MINISTÉRIO DA EDUCAÇÃO
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA
CAMPUS VILHENA
PÓS-GRADUAÇÃO LATO SENSU EM DESENVOLVIMENTO WEB

DESENVOLVIMENTO DE PLATAFORMA PARA MUSEU DE IMAGEM DE VILHENA

ELEMAR LEONEL BALDUINO OLIVEIRA
PEDRO LUIS FERRONATO BARRETO
PAULO DE PAULA
RAFAEL ALMEIDA LARA

VILHENA

2025

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE RONDÔNIA
CAMPUS *VILHENA*

DESENVOLVIMENTO DE PLATAFORMA PARA MUSEU DE IMAGEM DE VILHENA

ELEMAR LEONEL BALDUINO OLIVEIRA
PEDRO LUIS FERRONATO BARRETO
PAULO DE PAULA
RAFAEL ALMEIDA LARA

Produto apresentado ao Instituto Federal de Educação, Ciência e Tecnologia de Rondônia, como requisito avaliativo para conclusão do curso de Pós-graduação em desenvolvimento WEB, sob a orientação do Professor Gilberto Pereira da Silva.

VILHENA

2025

Ficha catalográfica elaborada pelo Sistema Gerador de Ficha Catalográfica do IFRO.

Desenvolvimento de plataforma para museu de imagem de Vilhena / Elemar Leonel Balduino Oliveira ... [et al] / Vilhena, 2025.

11f. : il.

Orientador(a):Prof. Me.Gilberto Pereira da Silva

Trabalho de Conclusão de Curso (Pós-Graduação Lato Sensu em Desenvolvimento Web) – Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO, Vilhena, 2025.

1.Barreto, Pedro Luis Ferronato 2.Paula, Paulo de 3. Lara, Rafael Almeida I. Silva, Gilberto Pereira da (orient.) II. Instituto Federal de Educação, Ciência e Tecnologia de Rondônia - IFRO. III. Título.

Bibliotecário(a) Responsável: Rosilene Maria do Couto Marques, CRB-11/32

DESENVOLVIMENTO DE PLATAFORMA PARA MUSEU DE IMAGEM DE VILHENA

RESUMO: O acesso ao conteúdo histórico e cultural de uma localidade pode ser afetado por diversos fatores, como falta de registro, exposição e local apropriado. A tecnologia, por sua vez, proporciona a conexão e comunicação entre pontos, fazendo surgir assim a oportunidade de atender a necessidade de registrar e expor em meio digital os registros de imagens históricas do estado de Rondônia, principalmente da cidade de Vilhena. Esse projeto busca unir a conexão e exposição proporcionada pela tecnologia com a importância cultural dos registros históricos.

ABSTRACT: Access to the historical and cultural content of a locality can be hindered by several factors such as insuflcient documentation, limited exposure, inadequate preservation, and various other constraints. Technology, on the other hand, serves as a bridge connecting different points, thus offering the opportunity to digitally capture and showcase historical images from the state of Rondônia, particularly from the city of Vilhena. This project aims to merge technological connectivity and exhibition capabilities to preserve and highlight the cultural significance embedded within historical records.

1. INTRODUÇÃO

O projeto se propõe a construir um orquestrador de sistemas, sendo a primeira parte um ambiente de visualização e consulta das imagens e a segunda parte, uma plataforma para gerenciamento de requisições HTTP (*Hypertext Transfer Protocol*) responsável por registrar metadados das imagens para as consultas, enviar requisições de tratamento das imagens em serviço de redimensionamento e por fim, registrá-las de forma consistente no sistema de arquivo.

O desenvolvimento de plataformas por uma equipe pode se tornar complexo na falta de processos e metodologias bem definidas, para o desenvolvimento da plataforma de registro de imagens, a equipe optou por organizar os requisitos e entregas por meio do método Kanban e também dividir a equipe em frentes de desenvolvimento, a fim de simplificar o processo de versionamento de código e paralelizar a produção de valor.

A produção de um sistema orquestrador proporcionou a visibilidade de prós e contras quanto à metodologia, conectar tecnologias distintas e padrões de comunicação distintas pode adicionar dificuldade não vista em sistemas monolíticos, porém, ficou nítido o saldo positivo da decisão, pois, quando se leva em consideração os pontos positivos de redução cognitiva necessária para cada entrega de valor, possibilidade de utilização de tecnologias eficientes para cada cenário e funcionalidade, bem como a possibilidade de construir um sistema robusto com equipe pequena em paralelo, sobrepuseram o atrito de

comunicação dos sistemas, gerando os primeiros passos para ampliar o contato da comunidade com o material histórico da cidade.

