ISSN: 2358-8829

ARTE DE CADERNO: CONSTRUÇÃO DO BACKEND E DESENVOLVIMENTO DO BANCO DE DADOS EM UMA PLATAFORMA DE ARTE DIGITAL[1]

Rebeca Domingos Rosa [2]
Jonata Martins de Sousa [3]
Douglas Fabiano de Sousa Nunes [4]
Giselle Cristina Cardoso[5]
Márcio Luiz Bess[6]

RESUMO

O Projeto Arte de Caderno é uma iniciativa nacional que visa promover a arte nas escolas públicas, incluindo concursos de desenhos realizados por alunos de todo o Brasil. Com a expansão desse concurso e a necessidade de um suporte mais eficiente, foi desenvolvida uma plataforma digital que permite o cadastro de usuários, o envio de desenhos e a avaliação dos mesmos. Este trabalho focou especificamente no desenvolvimento do *backend* (servidor) desta aplicação web, pois por meio desse servidor ocorre a comunicação entre a interface visual e o banco de dados. Optou-se pela utilização da tecnologia *NodeJs* com *Express* para a construção do servidor e das rotas. Além disso, foram realizados testes com a ferramenta *Jest*, a documentação foi feita com linguagem de marcação e o código foi versionado usando *GitHub*. Isso resultou em uma aplicação *backend* funcional, bem documentada e organizada, capaz de suportar o fluxo de dados da plataforma Arte de Caderno de forma eficiente e segura. Simultaneamente, este artigo abordou a concepção de um banco de dados *NoSQL*, especificamente o *MongoDB*, para a plataforma Arte de Caderno. O escopo do banco de dados foi desenhado para preservar os registros das submissões dos concursos de forma a simplificar o controle das submissões artísticas e agilizar sua avaliação subsequente. Essa integração do banco de dados à plataforma representa uma abordagem inovadora para otimizar a eficiência e a eficácia do gerenciamento do Projeto Arte de Caderno. Com o banco de dados integrado, foi possível ter um controle mais preciso e seguro das submissões, além de uma administração simplificada do fluxo de dados. Assim, este trabalho propõe não apenas o desenvolvimento do *backend*, mas também a criação de um banco de dados integrado, ambos elementos essenciais para o sucesso e a escalabilidade da plataforma Arte de Caderno.

Palavras-chave: *Backend*. Servidor. Banco de Dados. *NodeJS*. *MongoDB*.

---

[1] Projeto fomentado pelo EDITAL PROEX 22/2023 - APOIO A PROJETOS DE ARTE E CULTURA do Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais - IFSULDEMINAS e realizado pelo Laboratório de Criatividade VOA IFSULDEMINAS campus Poços de Caldas.
[2]Bacharela em Engenharia da Computação pelo IFSULDEMINAS campus Poços de Caldas - rebeca.domingos.rosa@gmail.com;
[3]Discente no curso de Engenharia de Computação no IFSULDEMINAS campus Poços de Caldas. jonata.martins@alunos.ifsuldeminas.edu.br;
[4]Doutor em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo, USP. douglas.nunes@ifsuldeminas.edu.br;
[5]Mestre em Engenharia Elétrica pela Universidade Estadual de Campinas, UNICAMP. giselle.cardoso@ifsuldeminas.edu.br;
[6]Doutor em Desenvolvimento Humano e Tecnologias pela Universidade Estadual Paulista Júlio de Mesquita Filho, UNESP. marcio.bess@ifsuldeminas.edu.br.

ISSN: 2358-8829

INTRODUÇÃO

Idealizado pelo professor Márcio Luiz Bess[7], o projeto Arte de Caderno surgiu em 2009, no estado de Santa Catarina. O projeto tem o formato de concurso, objetivando uma ação educativa de promoção da arte, incentivando a produção artística nos meios corretos, e de proteção do patrimônio público.

Atualmente sendo executado no Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais - *campus* Poços de Caldas, o concurso tem recebido desenhos da comunidade externa e interna e de diversas localidades do país.

