Minha Missa Core: uma solução tecnológica para apoiar na divulgação de eventos católicos

Walisson R. Ferraz[1], Felipe Q. Bueno[1], Gabriel L. Freitas[1], Daniel D. Alves[1]

[1]Instituto Federal de Mato Grosso (IFMT)
Rondonópolis – MT – Brasil

{walissonrodriferraz, lipemotog4plus, gabriel.laroca14}@gmail.com,
daniel.alves@ifmt.edu.br

Abstract. *The Catholic Church of Rondonópolis-MT publicizes events on social networks and masses, but this has been inefficient, as only those present at masses receive the information. This article presents the development of an Application Programming Interface (API) to improve this dissemination, using interviews to elicit requirements and the Scrum framework for team management and collaboration. The API was developed with ASP.NET Core, Entity Framework Core and Swashbuckle (Swagger), connected to a SQL Server database, meeting the expectations of a Minimum Viable Product (MVP). API aims to facilitate the publication of Catholic events in the city.*

Resumo. *A Igreja Católica de Rondonópolis-MT divulga eventos em redes sociais e missas, mas isso tem sido ineficiente, pois apenas os presentes nas missas recebem as informações. Este artigo apresenta o desenvolvimento de uma Application Programming Interface (API) para melhorar essa divulgação, utilizando entrevistas para elicitar requisitos e o framework Scrum para o gerenciamento e colaboração da equipe. A API foi desenvolvida com ASP.NET Core, Entity Framework Core e Swashbuckle (Swagger), conectada a um banco de dados SQL Server atingindo as expectativas de um Mínimo Produto Viável (MVP). A API visa facilitar a publicação de eventos católicos na cidade.*

1. Introdução

Na cidade Rondonópolis-MT, 58% da população se declarou católica, conforme o censo de 2010 do Instituto Brasileiro de Geografia e Estatística (IBGE) (IBGE, 2010), que foi o último censo realizado especificando-se a religião por cidade. As paróquias de Rondonópolis-MT têm usado redes sociais, como Facebook© e Instagram©, para divulgar seus eventos religiosos, porém ainda há desafios no processo de comunicação, especialmente para fiéis que não frequentam regularmente ou estão visitando a cidade.

Com base nesses desafios, este estudo questionou: *Qual solução tecnológica pode contribuir na publicação de eventos católicos em Rondonópolis-MT?* Para resolver essa questão, o objetivo deste estudo foi desenvolver uma *Application Programming Interface* (API) para servir e armazenar os dados dos eventos, possibilitando maior comodidade na divulgação e acesso às informações pelas paróquias e fiéis.

Espera-se que o desenvolvimento do software contribua com a divulgação de eventos católicos em Rondonópolis-MT para atingir maior quantidade de fiéis e para dar-lhes comodidade e opções de eventos que a informação publicada na Internet possa trazer.

Este artigo apresenta os resultados de um projeto desenvolvido por duas equipes: i) *front-end*: responsável pelo planejamento do projeto, elicitação e especificação dos requisitos e design de interação; e ii) *back-end*: responsável pelo projeto do sistema (diagramas e projeto do banco de dados) e pelo desenvolvimento da API. Devido ao limite de páginas para a escrita deste artigo, serão apresentados apenas os resultados relacionados ao projeto e à implementação do Mínimo Produto Viável (MVP) da API "Minha Missa Core" e da estrutura de sua base de dados.

A Seção 2 descreve as principais definições e tecnologias utilizadas neste estudo. A Seção 3 apresenta o percurso metodológico, detalhando como este estudo foi conduzido. A Seção 4 descreve os principais resultados obtidos neste estudo, bem como discute esses resultados. Por fim, a Seção 5 apresenta as considerações finais e as limitações deste estudo.

2. Revisão da Literatura

A Igreja Católica é composta basicamente por dioceses, paróquias e comunidades organizadas hierarquicamente. Em termos de tecnologia, este estudo abordou:

- **Arquitetura de Microsserviços:** Uma abordagem ágil que permite criar e manter pequenos serviços independentes para o *backend*, facilitando a escalabilidade e a manutenção de sistemas complexos (Microsoft Docs, 2023).
- **Banco de Dados Relacional:** Optou-se pelo Microsoft SQL Server Express, devido à sua popularidade e integração com o *Integrated Development Environment* (IDE) Visual Studio, escolhido para gerenciar o banco de dados do projeto (Microsoft Docs, 2024).
- **API:** As APIs permitem a comunicação entre diferentes sistemas, como o exemplo do Google Maps para traçar rotas em aplicativos de *delivery* (Jacobson, D., Brail, G., Woods, D., 2012). A API do projeto, "Minha Missa Core", foi projetada para ser consumida pela aplicação web "Minha Missa React App", facilitando a divulgação dos eventos.