2. PROCESSO DE DESENVOLVIMENTO

A produção de soluções tecnológicas, quando feita por uma equipe com horários de disponibilidade diversos, pode encontrar impedimentos para desenvolvimento do produto, nós, a equipe, somos exemplo da situação citada acima, múltiplos desenvolvedores com horários incompatíveis, o que afeta diretamente reuniões e alinhamentos, porém, entram em cena as metodologias ágeis, tendo como boas opções, por experiências anteriores dos membros da equipe, literatura disponível e adesão do mercado as metodologias: *Scrum* e *Kanban*, sendo a primeira orientada a cerimônias buscando a adaptação constante do time às alterações solicitadas e impedimentos encontrados, buscando entregar sempre o valor desejado ou solicitado pelo cliente, Soares (2004), já a segunda se tratando de uma metodologia de cartões puxados, desenvolvida no Japão, principalmente na companhia Toyota, busca promover a fácil visualização do estado da produção, gargalos e hiatos, Aguiar e Peinado (2007).

Dado o contexto da equipe onde conflitos de agenda eram frequentes, afetando a possibilidade do desenvolvimento das cerimônias, necessitando constantemente adaptá-las, reduzindo valor e eficiência da metodologia *Scrum*, o time então optou por desenvolver o projeto baseando-nos na metodologia *Kanban* visto que mesmo em caso de longos períodos de tempo sem encontros síncronos da equipe o produtos seria menos afetado, vista que, quando disponível para produzir, um membro do grupo poderia analisar a tabela, reconhecer o estado atual e então tomar para si atividades dispostas na coluna de tarefas aguardando para início do desenvolvimento.