De acordo com Bess *et al.* (2017), em 2014 o Arte de Caderno alcançou um total de 1458 obras inscritas. Em 2015, foram 2354 obras submetidas ao evento, sendo de 12 estados diferentes. Tendo em vista o crescimento das obras cadastradas e o aumento da abrangência geográfica do projeto, tornou-se inviável realizar o processo de inscrição, submissão, catalogação e avaliação dos desenhos 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 foi desenvolver o *backend*[8] e o banco de dados para o funcionamento da aplicação. Dentre os objetivos específicos buscados, estão: projeto e desenvolvimento do esquema de banco de dados; criação e implementação dos objetos no *MongoDB*[9]; criação dos *schemas*; desenvolvimento de um servidor utilizando *NodeJs*[10] e *Express*[11]; criação de rotas de busca, inclusão, edição e exclusão de cadastro de professores, estudantes, avaliadores, desenhos e escolas; realização de uma validação de avaliações pendentes; realização de um tratamento de erro nas rotas e com o uso de *middlewares*; Conexão do sistema com o banco de dados através de esquemas; Manutenção de um versionamento de código através do *GitHub*[12], com *commits* organizados; teste e depuração das rotas criadas; e, por fim, documentação das funcionalidades do servidor e o banco de dados.

Esses objetivos permitiram a criação de um servidor e um banco de dados documentado, com uma cobertura de erros representativa, de modo a atender o *frontend* com rotas que recebem e devolvem adequadamente as informações requeridas.

---

[7] Lattes: http://lattes.cnpq.br/2142214809343643

[8] O backend é a parte de uma aplicação que liga a interface visual (frontend) às funcionalidades do sistema, como o banco de dados, a lógica de negócios, e outras operações do servidor. Ele é responsável por processar as solicitações dos usuários, interagir com o banco de dados, realizar cálculos, e retornar as informações ao frontend para que possam ser exibidas ao usuário.

[9] Veja mais em: https://www.mongodb.com/ja-jp

[10] Veja mais em: https://nodejs.org/en

[11] Veja mais em: https://expressjs.com/pt-br/

[12] Veja mais em: https://github.com/

ISSN: 2358-8829

METODOLOGIA

A metodologia utilizada neste trabalho foi prática, como tecnologia aplicada na educação, focada na construção e implementação do projeto descrito. O primeiro passo realizado foi o levantamento de requisitos, que envolveu reuniões com desenvolvedores e professores para identificar as necessidades do sistema Arte de Caderno. Os requisitos definidos incluíram o cadastro de usuários (professores, alunos e avaliadores), cadastro de escolas, uso de *APIs* para validação de CEP e CPF, cadastro de desenhos e a distribuição dos desenhos para três avaliadores diferentes, o que permitiu definir o escopo do banco de dados e as rotas do servidor.

Em seguida, foram escolhidas as tecnologias a serem utilizadas: *Node.js* (versão 19.3.0) para o servidor, *MongoDB* para o banco de dados, *GitHub* para o versionamento de código e Visual Studio Code para o desenvolvimento. As instalações dos *softwares* necessários foram realizadas nas máquinas de desenvolvimento, que neste projeto consistiu em um notebook de propósito geral, especificações intermediárias[13].

O próximo passo foi realizar as configurações do servidor, começando pela criação do arquivo *package.json*, que incluía nome, versão, descrição, nome do arquivo principal, *scripts*, dados do repositório remoto, dados do autor e dependências. Foi inicializado um repositório *Git* localmente e conectado a um repositório remoto no *GitHub* para armazenar e versionar o código, com *commits* claros e organizados a cada modificação.

Com a configuração e conexão ao banco de dados finalizadas, iniciou-se a construção dos modelos de documentos, conhecidos como *schemas*. Esses *schemas* definem a organização, estrutura e relações dos dados no banco de dados, estabelecendo regras para o armazenamento, organização, acesso e manipulação dos dados. Essa estrutura lógica abrange todas as tabelas, relações e restrições, garantindo consistência e facilitando a manutenção do sistema.

A partir disso, começou-se a construção das rotas para conectar o *frontend* (interface visual) com o banco de dados. Inicialmente, foram desenvolvidas as rotas de cadastro, criando os *endpoints* para: inserir estudantes, professores e escolas; listar os estados e cidades com escolas cadastradas; e por fim, listar as escolas pela cidade.

