Projecto de Registro de Memórias

Nome:Helizandra M. jose Victor _706230435

1. Introdução

Esta documentação descreve a aplicação Front-end desenvolvida para o projeto Registro de Memórias. A aplicação permite que os usuários registrem memórias pessoais, as categorizem, favoritem, editem, excluam, visualizem relatórios e interajam por meio de um chat. O objetivo é fornecer uma interface simples e intuitiva para gerenciar memórias de forma organizada, com suporte a funcionalidades interativas e relatórios visuais.

2. Informações Gerais

- **Nome da aplicação**: Registro de Memórias
- **Objetivo principal**: Permitir que usuários registrem, gerenciem e visualizem memórias pessoais, com opções para categorizar, favoritar, editar, excluir, visualizar relatórios e interagir via chat.
- **Público-alvo**: Usuários que desejam organizar memórias pessoais, como estudantes, profissionais ou qualquer pessoa interessada em registrar momentos importantes.
- **Tecnologias utilizadas**:
    - **Front-end**: HTML, CSS, JavaScript, Bootstrap 5.3.0 (para estilização e modais).
    - **Back-end**: Node.js com Express (para API e gerenciamento de dados).
    - **Armazenamento**: Arquivo JSON (memorias.json) como banco de dados simples.
    - **Outras ferramentas**: Fetch API (para comunicação com o backend), npm (para gerenciamento de dependências).

3. Instalação e Configuração

Pré-requisitos

- **Node.js** (versão 14.x ou superior) e **npm** instalados.
- Um navegador moderno (ex.: Chrome, Firefox).
- Editor de código como Visual Studio Code (opcional, mas recomendado).

Comandos de Instalação

1. Clone ou baixe o projeto para o seu computador.
2. Abra o terminal na pasta do projeto (ex.: C:\Users\Helizandra\Desktop\Projecto-Pagina-Web).
3. Instale as dependências:
4. npm install

Como Iniciar o Projeto

1. No terminal, na pasta do projeto, execute:
2. npm start
3. O servidor será iniciado na porta 3000. Acesse a aplicação em:
4. http://localhost:3000

Configuração de Variáveis de Ambiente

- Não há variáveis de ambiente configuradas no projeto atual. Todas as configurações (como a porta do servidor) estão hardcoded no arquivo server/app.js (porta 3000).

4. Estrutura do Projeto

A estrutura do projeto está organizada da seguinte forma:

```
/Projecto-Pagina-Web
  ├── /public
  │   ├── index.html       (página principal para registro de memórias)
  │   ├── favoritos.html   (página para listar memórias favoritas)
  │   ├── chat.html        (página de chat interativo)
  │   ├── painel.html      (página de relatórios com gráfico de pizza)
  │   ├── sobre.html       (página informativa sobre o projeto)
  │   └── estilo.css       (estilos globais da aplicação)
  ├── /server
  │   ├── app.js           (servidor Node.js com Express)
  │   ├── routes.js        (rotas da API)
  │   └── memorias.json    (arquivo JSON para armazenamento de dados)
```

```
├── package.json          (dependências e scripts do projeto)
└── README.md             (instruções básicas do projeto)
```

- **/public**: Contém os arquivos estáticos (HTML, CSS) servidos pelo servidor.
- **/server**: Contém o backend com Node.js, Express e o arquivo JSON que atua como banco de dados.
- **package.json**: Define as dependências (express, cors, body-parser) e scripts (start).

5. Funcionalidades

A aplicação possui as seguintes funcionalidades principais:

Registro de Memórias (index.html):

- Formulário para registrar memórias com campos para nome, idade, categoria (Família, Lazer, Conquista, Viagens, Trabalho) e descrição.
- Exibe uma lista de memórias registradas com opções para favoritar, editar e excluir.
- Suporte a filtro por categoria.
- Mensagens de feedback (ex.: "Memória registrada com sucesso!") exibidas em modais estilizados.

Gerenciamento de Favoritos (favoritos.html):

- Lista memórias marcadas como favoritas.
- Permite desfavoritar memórias com feedback em modal ("Memória desfavoritada!").

