ISSN: 2319-0124
josif 2023
15ª JORNADA CIENTÍFICA E TECNOLÓGICA
12º SIMPÓSIO DE PÓS-GRADUAÇÃO
DO IFSULDEMINAS
REALIZAÇÃO INSTITUTO FEDERAL Sul de Minas Gerais
APOIO FAPEMIG CNPq

CONSTRUÇÃO E ARQUITETURA DE ROTAS CADASTRAIS PARA A PLATAFORMA ARTE DE CADERNO

Rebeca D. ROSA,[1]; Beatriz P. NEAIME[2]; Caroline F. MELO.[3]; Cristhian C. BARBOSA[4] ; Douglas. F.S. NUNES[5]; Giselle C. CARDOSO[6]; Jônata M. SOUSA[7]; Márcio L. BESS[8]

RESUMO

O Arte de Caderno é um projeto que visa estimular a arte dentro dos colégios, através do envio de desenhos feito por alunos de diversas escolas do Brasil. Conforme a demanda do projeto foi crescendo, tornou-se necessário utilizar tecnologia para catalogar, avaliar e cadastrar esses desenhos. Por isso, o seguinte trabalho mostra como foi feito o desenvolvimento da conexão entre o banco de dados e a interface do usuário (frontend) através de rotas para listar, adicionar, atualizar e deletar informações de cadastro. Com isso, o frontend conseguiu acessar as informações necessárias, além de realizar manipulações de dados com respostas adequadas e com mapeamento de erros.

Palavras-chave:
Tecnologia; Backend; Desenvolvimento Web.

1. INTRODUÇÃO

Idealizado pelo professor Márcio Luiz Bess, o projeto Arte de Caderno surgiu em 2009 em Santa Catarina. Conforme Bess et al. (2017), o projeto tem o formato de concurso, objetivando uma ação educativa de promover a arte, além de proteger o patrimônio público, incentivando a produção artística nos meios corretos. Este recebe desenhos de diversas localidades do país, que são analisados e colocados em votação em redes sociais para eleger as melhores obras.

De acordo com Bess et al. (2017), em 2014 o Arte de Caderno alcançou o resultado de 1458 obras cadastradas. Já em 2015, foram 2354 obras inscritas de 12 estados diferentes. Tendo em vista o crescimento das obras cadastradas e o aumento da abrangência do projeto, tornou-se inviável realizar o processo do concurso de forma manual e descentralizada. Para tanto, surgiu a proposta de migrar esses processos para uma plataforma digital.

Em virtude disso, o objetivo geral deste trabalho é apresentar o desenvolvimento do

[1]Discente de Bacharelado de Engenharia de Computação, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: rebeca.rosa@alunos.ifsuldeminas.edu.br
[2]Bolsista de Extensão, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: beatriz.neaime@alunos.ifsuldeminas.edu.br
[3]Bolsista de Extensão, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: caroline.melo@alunos.ifsuldeminas.edu.br
[4]Discente de Bacharelado de Engenharia de Computação, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: cristhian.barbosa@alunos.ifsuldeminas.edu.br
[5]Orientador, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: douglas.nunes@ifsuldeminas.edu.br
[6]Docente de Bacharelado de Engenharia de Computação, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: giselle.cardoso@ifsuldeminas.edu.br
[7]Bolsista de Extensão, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: jonata.martins@alunos.ifsuldeminas.edu.br
[8]Docente de Artes, IFSULDEMINAS - *campus* Poços de Caldas. E-mail: marcio.bess@ifsuldeminas.edu.br

backend, mais especificamente as rotas de cadastros da aplicação. Com isso, pretendeu-se alcançar um servidor que possa atender o *frontend* com rotas que recebem e devolvem dados correspondentes de forma correta.

2. FUNDAMENTAÇÃO TEÓRICA

De acordo com o MDN Web Docs (2023), o *backend* processa e gerencia os dados por trás das cenas. Ele executa a lógica do aplicativo, realiza cálculos e manipulações nos dados, antes de enviar as respostas ao *frontend* (interface visual do site) para exibição. O desenvolvimento do *backend* pode ser feito utilizando diversas linguagens, entre elas o *NodeJs*.

Conforme a documentação do *NodeJs* (2023), ele é um ambiente em tempo de execução de código aberto que utiliza a linguagem *Javascript*. Juntamente, pode-se utilizar uma biblioteca chamada *Express*. De acordo com a documentação do *Express*, "O *Express* é um framework para aplicativo da web do *Node.js* mínimo e flexível que fornece um conjunto robusto de recursos para aplicativos web e móvel" (Express.js, 2017).

Através da combinação destas tecnologias, pode-se realizar a construção de rotas, ou endpoints de API. De acordo com o IBM Cloud Education, uma API (Interface de Programação de Aplicativos) é o conjunto de regras que permite a comunicação entre diferentes softwares de maneira padronizada. Para isso, utiliza-se os *endpoints* ou rotas, que é a forma como os serviços de API são acessados por outros softwares. Essas rotas podem ser desenvolvidas utilizando o método CRUD - *Create, Read, Update and Delete*-, ou seja, podendo criar, ler, atualizar e deletar informações.

Existem métodos para o acesso desses endpoints. O método GET é usado para solicitar dados de um servidor, o método POST é usado para enviar dados ao servidor, o método PUT é utilizado para atualizar ou substituir informações existentes no servidor e, por fim, o método DELETE é responsável por solicitar a remoção de recursos..