3. Percurso Metodológico

O estudo utilizou uma abordagem qualitativa com entrevistas semiestruturadas realizadas com secretários de paróquias e líderes religiosos para entender a atual forma de divulgação de eventos. Também foi conduzida uma pesquisa de opinião sobre a aceitação de uma solução web para centralizar as publicações de eventos.

O desenvolvimento da API seguiu a metodologia ágil Scrum (Scrum Guides, 2020), com *sprints* de 2 a 3 semanas e reuniões periódicas. A equipe de *backend* ficou responsável pela implementação da API, utilizando o banco de dados SQL Server e o *framework* Entity Framework Core para mapeamento objeto-relacional. O Swagger foi utilizado para documentar a API e facilitar seu uso por outros desenvolvedores.

Na modelagem do sistema, foi utilizado o diagrama de classes na *Unified Modeling Language* (UML) com o auxílio da ferramenta *Computer-Aided Software Engineering (*CASE) Lucidchart.

4. Resultados e Discussões

No estudo de entrevistas semiestruturadas realizado com secretários de paróquias e entrevistas com fiéis e padres, identificou-se que os eventos são divulgados principalmente pelas redes sociais e presencialmente nas missas. Destaca-se que a ideia de centralizar essas informações em um único website foi bem recebida pelos entrevistados.

O desenvolvimento da API seguiu o *framework* Scrum, com *sprints* periódicas, reuniões de planejamento, retrospectivas e revisões para avaliar o progresso. Para organização das tarefas, a equipe utilizou o quadro Kanban no GitHub, o que melhorou a eficiência e a clareza no trabalho.

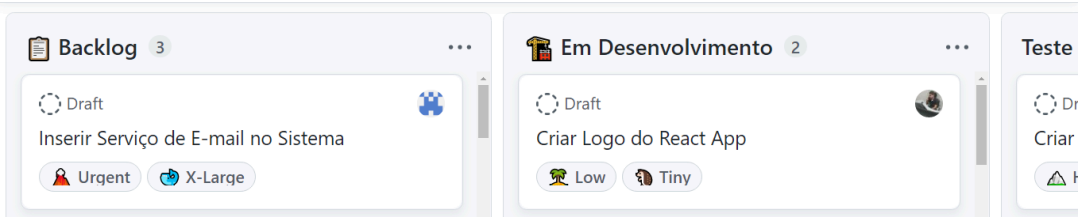

Figura 1. Quadro Kanban utilizado no desenvolvimento do projeto

A Figura 1 ilustra um recorte do quadro utilizado pela equipe de desenvolvimento, em que podem ser observados cartões com tarefas separadas por seções progressivas (“*Backlog”*, “Em desenvolvimento” e “Teste”). No quadro completo, há mais algumas seções além das apresentadas, tais como: “Lista de desejos”, “Em revisão”, “Precisa de atualização” e “Feito”.

Durante a execução das tarefas, os desenvolvedores moviam os cartões entre seções até a sua conclusão ("Feito"), permitindo à equipe avançar no projeto. Algumas tarefas não prioritárias para o MVP da API não foram realizadas dentro do prazo estipulado pelo professor orientador.

Para o desenvolvimento da API, foi criada uma organização no GitHub para gerenciar repositórios e versões do projeto de forma colaborativa. Ramificações (*branches*) foram utilizadas para que os colaboradores pudessem trabalhar em melhorias sem interferir no progresso principal. Após cada implementação, o líder técnico revisava o código e, quando necessário, solicitava ajustes.

O desenvolvimento começou com uma API genérica em ASP.NET Core, que realizava operações de Criar, Deletar, Atualizar e Excluir (CRUD) com base em uma REST API, integrada a uma base de dados genérica e testada com a interface Swagger. A validação foi bem-sucedida, permitindo a modelagem dos domínios e o uso do Entity Framework Core para gerar *scripts* e criar as primeiras tabelas e relacionamentos do banco de dados.

Assim, realizou-se a modelagem das classes para definir os modelos dos objetos da aplicação, seguindo o princípio de *model first* (Microsoft Docs, 2023). Essa metodologia de desenvolvimento aborda a construção dos modelos primeiro, para que depois seja criada a base de dados do projeto.

