Documentação Desafio Campanhas

Descrição do Projeto:

O Campaign Challenge é um projeto completo para gerenciar campanhas, composto por um backend e um frontend. O backend fornece uma API RESTful para gerenciar as campanhas, enquanto o frontend fornece uma interface de usuário para interagir com essa API. Este projeto visa facilitar a criação, edição, exclusão e visualização de campanhas, oferecendo uma interface amigável e intuitiva.

Tecnologias Utilizadas:

Backend:

- Node.js: Ambiente de execução JavaScript server-side.
- Express: Framework web para Node.js, usado para construir a API RESTful.
- Joi: Biblioteca de validação de dados.
- Jest: Framework de testes para JavaScript.
- Next**: Mesmo sendo uma ferramenta do front-end foi utilizada de referência para criação das API's.
- Postman: Ferramenta para teste das API's.
- Swagger: Ferramenta para documentação das API's.

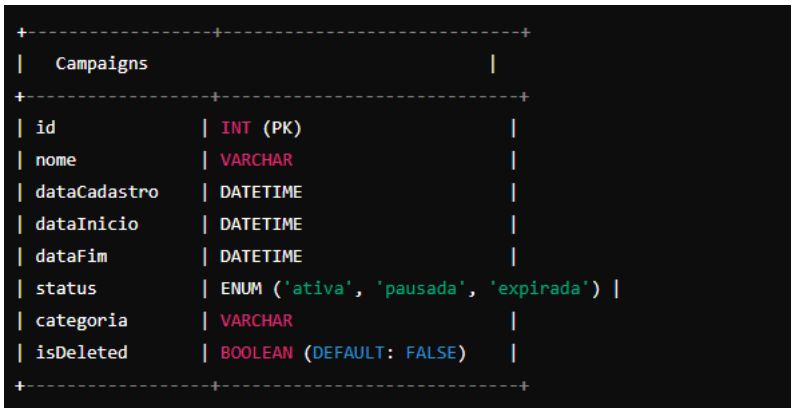
Frontend:

- Next.js: Framework React para desenvolvimento de aplicações web.
- React: Biblioteca JavaScript para construção de interfaces de usuário.
- Axios: Cliente HTTP para fazer requisições à API.
- Material-UI: Biblioteca de componentes React para estilização.
- Jest: Framework de testes para JavaScript.
- React Testing Library: Biblioteca para testar componentes React.
- Git: Ferramenta para controle de versão, também utilizada no backend.
- Visual Studio Code + Dev Tools

Instruções de Compilação e Execução

O arquivo README.md presente nesse mesmo diretório tem o detalhamento para compilação, execução e realização dos testes.

Diagrama Entidade-Relacionamento (ER)



Descrição dos Campos

id: Chave primária, auto-incrementada.

nome: Nome da campanha.

dataCadastro: Data e hora de criação da campanha, definida automaticamente.

dataInicio: Data de início da campanha.

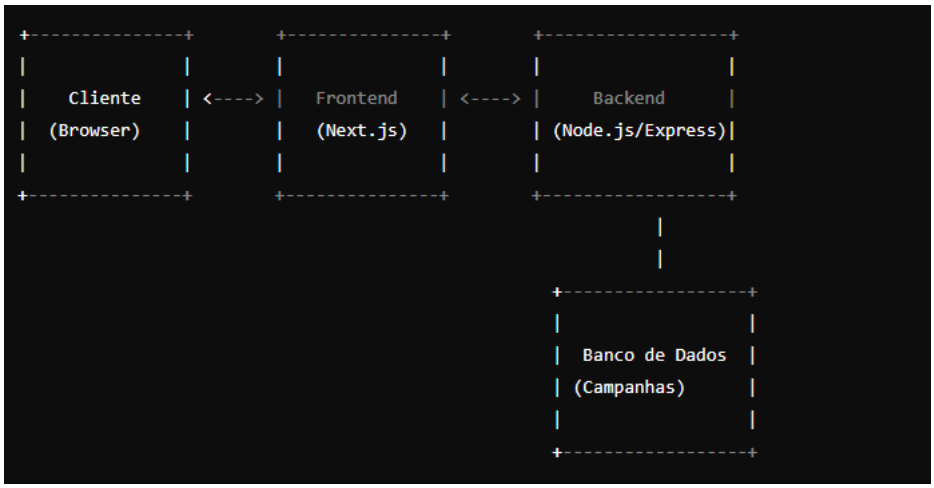
dataFim: Data de término da campanha.

status: Status da campanha, pode ser ativa, pausada ou expirada.

categoria: Categoria da campanha.

isDeleted: Indicador de exclusão lógica, padrão é FALSE.

Diagrama de arquitetura



Elementos do Diagrama

- **Cliente (Browser)**: Representa os usuários que interagem com a aplicação.
- **Frontend (Next.js)**: Lida com a renderização do conteúdo e interações do usuário.
- **Backend (Node.js/Express)**: Gerencia a lógica do negócio, validação e processamento de dados.
- **Banco de Dados**: Armazena os dados das campanhas.

Descrição das principais funcionalidades

Backend

1. Criação de Campanhas:
   - Rota: POST /api/campaigns
   - Descrição: Permite a criação de uma nova campanha com os campos nome, data de início, data de fim, status e categoria.

2. Leitura de Campanhas:
   - Rota: GET /api/campaigns
   - Descrição: Retorna uma lista de todas as campanhas existentes.

