API/. Acesso em: 26 nov. 2020. Citado na página 8.