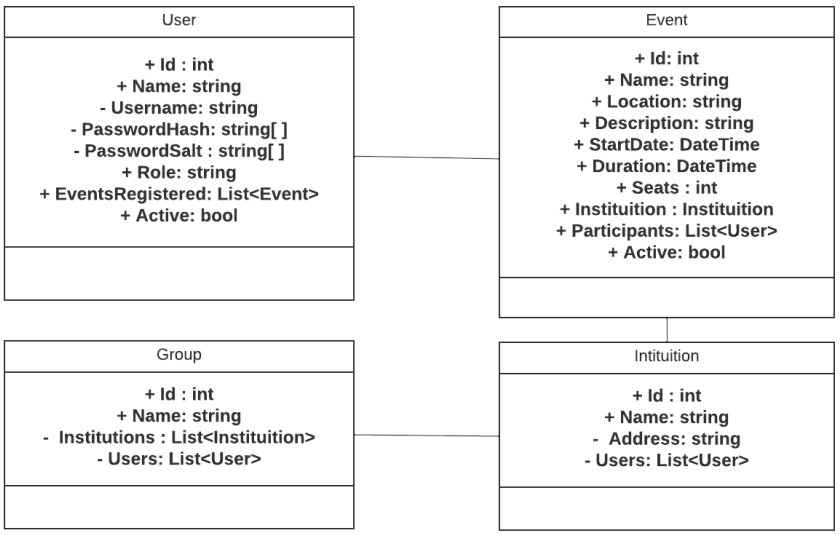

Figura 2. Diagrama de classes do software

Na Figura 2, é apresentado o diagrama de classes baseado na UML composto por 4 classes (*users, events, group* e *intuitions*), cada uma com seus atributos. Com base neste diagrama, foram criados os modelos *users, events, group* e *intuitions*, definindo assim os atributos e métodos de cada objeto em suas respectivas classes.

Em seguida, executou-se um comando de atualização da base de dados, gerando um arquivo de migrações. Esses arquivos executaram a implementação automática dos modelos diretamente no banco de dados.

Com os modelos da API definidos, criou-se os *controllers* dos modelos conforme a demanda da aplicação web "Minha Missa React App" para que fosse possível a utilização dos recursos.

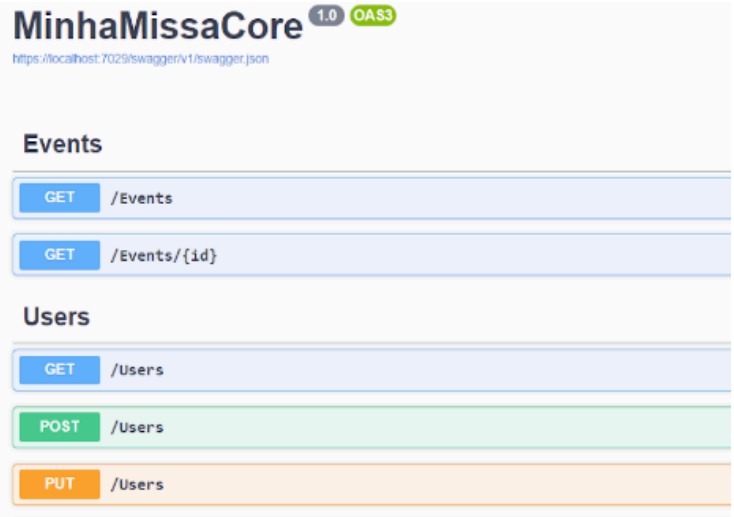

Figura 3. Recorte da tela da interface da API que mostra os *controllers* dos domínios Events e Users

A Figura 3 ilustra a documentação da API, em que é possível visualizar os domínios e operações disponíveis. Por meio da interface, ao selecionar os itens, é

possível interagir com a aplicação, podendo fazer consultas, inserções, atualizações e até mesmo deletar os dados.

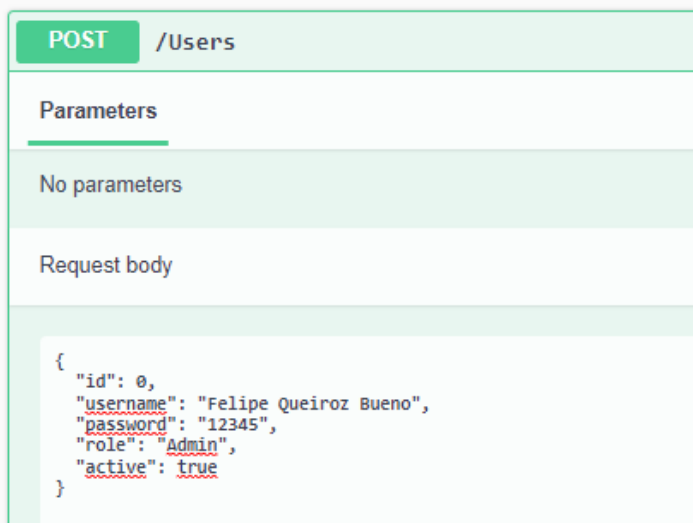

Figura 4. Realizando a inserção de um usuário na aplicação

A Figura 4 mostra um exemplo de como inserir um usuário na base de dados por meio da operação *post,* obedecendo os conceitos de uma API. A inserção é feita por meio de um objeto em Notação de objeto JavaScript (JSON do inglês *JavaScript Object Notation*). Fazer uso dele facilitou a integração com a aplicação "Minha Missa React App", que utilizou o *framework* ReactJs e tem como base a linguagem Javascript.