3. Leitura de uma Campanha Específica:
    - Rota: GET /api/campaigns/:id
    - Descrição: Retorna os detalhes de uma campanha específica pelo ID.

4. Atualização de Campanhas:
    - Rota: PUT /api/campaigns/:id
    - Descrição: Permite atualizar os detalhes de uma campanha específica pelo ID.

5. Exclusão de Campanhas (Soft Delete):
    - Rota: DELETE /api/campaigns/:id
    - Descrição: Permite a exclusão de uma campanha específica pelo ID, marcando-a como deletada sem removê-la fisicamente do banco de dados.

Frontend

1. Listar todas as campanhas:
    - Página inicial que exibe todas as campanhas em uma tabela ou lista.

2. Criar uma nova campanha:
    - Formulário para criar uma nova campanha com campos para nome, data de início, data de fim, status e categoria.

3. Editar uma campanha existente:
    - Formulário para editar os detalhes de uma campanha existente.

4. Excluir uma campanha:
    - Opção para excluir uma campanha da lista.

5. Exibir detalhes de uma campanha:
    - Página de detalhes que exibe todas as informações de uma campanha específica.

Testes Unitários

Backend

Os testes unitários no backend são realizados usando o Jest e cobrem as funcionalidades principais da API, incluindo:

1. **Testes de Validação e Status da Campanha:**
    - Validação dos dados de entrada ao criar ou atualizar campanhas.
    - Atualização do status das campanhas baseado na data de fim.

2. **Testes de Rotas:**
    - Criação de campanhas (POST /api/campaigns).
    - Leitura de todas as campanhas (GET /api/campaigns).
    - Leitura de uma campanha específica (GET /api/campaigns/:id).
    - Atualização de campanhas (PUT /api/campaigns/:id).
    - Exclusão de campanhas (DELETE /api/campaigns/:id).

Frontend

Os testes unitários no frontend são realizados usando o Jest e React Testing Library e cobrem as funcionalidades principais dos componentes React, incluindo:

1. **Testes de Renderização:**
    - Verificação se os componentes principais são renderizados corretamente.
    - Renderização da lista de campanhas na página inicial.
    - Renderização do formulário de criação e edição de campanhas.

2. **Testes de Interação:**
    - Interações do usuário como preencher formulários e clicar em botões.
    - Testes de envio de formulários para criar e editar campanhas.
    - Testes de exclusão de campanhas.

Considerações Finais

Desafios Encontrados

1. Configuração do Ambiente de Desenvolvimento:

   - Configurar corretamente o ambiente de desenvolvimento para garantir compatibilidade entre as diferentes versões de dependências foi um desafio inicial significativo. Especificamente, configurar o Jest e garantir que as transformações de módulos funcionassem corretamente exigiu vários ajustes e testes.

2. Integração Frontend e Backend:

   - Assegurar que o frontend e o backend se comuniquem corretamente, especialmente com o uso de CORS e configuração de proxies, foi um desafio. Garantir que as requisições do frontend fossem direcionadas corretamente para o backend envolveu várias tentativas e configuração precisa.

3. Validação de Dados:

   - Implementar validações robustas tanto no frontend quanto no backend para assegurar que os dados das campanhas fossem consistentes e válidos foi desafiador. Usar bibliotecas como Joi no backend ajudou, mas ainda assim, garantir a integridade dos dados em todos os cenários exigiu cuidado extra.

4. Testes Unitários e de Integração:

   - Configurar e escrever testes abrangentes que cobrissem todas as funcionalidades principais do sistema foi um desafio, especialmente ao garantir que os testes fossem independentes e pudessem ser executados consistentemente em diferentes ambientes.

Lições Aprendidas

1. Importância da Documentação:

   - Ter uma documentação clara e detalhada é essencial para o sucesso de um projeto. Isso não só ajuda na fase de desenvolvimento, mas também é crucial para a manutenção e escalabilidade futuras do sistema.

2. Gestão de Dependências:

   - Gerenciar dependências corretamente e entender as implicações de atualizações e compatibilidades é crucial. Ferramentas como `npm audit` podem ser extremamente úteis para identificar vulnerabilidades e incompatibilidades.

3. Comunicação Clara entre Frontend e Backend:

   - Garantir uma comunicação clara e eficiente entre o frontend e o backend é vital. Configurações corretas de CORS e proxies facilitam essa comunicação e evitam muitos problemas durante o desenvolvimento.

Pontos de Melhoria

1. Otimização de Desempenho:

   - Implementar técnicas de caching e otimização de consultas no backend pode melhorar significativamente o desempenho do sistema, especialmente quando lidar com grandes volumes de dados.

2. UI/UX Melhorada:

   - Embora a interface do usuário tenha sido funcional, há sempre espaço para melhorias em termos de usabilidade e design. Realizar testes de usabilidade com usuários finais pode fornecer insights valiosos para melhorias contínuas.

3. Monitoramento e Logging:

   - Implementar uma solução de monitoramento e logging mais robusta pode ajudar a identificar e resolver problemas mais rapidamente em ambientes de

produção. Ferramentas como ELK Stack (Elasticsearch, Logstash, Kibana) ou serviços de monitoramento em nuvem podem ser considerados.

Conclusão

O projeto Campaign Challenge proporcionou uma experiência rica em aprendizado, desde a configuração inicial até a implementação de funcionalidades e testes. Embora tenha havido desafios, as lições aprendidas e os pontos de melhoria identificados fornecem um caminho claro para aprimorar ainda mais o sistema, tornando-o mais robusto, escalável e amigável para os usuários finais.