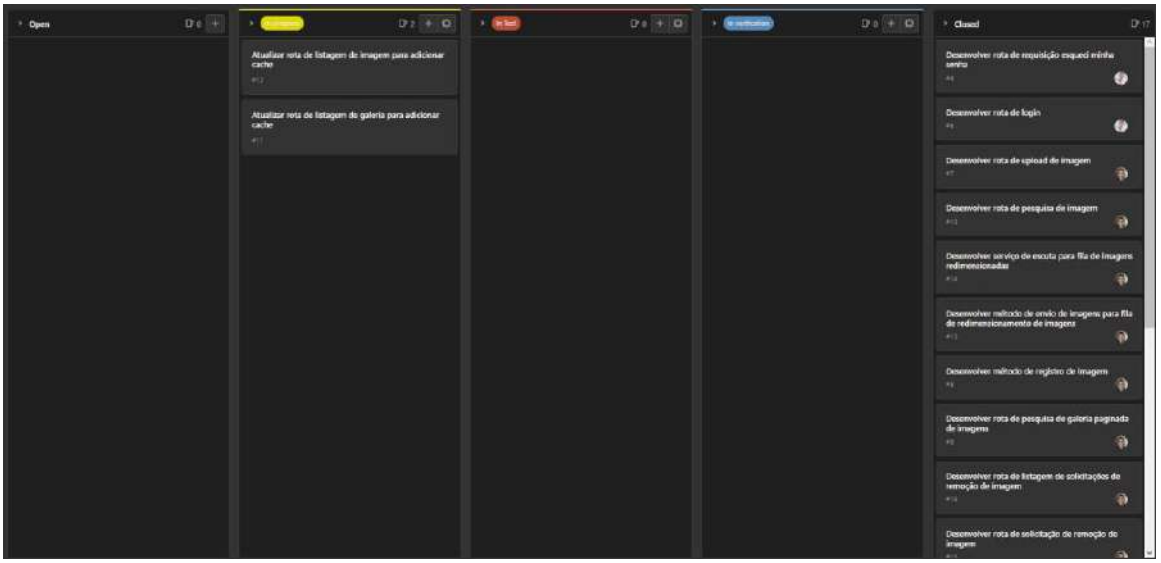

Figura 1. Captura do quadro durante finalização do projeto

3. PRODUTO

3.1. Arquitetura da aplicação

Como citado anteriormente o projeto buscou produzir um sistema orquestrador de requisições a fim de registrar imagens e seus metadados, seguindo conceitos de microsserviços, sua responsabilidade ficaria segmentada da seguinte forma, para a aplicação *front-end*, deveria receber arquivos de imagens e seus metadados e então enviá-los para a *API* (*Application Programming Interface*) no *back-end*, essa por sua vez teria de receber requisições *HTTP* (*Hypertext Transfer Protocol*), validar o conteúdo, separar a requisição em arquivo, enviado para o serviço externo de redimensionamento de imagens e metadados, enviados para banco de dados gerenciado pela própria API, esses servindo de base para exibição, consulta e recuperação das imagens que seriam exibidas novamente na aplicação *front-end*, ficou definido também que a aplicação *back-end* ficaria responsável por escutar fila de mensageria de resultados de redimensionamento de imagens, recebendo tais mensagens para registrar, novamente em serviço externo, os diferentes formatos da imagem em sistema de arquivos.

A utilização de microsserviços foi essencial para a redução de carga cognitiva necessária por parte da equipe de desenvolvimento nos processos de planejamento e desenvolvimento da solução, podendo assim focarmos nossos esforços no orquestramento das requisições e serviços, abrindo espaço para que os serviços consumidos fossem substituídos, tanto para soluções em menor prazo ou então para adaptação das necessidades tecnológicas do projeto, Justino (2018).

Ademais, ao não incluirmos serviços que aqui chamamos de externos abrimos a oportunidade, levando em consideração também o ambiente da instituição em que o projeto foi desenvolvido, de aprofundarmos tais produtos em demais projetos, sendo assim, autenticação, sistemas de arquivos e redimensionamento de imagens puderam ser explorados com sua devida atenção por diferentes equipes, com diversas tecnologias, também foi possível dedicar esforço do time para serviços periféricos em momentos externos ao presente projeto.

Para comunicação entre os serviços o time optou por adicionar o elemento de filas de mensageria, o caso de uso neste projeto foi para tornar assíncrona a comunicação entre a API *back-end* e o serviço de redimensionamento de imagens, sendo assim metadados e formato poderiam ser registrados e consumidos antes mesmo da produção das diferentes dimensões do arquivo serem produzidos, ademais, em caso de falha temporária a submissão de novos registros seria apenas parcialmente afetada, Marcos (2016).