[13] Ryzen 5 4600G com Radeon Graphics 3.70 GHz, 8 Gigas de Ram

ISSN: 2358-8829

Depois, foram criadas duas rotas usando dados externos. A primeira delas foi a de busca de CEP utilizando um serviço chamado ViaCep[14], que é um *webservice* para consulta de CEP. A segunda foi com o uso da biblioteca validar-cpf , criando uma rota que recebe o CPF no parâmetro do *endpoint* e retorna se o mesmo é ou não válido. Ambos os *endpoints* foram importantes nas telas de cadastro de usuários. Subsequentemente, foram desenvolvidas mais rotas para os usuários de professor e de estudante, tal como: listar todos os estudantes e professores; listar o estudante ou professor pelo seu código de identificação no banco; atualizar informações de professor ou estudante; deletar professor ou estudante; listar os desenhos cadastrados por estudante; possibilitar o professor inserir um aluno pelo seu usuário; listar estudantes por professor através de seu id; e listar as escolas por professor através de seu id.

Também foram desenvolvidas rotas para os avaliadores, tais como: inserir, listar todos, listar pelo código de identificação e listar desenhos a serem avaliados por cada jurado.

Em relação às rotas de desenho, foram desenvolvidas rotas para: listar todos os desenhos e buscar um desenho pelo id; buscar desenhos por estudante; buscar desenhos pela categoria; inserir um desenho; avaliar um desenho ou desclassificar; e distribuir os desenhos aos avaliadores.

Após o prazo de submissão, os desenhos são distribuídos entre os avaliadores, garantindo que cada um seja avaliado por três pessoas diferentes. Um código automatiza o processo, verificando desenhos pendentes e enviando *e-mails* aos avaliadores. Se uma obra não for avaliada em 10 dias, ela é redistribuída. Utiliza-se a biblioteca *NodeCron*[15] para agendar essas tarefas em *Node.js*.

As rotas foram construídas com métodos *HTTP*: *Get* para buscas, *Post* para inserção e atualização, e *Delete* para exclusão. Cada rota tem um controlador específico que define a lógica e trata os erros. Um *middleware* centralizado gerencia erros de servidor e de validação de dados.

Os testes de integração, realizados com *Jest* e *SuperTest*, abrangeram rotas de professor, estudante, desenho, cadastro, avaliadores e escola, utilizando técnicas de *mocking* para simular diferentes tipos de requisições e corrigir inconsistências nas funções de controle dos *endpoints*.

[14] Veja mais em: https://viacep.com.br/
[15] Veja mais em: https://www.npmjs.com/package/node-cron

ISSN: 2358-8829

A documentação, feita em *MarkDown*[16], descreve as rotas, métodos *HTTP*, requisições esperadas e respostas possíveis, utilizando tabelas para estruturar os objetos e variações de tamanhos de letras para hierarquizar o texto.

REFERENCIAL TEÓRICO

O *backend* é a parte do *software* que lida com a lógica de negócios, processamento de dados, armazenamento de dados e interação com o banco de dados. Conforme Duarte (2017c, 5ed., p.17), "todo o núcleo da sua aplicação fica longe do usuário, em um servidor web, que tratará as requisições de todos usuários em um único lugar, com um único código fonte".

O desenvolvimento do *backend* pode ser feito em diversas linguagens, como Python, Java ou Javascript. De acordo com Duarte (2017b), Javascript é uma linguagem que suporta *scripts* - programas que são usados para automatizar tarefas ou executar ações específicas - de tipagem dinâmica, ou seja, os tipos são associados com valores e não com variáveis. Para iniciar um servidor com essa linguagem, utilizoua-se do *NodeJs* , que é "um ambiente de execução de código JavaScript no lado do servidor, *open-source* e multiplataforma" (Duarte, 2017b, p.11).

Para tornar os serviços de *backend* disponíveis aos clientes externos, como sites e aplicativos móveis, nos valemos das chamadas *API's*. Uma *API* (Interface de Programação de Aplicativos), é, conforme Gaitatzis (2019, p.1, tradução nossa), um programa de computador que se comunica com outros programas, fornecendo uma interface de comunicação para os aplicativos". Gaitatzis conceitua uma Representational State Transfer (REST) *API* como "um tipo de *API* hospedada por um servidor web, geralmente fornecendo ou analisando dados para clientes por meio da web"(Gaitatzis, 2019, p.1, tradução nossa).

Uma REST *API* é acessível através de um *endpoint*, que é a representação a partir de uma identificação chamada URL para um serviço ou recurso oferecido por essa *API*. Uma REST utiliza um protocolo chamado *HTTP* (*HyperText Transfer Protocol*) em que Gaitatzis (2019) define como sendo a forma de transporte de informações pela internet, sendo o cliente enviando requisições para o servidor que devolve uma resposta ao cliente.