3. MATERIAL E MÉTODOS

Para a construção do backend, foi realizado primeiramente o levantamento de requisitos, definindo as informações que o frontend precisaria acessar.

- Rotas para inserção, exclusão, edição e listagem dos cadastros de alunos, escolas, professores, obras e avaliadores;
- Rotas para validação de Cadastro de Pessoa Física (CPF);
- Rotas para a busca de endereço através do Código de Endereçamento Postal (CEP);

Para realizar o desenvolvimento das, utilizou o *NodeJs* juntamente com o *Express*. Com isso, realizou-se o desenvolvimento da aplicação, utilizando o método CRUD. Para a criação e

atualização, utilizou-se o método *post*, para a leitura de dados, utilizou-se o *get* e para a exclusão, usou-se o *delete*.

Também foi realizado o mapeamento de erros, ou seja, se houver alguma requisição incorreta, erro no servidor, ou algum dado não encontrado, o servidor devolve um código de status seguido por uma mensagem. Após o desenvolvimento, foi realizada uma documentação a fim de que as rotas pudessem ser utilizadas corretamente pelo *frontend*.

4. RESULTADOS E DISCUSSÃO

Com isso, conforme a Tabela 1, foi possível realizar a construção dos seguintes *endpoints* para o projeto Arte de Caderno e as Figuras 1 e 2 mostram trechos de códigos que realizaram o desenvolvimento das rotas de CEP e de escolas.

Tipo de Rota	Endpoints	Método
Listar professores, listar escolas e listar alunos	/professor, /school, /student	GET
Listar escolas por professor	/professor/school/:id	GET
Listar estudantes por professor	/professor/getStudent/:id	GET
Inserir estudante pelo professor	/professor/student/:id	POST
Cadastrar professor, cadastrar escola e cadastrar aluno.	/professor, /school, /student	POST
Atualizar professor e atualizar aluno	/professor/update/:id, /student/update/:id	POST
Deletar professor e deletar estudante	/professor/:id, /student/:id	DELETE
Validar CPF	/cpf/:cpf	GET
Buscar endereço pelo CEP	/cep/:cep	GET

Tabela 1 - Lista de endpoints da aplicação
Fonte: Elaborado pelo autor

```
const cepRouter = express.Router();

cepRouter.get('/cep/:cep', async (req, res) => {
    const { cep } = req.params;

    let url = `https://viacep.com.br/ws/${cep}/json/`;
    let options = { method: 'GET' };

    try {
        const a = await fetch(url, options);
        const b = await a.json();
        res.json(b);
    } catch (error) {
        res.json(error);
    }
});

export default cepRouter;
```

Figura 1: Rota para busca de CEP
Fonte: Elaborado pelo autor

```
const schoolRouter = express.Router();
schoolRouter.use(authenticateTokenJwt);

schoolRouter.get("/school", SchoolController.listSchool)
            .get("/school/:id", SchoolController.getSchoolById)
            .post("/school", SchoolController.insertSchool)
            .post("/school/city", SchoolController.listCitiesByUf)
            .get("/school/uf", SchoolController.listUfs)
            .post("/school/listByCity", SchoolController.listSchoolByCity);

export default schoolRouter;
```

Figura 2: Rota de escola e os seus endpoints
Fonte: Elaborado pelo autor

Para cada resposta, também foi devolvido uma resposta padrão, a fim de facilitar o desenvolvedor *frontend* a identificar se a requisição foi um erro ou um sucesso. Essa resposta é constituída por uma mensagem e um código de status, conforme mostram as Tabelas 2 e 3.

Status	Mensagem
200	Sucesso
201	Inserido com sucesso

Tabela 2: Mapeamento de resposta com sucesso
Fonte: Elaborado pelo autor

Status	Mensagem
400	Má Requisição
404	Não Encontrado
500	Erro no servidor

Tabela 3: Mapeamento de resposta com erro
Fonte: Elaborado pelo autor

5. CONCLUSÃO

Com as rotas desenvolvidas e o mapeamento das respostas feito, o *frontend* conseguiu realizar requisições necessárias para que o usuário seja capaz de cadastrar, atualizar, consultar e deletar informações no banco de dados.

Futuramente, deve-se implementar as rotas de CRUD para o desenho e avaliadores de desenho. Também espera-se deixar o servidor mais robusto e escalável, permitindo o acesso simultâneo de várias pessoas ao mesmo tempo. Por último, realizar testes unitários para cada rota, a fim de cobrir possíveis erros e falhas.

REFERÊNCIAS

BESS, Márcio et al. Arte de Caderno. 8ª JORNADA CIENTÍFICA E TECNOLÓGICA E 5º SIMPÓSIO DA PÓS-GRADUAÇÃO DO IFSULDEMINAS. 2017. Disponível em: https://memoriajornada.ifsuldeminas.edu.br/index.php/jcpas/jspas/paper/viewFile/2685/2017. Acesso em: 8 de agosto de 2023.

Express.js. Express Documentation, 2017. Disponível em: https://expressjs.com/pt-br/. Acesso em: 8 de agosto de 2023.

IBM Cloud Education. O que é uma API? Disponível em: https://www.ibm.com/cloud/learn/api. Acesso em: 8 de agosto de 2023.

MDN Web Docs. Introduction to the server side, 2023. Disponível em: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction#summary. Acesso em: 8 de agosto de 2023.

Node.js. In: Node.js Documentation, 2023. Disponível em: https://nodejs.org/en/about/. Acesso em: 8 de agosto de 2023.