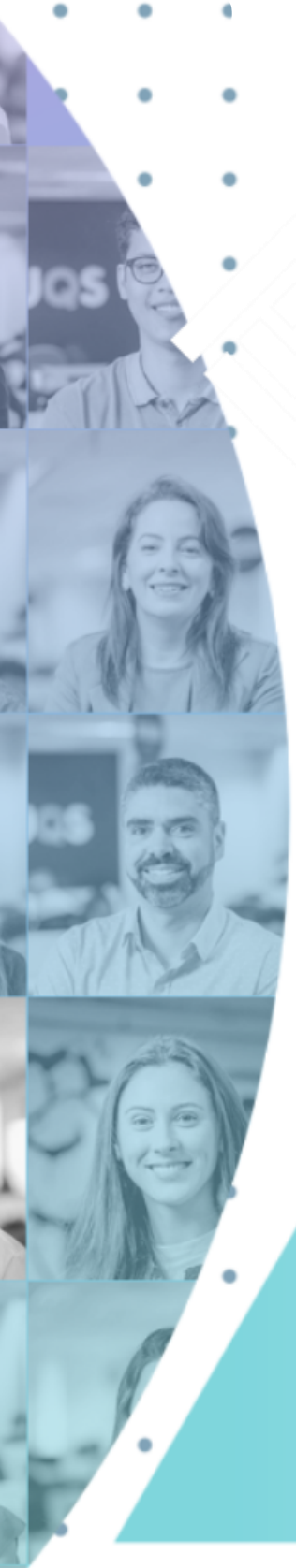
YDUQS



DOCUMENTO DE ESPECIFICAÇÃO

SUMÁRIO

Sumário

SUMÁRIO 2
1. OBJETIVO 3
2. ESCOPO 3
3. Funcionalidades 3
3.1 Cadastro de Canal de Notificação 3
3.2 Envio de Notificações 4
3.3 Histórico de Notificações 4
3.4 Status de Envio em Tempo Real 5
4. Arquitetura do Sistema 5
4.1 Backend 5
4.2 Frontend 6
4.3 Banco de Dados 6

1. OBJETIVO

Este documento tem como objetivo descrever as especificações funcionais e técnicas da Central de Notificações, uma aplicação que permite o envio de notificações em tempo real por múltiplos canais, como Email, SMS, e Push. O sistema foi projetado para ser extensível, permitindo a fácil adição de novos canais de notificação sem impactar o código existente.

2. ESCOPO

A solução abrange:

- Cadastro de Canais: Adicionar, ativar ou desativar canais de notificação.
- Envio de Notificações: Permite enviar notificações através de canais ativos selecionados.
- Status em Tempo Real: Monitora e exibe o status do envio das notificações.
- Histórico: Exibe as últimas 20 notificações enviadas.

3. Funcionalidades

3.1 Cadastro de Canal de Notificação

- Exibe os canais disponíveis: Email, SMS, Push.
- Permite ativar/desativar cada canal.
- Validação de status de canal e controle de permissões.

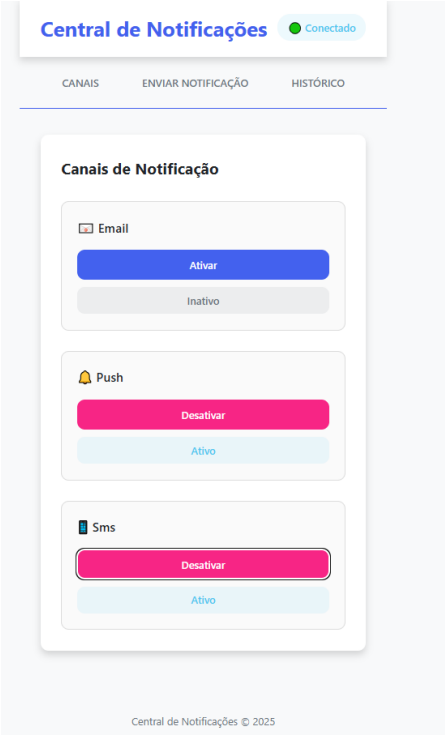


Figura 1: Canal de Notificação

3.2 Envio de Notificações

- Formulário de envio de assunto e mensagem.
- Seleção de canais para envio (checkbox).
- Backend envia as notificações para os canais ativos.

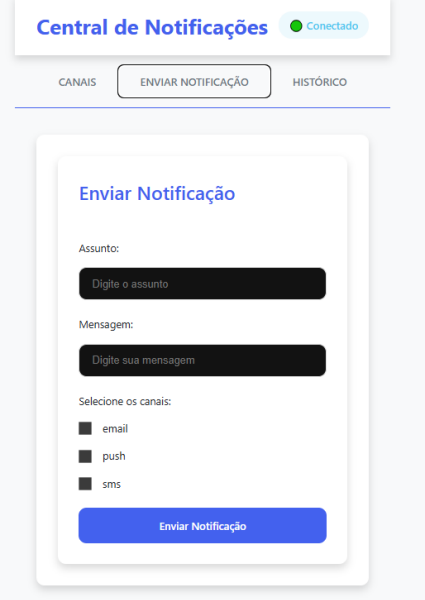


Figura 2: Envio de Notificações

3.3 Histórico de Notificações

Exibe as últimas 20 notificações enviadas, com assunto, mensagem, canais e data.