Uma chamada *HTTP* contém cabeçalhos, uma *URI*, um método e um *payload* (Gaitatzis, 2019). O cabeçalho inclui informações importantes, como *tokens* de autorização. A *URI* (*Uniform Resource Identifier*) identifica a chamada via endereço web. Os métodos

[16] Veja mais em: https://www.markdownguide.org/

ISSN: 2358-8829

podem ser GET, POST, PUT, PATH e DELETE. Segundo Brown (2019), uma solicitação GET consulta dados sem passar informações no corpo da requisição, enquanto uma solicitação POST envia um *payload* no corpo. Na solicitação DELETE, a identificação do objeto é passada na URI para apagá-lo. O *payload* pode ser em formato *JSON*, que, conforme Duarte (2017b, 3ed., p.187), "pode-se representar qualquer estrutura de dados em um objeto JSON, uma vez que são 'orientados a objetos."

O *Express* é, conforme Duarte (2017b, 3ed., p.187), "uma camada que fica entre o *HTTP server* criado usando o módulo *http* do *Node.js* e a sua aplicação web, interceptando cada uma das requisições". Com o *Express* utiliza-se *de* rotas, que manipulam as requisições a partir de funções feitas nos controladores.

Uma das funções do *backend* é conectar-se à base de dados, que pode ser relacional (SQL) ou não relacional (NoSQL). Duarte (2017a) explica que NoSQL não requer entidades e relacionamentos nem usa SQL para consultas e manipulação de dados. O *MongoDB Shell*, conforme sua documentação, "é um ambiente REPL JavaScript e *Node.js* para interagir com implantações do *MongoDB* no Atlas, localmente ou em outro *host* remoto." (*MongoDB*, 2023). Para conectar *Node.js* ao banco, foi usada a biblioteca *Mongoose*[17], que "fornece uma solução direta e baseada em esquema para modelar os dados da sua aplicação" (*Mongoose*, 2023). Com *Mongoose*, é possível conectar, criar modelos de documentos e realizar operações de busca, inserção, edição e exclusão de dados.

Também é uma boa prática realizar o versionamento de código. Conforme Regino (2022), versionamento é o gerenciamento de diferentes versões de um código ou de um projeto. Para isso, foi empregada a ferramenta Git que possibilitou o gerenciamento e o registro de histórico de alterações do trabalho. Outra ferramenta utilizada foi o *GitHub*, para armazenar e compartilhar o código online. Para registrar as modificações, pelo controle de versionamento, usam-se os *commits*, que segundo a documentação, servem para guardar os estados do seu repositório naquele momento (Git, 2023).

Documentar o código é uma tarefa que foi realizada durante e no final da implementação. Rocha diz que

> Efectivamente, um sistema bem documentado torna-se mais fácil de perceber por toda a equipa que o desenvolve, além de aumentar a eficácia e eficiência dos elementos da equipe quando é necessário realizar tarefas de manutenção (Rocha, 2008).

Portanto, documentar o código foi uma tarefa relevante no desenvolvimento. Pode-se utilizar diversas linguagens ou meios para realizar a documentação da aplicação, mas foi

[17] Veja mais em: https://mongoosejs.com/

ISSN: 2358-8829

utilizada a linguagem do tipo *MarkDown*. Segundo Simpkin (2021), "*markdown* se refere a: (1) um modo de formatação de ficheiros de texto, e também (2) uma ferramenta Perl para converter ficheiros Markdown em HTML".

RESULTADOS E DISCUSSÃO

Através da metodologia aplicada, foi construído um sistema em *NodeJs* com Express que faz uma conexão com o banco de dados *MongoDB* e escuta em uma porta determinada. Conseguiu-se atender as demandas da interface do aplicativo, através da criação de diversas rotas. A Tabela 1 mostra um balanço de quantas rotas foram criadas de acordo com seu tipo e com seu método HTTP. Foram desenvolvidas rotas abrangendo listagem, listagem por identificador, inserção, atualização e exclusão das informações do banco de dados. Além disso, também foram criados dois *endpoints* que utilizam dados externos, como a do ViaCep e a do Validar CPF. A Figura 1 mostra um exemplo de uso da rota de listar os estudantes cadastrados através de um professor na interface visual da aplicação do sistema Arte de Caderno, verificando sucesso na integração entre os sistemas e no funcionamento do banco de dados.