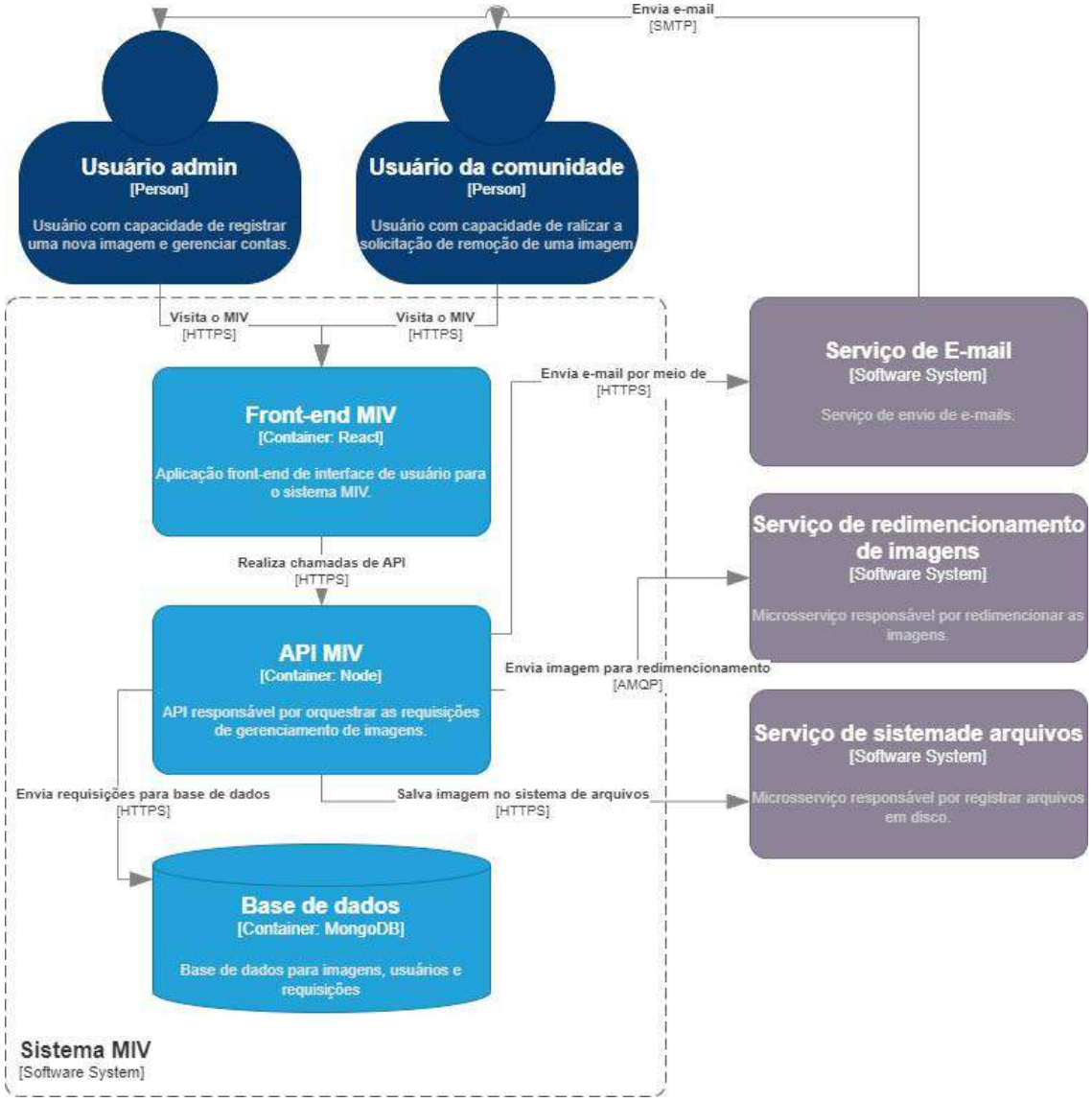

Figura 2. Diagrama de Container da arquitetura do projeto

Na figura 1 acima podemos analisar a arquitetura do produto, o sistema pode ser utilizado por dois arquétipos de usuários, usuário administrador, capaz de consumir e registrar imagens e usuário da comunidade, capaz apenas de visualizar imagens, o acesso de ambos sempre será realizado apenas pela aplicação *front-end.*

A aplicação *front-end* então coleta os dados enviados pelo usuário administrador e os envia para o *back-end* que adiciona os arquivos em uma fila AMQP (*Advanced Message Queuing Protocol*) com os tamanhos de imagem desejados, os metadados são registrados na base de dados MongoDB, nesse momento a imagem já pode ser parcialmente consumida, o processo de escuta fica responsável por receber os resultados do redimensionamento e enviá-los para o serviço de registro em sistema de arquivos, nesse momento uma imagem está pronta para ser completamente consumida.

3.2. Modelo de dados

O produto gerencia atualmente três modelos de dados, sendo dois deles, imagens e solicitação de remoção, centrais e essenciais para o atendimento adequado das regras de negócio levantadas, já o modelo de usuário é um desvio temporário de responsabilidade, este fica responsável pela autenticação e autorização dos usuários na aplicação, tanto *back-end* quanto *front-end*, a equipe pretende futuramente remover essa funcionalidade segmentando-a para novo serviço externo.

Tabela 1. Modelo de dados de imagens

Campo	Tipo	Objetivo do campo
id	String	Busca identificar uma imagem, responsável por apontar para registros dos metadados e recuperação em sistema de arquivos.
description	String	Busca registrar uma descrição legível a humanos, usada para apresentação.
sizes	Matriz numérica	Busca registrar os tamanhos registrados para uma imagem.
tags	Lista de Strings	Busca registrar marcadores para futura consulta da imagem.

Tabela 2. Modelo de dados de requisição de remoção

Campo	Tipo	Objetivo do campo
id	String	Busca identificar uma requisição de remoção de imagens
reason	String	Busca registrar o motivo da solicitação.
requesterName	String	Busca registrar o nome do solicitante.
email	String	Busca registrar o email do solicitante.
resolveMessage	String	Busca registrar o parecer do avaliador.
status	String	Busca registrar o estado de processamento atual da requisição.

imageId	String	Busca registrar a imagem a ser removida.
userId	String	Busca registrar o avaliador da requisição.

Tabela 3. Modelo de dados de usuários

Campo	Tipo	Objetivo do campo
id	String	Busca identificar um usuário.
admin	Boolean	Busca registrar o nível de autorização de um usuário.
email	String	Busca registrar o email de um usuário.
name	String	Busca registrar o nome do usuário.
password	String	Busca registrar o hash para autenticação do usuário.