Ainda na tela apresentada na Figura 4, o desenvolvedor pode editar algumas propriedades para testar como: "username", "password", "role" e "active" e executar a operação. O resultado é a inserção na base de dados.

Os testes foram realizados a partir da interface do Swagger. Eles consistiram em realizar as operações de CRUD no objeto JSON. Como os resultados dessas operações foram de acordo com o esperado, então a API foi validada.

O que esperava-se é que usando: o *post*, a API enviasse esses novos dados para o banco de dados; o *get*, alguns dados do banco fossem retornados; o *put,* alguns dados fossem atualizados no banco; e o *delete,* alguns dados fossem excluídos do banco. E também que ela atendesse às igrejas católicas de Rondonópolis-MT no que tange a publicação centralizada de eventos.

5. Considerações Finais

Este artigo apresentou um projeto conduzido para desenvolver uma solução tecnológica de divulgação de eventos católicos em Rondonópolis-MT. O contexto, os objetivos e a fundamentação teórica foram abordados, destacando o uso de tecnologias e ferramentas no desenvolvimento da API e da base de dados.

Espera-se que a API "Minha Missa Core" seja uma solução eficiente para a divulgação de eventos católicos em Rondonópolis-MT, contribuindo com a comunidade local. Embora algumas funcionalidades, como a autenticação por JSON Web Token

(JWT) e o serviço de e-mail, não tenham sido implementadas, o projeto cumpriu seus principais objetivos e pode ser expandido em trabalhos futuros.

Essas são as principais conclusões do projeto, que mostrou a viabilidade de uma solução tecnológica para melhorar a comunicação entre as paróquias e seus fiéis, aumentando a acessibilidade e a eficiência na divulgação de eventos.

6. Referências

IBGE. Instituto Brasileiro de Geografia e Estatística. “Panorama da cidade de Rondonópolis-MT - População residente por religião. Censo de 2010”. Disponível em: https://cidades.ibge.gov.br/brasil/mt/rondonopolis/panorama. Acesso em: 18 out. 2022.

Canção Nova. “Saiba o que é província eclesiástica”. Jan. 2014. Disponível em: https://noticias.cancaonova.com/brasil/saiba-o-que-e-provincia-eclesiastica/. Acesso em: 10 out. 2022.

Microsoft Docs. “Microservices architecture”. 2023. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/architect-microservice-container-applications/microservices-architecture. Acesso em: 26 set. 2024.

De La Torre, Cesar; Wagner, Bill; Rousos, Mike. “.NET Microservices: Architecture for Containerized .NET Applications”. 6. ed. Microsoft Developer Division, .NET and Visual Studio product teams. Redmond, Washington, 2022.

Heuser, Carlos. “Projeto de banco de dados”. 6. ed. Porto Alegre - RS: Artmed, 2009.

DB-ENGINES. “DB-Engines Ranking”. 2024. Disponível em: https://db-engines.com/en/ranking. Acesso em: 26 set. 2024.

Jacobson, D.; Brail, G.; Woods, D. “APIs: A Strategy Guide”. 1. ed. Sebatopol, USA: O'Reilly, 2012. Disponível em: https://books.google.com.br/books?id=om5tNwKW4xkC&printsec=frontcover&redir_esc=y#v=onepage&q&f=true. Acesso em: 10 out. 2022.

Microsoft Docs. “Creating a simple data-driven CRUD microservice”. 2022. Disponível em: https://learn.microsoft.com/pt-br/dotnet/architecture/microservices/multi-container-microservice-net-applications/data-driven-crud-microservice. Acesso em: 17 ago. 2022.

Scrum Guides. “The Scrum Guide”. 2020. Disponível em: https://scrumguides.org/scrum-guide.html. Acesso em: 26 set. 2024.

Microsoft Docs. Model First. 2023. Disponível em: https://learn.microsoft.com/pt-br/ef/ef6/modeling/designer/workflows/model-first. Acesso em: 26 set. 2024.