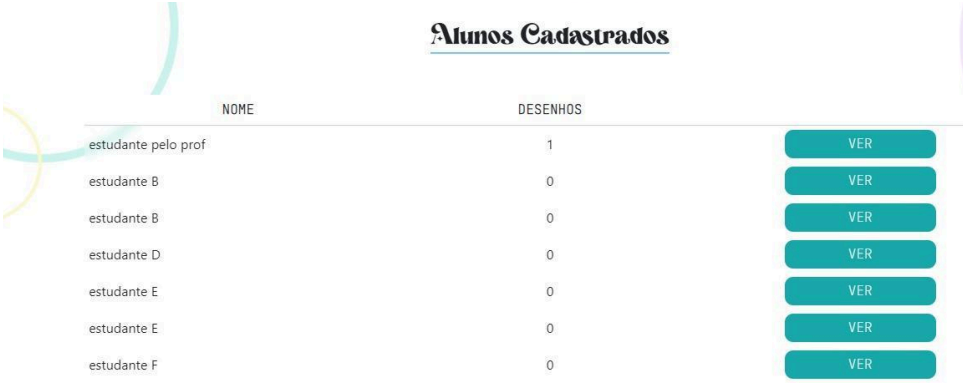
Tabela 1 - Quantidade de rotas por tipo

Nome do arquivo	Quantidade de rotas Get	Quantidades de rotas Post	Quantidade de rotas Delete
Professor	4	2	1
Estudante	3	1	1
Cadastro	2	5	0
Desenho	3	5	0
Avaliador	2	2	0
Escola	2	0	0
ViaCep e Validar CPF	2	0	0

Fonte: Elaborada pelo autor

ISSN: 2358-8829

Figura 6 - Tela de listar alunos por professor



Fonte: elaborada pelo autor

Também, foi possível resultar um código versionado, com *commits* feitos com frequência ao longo do desenvolvimento e com mensagens objetivas sobre as alterações feitas. Além disso, foi realizada uma documentação organizada, mostrando os *endpoints*, método *HTTP*, *request* (requisição) esperada e *response* (resposta) possíveis.

CONSIDERAÇÕES FINAIS

Em síntese, o desenvolvimento desta plataforma visa promover o aumento da expressão artística de forma organizada, contribuindo para o alcance cada vez maior dos objetivos do projeto Arte de Caderno, no qual podemos destacar a maior participação dos alunos nas escolas públicas. A plataforma apresenta-se como uma ferramenta robusta e adaptável para a submissão de desenhos de escolas públicas para concursos, com potencial para melhorias contínuas que irão aumentar sua funcionalidade e eficiência. Futuramente, espera-se automatizar a validação de professores e criar rotas para um usuário administrador.

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 abril de 2024.

BROWN, Ethan. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd ed. Sebastopol: O'Reilly Media, 2019.

DUARTE, Luiz. MongoDB para Iniciantes: Um Guia Prático. 4. ed. [s.l.]: LuizTools, 2017a.

ISSN: 2358-8829

_____, Luiz. Node.js e Microservices: Um Guia Prático. 3. ed. [s.l.]: LuizTools, 2017b.

_____, Luiz. Programação Web com Node.js: completo, do Front-end ao Back-end. 5. ed. [s.l.]: LuizTools, 2017c.

GAITATZIS, Adonis. Learn REST APIs: Your guide to how to find, learn, and connect to the REST APIs that powers the Internet of Things revolution. 2. ed. [s.l.]: BackupBrain Press, 2019.

Git. Documentação do Git. Disponível em: https://git-scm.com/doc. Acesso em: 15 de setembro de 2023.

MongoDB. Documentação do MongoDB Shell. Disponível em: https://www.mongodb.com/docs/mongodb-shell/. Acesso em: 12 de Agosto de 2024.

Mongoose. Documentação do Mongoose. Disponível em: https://mongoosejs.com/docs/guide.html. Acesso em: 12 de Agosto de 2024.

ROCHA, Nuno. Documentação de Software: Integração de Ferramentas de Modelação e Processamento de Texto. 2008. Dissertação (Mestrado) - Curso de Engenharia da Informática e Computação, Universidade do Porto, Porto, 2008. Disponível em: https://repositorio-aberto.up.pt/bitstream/10216/59642/1/000129520.pdf. Acesso em: 9 de abril de 2024.