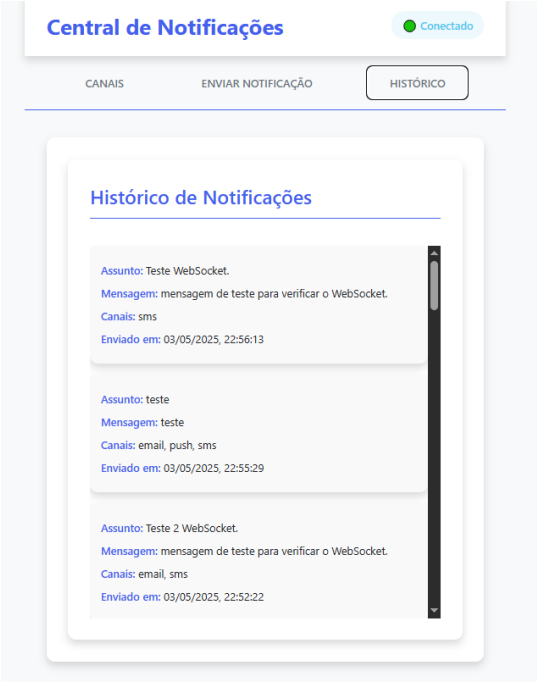


Figura 3: Histórico de Notificações

3.4 Status de Envio em Tempo Real

- Exibição do status das notificações utilizando WebSocket.
- Atualização em tempo real dos status, como Enviado, Falhou.