Chat Interativo (chat.html):

- Interface de chat com mensagens automáticas de resposta (ex.: "Oi! Como posso te ajudar?").
- Suporte a mensagens do usuário e respostas automáticas.

Relatórios (painel.html):

- Exibe um gráfico de pizza mostrando a distribuição de memórias favoritas por categoria.
- Usa Chart.js para renderizar o gráfico.

Página Sobre (sobre.html):

- Informações sobre o projeto.

Responsividade:

- Interface adaptada para dispositivos móveis usando Bootstrap e media queries no estilo.css.

- **Integração com API**:

  - Comunicação com o backend Node.js via Fetch API para operações CRUD (criar, ler, atualizar, deletar memórias).
  - Endpoints usados: /api/memories (GET, POST, PUT, DELETE) e /api/memories/:id/favorite (PATCH).

6. Documentação de Código

Comentários no Código:

- Os arquivos HTML e JavaScript contêm comentários explicando seções principais (ex.: "Seleciona o formulário de registro", "Função para carregar as memórias").
- O arquivo estilo.css inclui comentários para separar seções (ex.: /* Estilo dos modais de feedback e edição */).

- **Boas Práticas**:

  - Uso de nomes descritivos para variáveis e funções (ex.: loadMemories, showFeedback).
  - Estrutura modular com funções separadas para cada funcionalidade (ex.: showConfirmDelete, loadFavorites).

- Não há uso de ferramentas como JSDoc, mas o código é autoexplicativo com comentários inline.

7. Testes

- **Tipos de Testes**:
  - Não foram implementados testes automatizados (unitários, de integração ou end-to-end).

Testes Manuais:

- Testes manuais foram realizados para todas as funcionalidades:
  - Registro, edição, exclusão e favoritação de memórias.
  - Filtragem por categoria e listagem de favoritos.
  - Funcionamento do chat e do gráfico de pizza.
  - Responsividade em diferentes dispositivos (desktop, tablet, celular).

Ferramentas Utilizadas:

- Navegador (Chrome) para testes manuais e DevTools para verificar responsividade.

Como Executar Testes:

- Inicie o servidor com npm start e acesse http://localhost:3000. Teste manualmente cada funcionalidade.

8. Hospedagem

Plataforma de Hospedagem: https://desenvolvimento-de-pagina-web-com-hm47.onrender.com

Passos para o Deploy:

1. Crie um repositório no GitHub e envie o projeto:
2. git init
3. git add .

4. git commit -m "Deploy inicial do projeto Registro de Memórias"
5. git branch -M main
6. git remote add origin <URL-do-seu-repositório>
7. git push -u origin main
8. No Render, crie um **Web Service**:
    - Conecte o repositório GitHub.
    - Configure:
        - **Environment**: Node
        - **Build Command**: npm install
        - **Start Command**: npm start
        - **Instance Type**: Free
9. Após o deploy, o Render fornece o link de acesso (ex.: https://<seu-projeto>.onrender.com).

- **Link de Acesso à Aplicação**:
    - O link será gerado após o deploy no Render (ex.: https://registro-memorias.onrender.com). Verifique no painel do Render.

9. Conclusão

A aplicação "Registro de Memórias" foi desenvolvida com sucesso, atendendo aos objetivos de permitir o registro e gerenciamento de memórias de forma intuitiva e visualmente agradável. Durante o desenvolvimento, foram aprendidas lições importantes, como:

- A importância de usar modais estilizados para feedback de usuário.
- Gerenciamento de dados simples com JSON e Node.js.
- Integração de gráficos com Chart.js e estilização responsiva com Bootstrap.

Sugestões para Melhorias Futuras:

- Implementar autenticação de usuários para maior segurança.
- Adicionar um banco de dados relacional (ex.: MongoDB) em vez de usar JSON.
- Incluir testes automatizados com ferramentas como Jest.
- Melhorar o chat com respostas mais inteligentes (ex.: integração com IA).
- Adicionar mais tipos de gráficos no painel de relatórios.