Universidade Catolica De Moçambique e Faculdade de Gestao de Turismos e informatica

Documentação da Aplicação Front-end - Agenda de Contatos

Jose Maria Dos Santos Celso-706230146

1. Introdução

Esta documentação descreve a aplicação front-end da "Agenda de Contatos", um sistema web para gerenciamento de contatos com funcionalidades de autenticação, chat interativo, e relatórios administrativos. O objetivo é fornecer uma visão completa da estrutura, tecnologias, funcionalidades, instalação, testes, hospedagem, e sugestões de melhorias, facilitando o uso, manutenção, e evolução do projeto.

2. Informações Gerais

- **Nome da aplicação**: Agenda de Contatos
- **Objetivo principal**: Permitir que usuários gerenciem contatos, interajam via chat com um bot (limite de 5 perguntas) e, após o limite, com um administrador via WebSocket, além de oferecer relatórios detalhados para administradores.
- **Público-alvo**: Usuários que precisam organizar contatos (ex.: clientes, amigos) e administradores que monitoram atividades e geram relatórios.
- **Tecnologias utilizadas**:
    - **Front-end**: HTML5, CSS3, JavaScript (vanilla, sem frameworks)
    - **Back-end** (integrado): Node.js, Express, WebSocket (`ws`)
    - **Ferramentas**: Git, npm, wscat (para testar WebSocket)
    - **Hospedagem**: Local (pode ser hospedado em plataformas como Vercel ou Netlify)

3. Instalação e Configuração

Pré-requisitos

- **Node.js**: Versão 22.15.0 ou superior (verifique com `node -v`).
- **npm**: Incluído com Node.js (verifique com `npm -v`).
- **Git**: Para controle de versão (verifique com `git --version`).
- **Navegador**: Chrome, Firefox, ou Edge (recomenda-se Chrome para depuração).
- **Sistema operacional**: Testado em Windows (adaptável para Linux/macOS).

Comandos de instalação

1. Clone o repositório (se aplicável):
2. `git clone https://github.com/Jose105555/Desenvolvimento-de-pagina-web-com-intregracao-de-tecnologias-web-modernas.git`
3. `cd projeto web`

4. Navegue até a pasta do servidor:
5. `cd server`
6. Instale as dependências:
7. `npm install`

Como iniciar o projeto

1. Inicie o servidor:
2. `npm start`
    - **Saída esperada**:
    - `Servidor rodando em http://localhost:3000`
3. Abra o front-end no navegador:
4. `http://localhost:3000/index.html`

Configuração de variáveis de ambiente

- Não são necessárias variáveis de ambiente no momento, pois o projeto usa configurações fixas (ex.: porta 3000).
- Para mudar a porta (ex.: se 3000 estiver ocupada), edite `server/app.js`:
- `const port = 3001;`

  E atualize `public/index.html`:

  ```
  ws = new WebSocket('ws://localhost:3001');
  fetch('http://localhost:3001/api/ping');
  ```

4. Estrutura do Projeto

A aplicação segue uma estrutura modular, separando front-end e back-end:

```
projeto web/
├── public/                # Arquivos do front-end
│   ├── index.html         # Página principal (login, dashboard, chat)
│   └── estilo.css         # Estilos CSS (personalizável)
├── server/                # Arquivos do back-end
│   ├── app.js             # Configuração do servidor Express e WebSocket
│   ├── routes.js          # Rotas da API (login, contatos, relatórios)
│   └── package.json       # Dependências e scripts
└── .gitignore             # Ignora node_modules, logs, etc.
```

- **/public**:
    - `index.html`: Contém a interface de login, dashboard com abas (Contatos, Chat, Usuários, Relatórios), e lógica JavaScript para interação com o back-end.
    - `estilo.css`: Estilos visuais (atualmente mínimo, pode ser expandido).
- **/server**:
    - `app.js`: Inicializa o servidor, serve arquivos estáticos, e gerencia WebSocket.

- `routes.js`: Define endpoints da API (ex.: `/api/login`, `/api/contacts`).

5. Funcionalidades

Página de Login

- **Descrição**: Autentica usuários com username e senha.
- **Como usar**:
  - Acesse `http://localhost:3000/index.html`.
  - Insira credenciais:
    - Usuário: `user`, Senha: `user123` (role: user)
    - Administrador: `admin`, Senha: `admin123` (role: admin)
  - Clique em "Entrar" para acessar o dashboard.
- **Integração**: Chama `POST /api/login` no back-end.

Registro de Usuários

- **Descrição**: Permite criar novos usuários.
- **Como usar**:
  - Na tela de login, clique em "Registre-se".
  - Preencha: username, senha, e-mail, tipo (user ou admin).
  - Exemplo: `test`, `test123`, `test@example.com`, `user`.
- **Integração**: Chama `POST /api/register`.

Gerenciamento de Contatos

- **Descrição**: Usuários podem adicionar, listar, interagir, e excluir contatos.
- **Como usar**:
  - No dashboard, aba "Contatos":
    - **Adicionar**: Preencha nome, telefone, e-mail, categoria, aniversário.
    - **Listar**: Exibe contatos do usuário logado.
    - **Interagir**: Registra interações (ex.: ligações).
    - **Excluir**: Remove contatos.
- **Integração**: Usa `GET /api/contacts`, `POST /api/contacts`, `POST /api/contacts/:id/interact`, `DELETE /api/contacts/:id`.