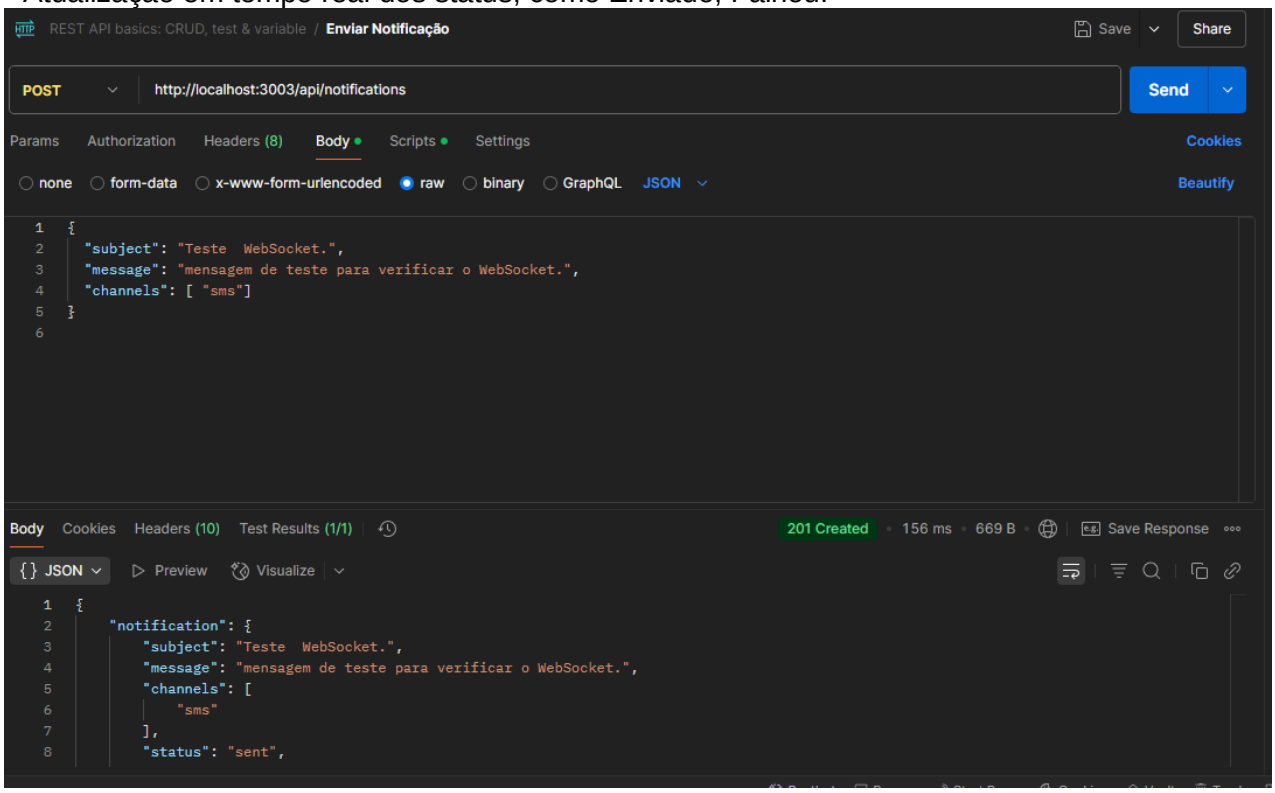


Figura 4: Websocket via postman

4. Arquitetura do Sistema

4.1 **Backend**

1. node_modules: Diretório contendo todas as dependências necessárias para o backend.
2. src: Contém o código principal do backend, estruturado em módulos:
    - config: Arquivos de configuração do sistema, como variáveis de ambiente e configurações de banco de dados.
    - core: Camada central do sistema, onde reside a lógica principal do backend.
    - modules: Dividido em submódulos que representam diferentes funcionalidades do sistema, como canais e controladores:
        - channels: Configuração e gerenciamento dos canais de notificação (email, SMS, etc.).
        - controllers: Controladores que recebem as requisições e interagem com a lógica de negócios.
        - factories: Criação dinâmica de instâncias de envio de notificações baseadas no tipo de canal.
        - middlewares: Funções intermediárias para validação de dados e autenticação.
        - models: Definição dos modelos de dados (provavelmente para o banco de dados).

- routes: Definição das rotas da API REST.
- strategies: Estratégias de envio de notificações por diferentes canais.

- seed: Para popular o banco de dados com dados iniciais.
- utils: Funções auxiliares e utilitárias.

3. .env: Arquivo de configuração de variáveis de ambiente.
4. jest.config.ts / jest.setup.ts: Configurações de testes automatizados utilizando o Jest.
5. package.json e package-lock.json: Gerenciam as dependências do projeto.
6. tsconfig.json: Configuração do TypeScript para o backend.

4.2 Frontend

1. node_modules: Diretório de dependências do frontend.
2. public: Contém arquivos públicos acessíveis diretamente, como index.html.
3. src: Contém o código fonte do frontend:
   - assets: Arquivos estáticos como imagens e ícones.
   - components: Componentes reutilizáveis para a interface, como formulários e listas de notificações.
   - context: Context API para gerenciar o estado global, como o status dos canais e o histórico de notificações.
   - hooks: Hooks personalizados para abstrair lógicas reutilizáveis, como manipulação de canais e notificações.
   - core: Lógica central do frontend.
4. .env: Arquivo de variáveis de ambiente específicas para o frontend.
5. package.json e package-lock.json: Gerenciam as dependências do frontend.
6. tsconfig.json: Configuração do TypeScript para o frontend.

4.3 Banco de Dados

O sistema utiliza MongoDB como banco de dados principal, com suporte para:

- Implementação local (MongoDB Community)
- Nuvem (MongoDB Atlas)

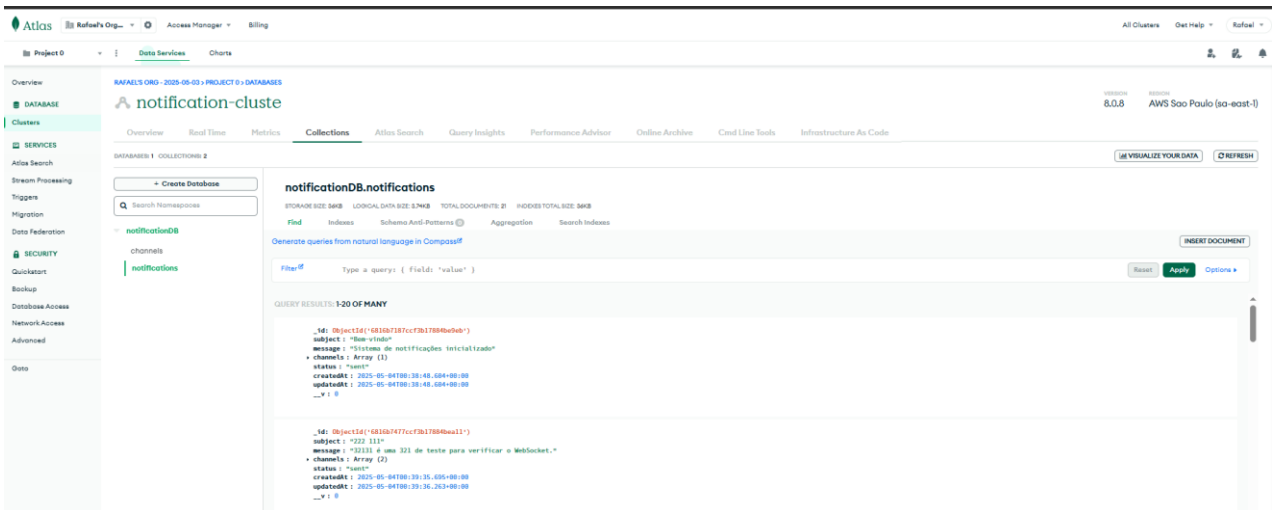


Figura 5: Banco MongoDB Atlas configurado com AWS