3.3. Documentação da API

Atualmente a API *back-end* disponibiliza três grupos de rotas para gerenciamento dos modelos de dados apresentados anteriormente, e como o modelo de dados, é pretendido futuramente remover a responsabilidade de autorização e autenticação do produto. Os modelos de dados são recebidos inteira ou parcialmente pelas rotas, a produção dos identificadores sempre fica como responsabilidade das regras de negócio na aplicação *back-end*.

Tabela 4. Grupo de rotas de imagens

Rota	Requer administrador	Função da rota
POST /image/	Sim	Rota de registro de nova imagem, é necessário o envio do modelo de dados parcial.
GET /image/:id	Não	Rota para consulta dos metadados de uma imagem.
GET /image/stream/:id	Não	Rota para recuperação das imagens, possibilita requisição de tamanho específico.
GET /image/	Não	Rota para consulta por tags, é necessário o envio de lista de tags para consulta.

Tabela 5. Grupo de rotas de requisição de remoção de imagem

Rota	Requer administrador	Função da rota
POST /remove-request/	Não	Rota de registro de nova requisição de remoção de imagem, é necessário o envio de modelo de dados parcial.
GET /remove-request/	Não	Rota de consulta de requisições de remoção por status de processamento.
PATCH /remove-request/	Sim	Rota de parecer de uma requisição de remoção, requer o envio de identificador e valor booleano e em caso de recusa deve informar também o motivo.

Tabela 6. Grupo de rotas de usuário

Rota	Requer administrador	Função da rota
POST /user/	Sim	Rota de registro de usuário, é necessário o envio de modelo de dados parcial.
PUT /user/:id	Não	Rota de atualização de usuário, é necessário o envio de modelo de dados parcial.
POST /auth/signin	Não	Rota de autenticação de usuário.
POST /auth/forgotPassword	Não	Rota de recuperação de senha de usuário.
POST /auth/resetPassword	Não	Rota de atualização de senha de usuário.

3.4. Capturas de tela

Figura 3. Tela Inicial de Pesquisa do Usuário

Na figura acima, temos o layout da tela inicial, onde o usuário poderá fazer a pesquisa das imagens que forem registradas pelo museu. O usuário pesquisa por tags, podendo fazer uso de pré-definidas. Depois de realizar a consulta, é redirecionado para uma página, onde mostrará os resultados.

Figura 4. Tela de Resultados da Pesquisa

Já nesta figura, temos a página de pesquisa do usuário, onde são retornados os dados, com a respectiva imagem e título da obra cadastrada. Há também uma funcionalidade de paginação para melhor experiência do usuário ao navegar dentro do site.

4. CONCLUSÃO

O desenvolvimento do presente produto se mostrou um bom desafio para a equipe, para muitos do grupo se tratava do primeiro contato prático com conceitos de mensageria e microsserviços, ademais, passar pelo processo de organização de horários, elencamento e divisão de tarefas, análise e implantação de metodologias ágeis se provaram experiências enriquecedoras para a equipe, reforçando conceitos já dominados e adicionando novos e essenciais conhecimentos para o nosso leque de habilidades profissionais.

REFERENCIAS

DOS SANTOS SOARES, Michel. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação**, v. 3, n. 1, 2004.

PEINADO, JURANDIR; AGUIAR, G. Compreendendo o Kanban: um ensino interativo ilustrado. **Revista DaVinci. Curitiba–PR**, v. 4, n. 1, p. 133-146, 2007.

Arruda, F. J. de C. Martins, D. (2020) "Análise Comparativa entre sistemas de mensageria: Apache Kafka vs RabbitMQ", In: Seminários de Trabalhos de Conclusão de Curso do Bacharelado em Sistemas de Informação. Brasil.

Marcos, P. B. (2016) "Plataforma de Monitorização de Recursos num Sistema Message Oriented Middleware", In: Faculdade de Engenharia da Universidade do Porto.

Araújo, R. O. (2021) "Uma revisão teórica sobre a arquitetura de microsserviços", In: Universidade Tecnológica Federal do Paraná. Brasil.

Justino, Y. de L. (2018) "Do monolito aos microsserviços: um relato de migração de sistemas legados da Secretaria de Estado da Tributação do Rio Grande do Norte". In: Universidade Federal do Rio Grande do Norte. Brasil.