Chat Interativo

- **Descrição**: Oferece um chatbot com limite de 5 perguntas; após o limite, conecta ao administrador via WebSocket.
- **Como usar**:
  - Aba "Chat":
    - Envie até 5 perguntas (ex.: "Como adicionar um contato?").
    - Após o limite, mensagem: "Limite de perguntas atingido. O administrador admin responderá em breve."

        - Envie mensagens ao administrador (ex.: "Preciso de ajuda com relatórios").
        - Administrador responde via `wscat -c ws://localhost:3000`.
- **Integração**: Usa WebSocket (`ws://localhost:3000`) e armazena histórico no `localStorage`.

Gerenciamento de Usuários (Admin)

- **Descrição**: Administradores podem listar, adicionar, e excluir usuários.
- **Como usar**:
    - Aba "Usuários" (visível apenas para role: admin):
        - Liste usuários cadastrados.
        - Adicione novos usuários.
        - Exclua usuários (exceto si mesmo).
- **Integração**: Usa `GET /api/users`, `POST /api/users`, `DELETE /api/users/:username`.

Relatórios (Admin)

- **Descrição**: Gera relatórios sobre contatos (ex.: por categoria, aniversários).
- **Como usar**:
    - Aba "Relatórios" (visível apenas para role: admin):
        - Selecione tipo (ex.: "Contatos por Categoria", "Aniversários").
        - Visualize dados agregados.
- **Integração**: Usa `GET /api/reports/:type`.

Responsividade

- **Descrição**: Interface adaptável para telas grandes e pequenas.
- **Detalhes**: Usa CSS flexível em `estilo.css` (pode ser melhorado com frameworks como Bootstrap).

7. Testes

Tipos de testes

- **Testes manuais**: Validação de login, chat, e funcionalidades de contatos/relatórios.
- **Testes de integração**: Verificação da comunicação front-end/back-end (ex.: chamadas à API).
- **Testes end-to-end**: Fluxo completo (login → chat → relatórios).

Ferramentas utilizadas

- **Navegador**: Console do Chrome (F12) para depuração.
- **wscat**: Teste de WebSocket:

- `wscat -c ws://localhost:3000`
- **curl/PowerShell**: Teste de APIs:
- `Invoke-RestMethod -Uri http://localhost:3000/api/ping`

Como executar os testes

- **Teste de servidor**:
- `cd server`
- `npm start`
- `Invoke-RestMethod -Uri http://localhost:3000/api/ping`
  - Esperado: `{"message":"Server is running"}`
- **Teste de login**:
  - Acesse `http://localhost:3000/index.html`.
  - Use `user`/`user123` e verifique o dashboard.
- **Teste de chat**:
  - Envie 5 perguntas na aba "Chat".
  - Conecte via `wscat` e responda como administrador.
- **Testes automatizados**: Não implementados (sugestão: usar Jest para JavaScript).

8. Hospedagem

Status atual

- A aplicação está configurada para execução local (`http://localhost:3000`).
- Não foi hospedada em plataformas externas (ex.: Netlify, Vercel) devido ao back-end Node.js.

Passos para hospedagem futura

1. **Front-end** (estático):
   - Hospede `public/` no Netlify:
     - Faça upload da pasta `public` ou conecte ao repositório GitHub.
     - Configure o build: `npm install` (se necessário) e copie `public/` para saída.
   - Link esperado: `https://sua-agenda-contatos.netlify.app`.
2. **Back-end**:
   - Hospede `server/` no Vercel ou Heroku:
     - Crie `vercel.json`:
     - `{`
     - `  "version": 2,`
     - `  "builds": [{ "src": "server/app.js", "use": "@vercel/node" }],`
     - `  "routes": [{ "src": "/(.*)", "dest": "server/app.js" }]`
     - `}`
     - Deploy: `vercel --prod`.

    - Atualize `index.html` com a URL do back-end (ex.: `https://seu-backend.vercel.app`).
3. **Comandos usados**:

```
vercel login
vercel
vercel --prod
```

Configuração de domínio

- Após deploy, configure um domínio personalizado em Netlify/Vercel (ex.: `www.agenda-contatos.com`).

9. Conclusão

A aplicação "Agenda de Contatos" oferece uma solução funcional para gerenciamento de contatos, com autenticação segura, chat interativo, e relatórios administrativos. Durante o desenvolvimento, foram superados desafios como:

- Erro de conexão com o servidor (`Servidor não está respondindo`).
- Erro `javascript is not defined` em `app.js`.
- Problemas de Git (`failed to push some refs`, `src refspec main does not match any`).

Lições aprendidas

- **Git**: Importância de sincronizar repositórios e usar `.gitignore`.
- **Back-end/Front-end**: Necessidade de validação rigorosa nas chamadas à API.
- **Depuração**: Uso do console do navegador e logs do servidor para identificar erros.

Sugestões para melhorias futuras

- **Front-end**: Adotar um framework (ex.: React) para componentes reutilizáveis.
- **Back-end**: Implementar banco de dados (ex.: MongoDB) para persistência.
- **Testes**: Adicionar testes automatizados com Jest e Cypress.
- **Segurança**: Implementar autenticação com JWT e criptografia de senhas.
- **UI/UX**: Melhorar responsividade com Bootstrap ou Tailwind CSS.
- **Hospedagem**: Publicar em Netlify (front-end) e Vercel (back-end).