Nome: Agostinho Marques Sanli – 706230095

Curso: Tecnologias De Informacao

Cadeira: Tecnologias De Internet

Turno: Terceiro Ano – Laboral

Documentação do Sistema de Tarefas Acadêmicas (Front-end)

1. Introdução

Esta documentação descreve os aspectos relevantes da aplicação Front-end do **Sistema de Tarefas Acadêmicas**, uma plataforma voltada para o gerenciamento de tarefas acadêmicas por alunos e administradores. O sistema inclui funcionalidades como criação e acompanhamento de tarefas, chat com bot, notificações, e relatórios, com uma interface responsiva e integração com um Back-end em Node.js.

2. Informações Gerais

- **Nome da aplicação**: Sistema de Tarefas Acadêmicas
- **Objetivo principal**: Facilitar o gerenciamento de tarefas acadêmicas, permitindo que alunos criem, acompanhem e priorizem tarefas, enquanto administradores gerenciam tarefas e usuários, com suporte via chat e notificações.
- **Público-alvo**: Alunos (para gerenciar tarefas) e administradores (para gerenciar o sistema e oferecer suporte).
- **Tecnologias utilizadas**:
    - **Front-end**: HTML5, CSS3 (com Bootstrap), JavaScript
    - **Back-end** (integrado): Node.js, Express, Socket.IO, JWT
    - **Outros**: JSON para armazenamento de dados (db.json)

3. Instalação e Configuração

Pré-requisitos

- Node.js (versão 14 ou superior)
- npm (versão 6 ou superior)
- Navegador moderno (Chrome, Firefox, etc.)

Passos de Instalação

1. Clone o repositório:

2. git clone https://github.com/Agostinho007/A_sanli_Teste2
3. Navegue até o diretório do projeto:
4. cd A_sanli_Teste2
5. Instale as dependências:
6. npm install
7. Inicie o servidor a partir do diretório root
8. node app.js

   O servidor estará disponível em http://localhost:3000 localmente.

Variáveis de Ambiente

O Back-end utiliza variáveis de ambiente. Crie um arquivo .env:

PORT=3000

Nota: No Render, configure a variável PORT no painel de configurações do serviço, se necessário.

4. Estrutura do Projeto

A estrutura do projeto é única, tudo se encontra no meu directorio.

```
├── / A_sanli_Teste2
│   ├── estilo.css (Estilos globais e responsivos)
│   ├── index.html (Página de login)
│   ├── register.html (Página de cadastro de alunos)
│   ├── dashboard.html (Painel do aluno)
│   ├── admin.html (Painel do administrador)
│   ├── notifications.html (Página de notificações)
│   ├── app.js (Servidor principal)
│   ├── routes.js (Rotas do Back-end)
│   ├── server.js (Servidor alternativo, similar a app.js)
│   ├── package.json
│   ├── db.json (Banco de dados JSON)
```

5. Funcionalidades

O Front-end oferece as seguintes funcionalidades, divididas por tipo de usuário:

Para Alunos:

- **Página de Login** (index.html): Autenticação de usuários com username e senha, gerando um token JWT.
- **Cadastro de Alunos** (register.html): Formulário para criar uma conta de aluno com nome, matrícula, username e senha.
- **Painel do Aluno** (dashboard.html):
    - **Visão Geral**: Exibe estatísticas como tarefas pendentes, concluídas, atrasadas e próximas.
    - **Tarefas**: Lista tarefas com filtros por status (Pendente, Em andamento, Concluída) e permite criar/editar tarefas.
    - **Tarefas Priorizadas**: Lista tarefas ordenadas por prioridade e data de entrega, com opções de filtragem.
    - **Relatórios**: Exibe tarefas por disciplina e tipo.
    - **Suporte**: Chat com bot (até 5 interações) e opção de contatar administrador.
    - **Notificações**: Exibe notificações sobre criação, edição ou exclusão de tarefas.
- **Notificações** (notifications.html): Página dedicada para visualizar e recarregar notificações.

Para Administradores:

- **Painel do Administrador** (admin.html):
    - **Visão Geral**: Mostra total de tarefas, tarefas pendentes, atrasadas e alunos ativos.
    - **Gerenciar Tarefas**: Permite criar, editar e excluir tarefas para qualquer aluno.
    - **Gerenciar Administradores**: Adiciona ou remove administradores (com restrição de manter pelo menos um).
    - **Relatórios**: Exibe desempenho dos alunos, tarefas por tipo, disciplina e mês.

- **Suporte aos Usuários**: Chat para suporte em tempo real com alunos.
- **Notificações**: Visualiza notificações do sistema.

Funcionalidades Comuns:

- **Responsividade**: Interface adaptada para dispositivos móveis e desktops, com media queries em estilo.css.
- **Integração com Back-end**: Consome APIs RESTful para autenticação, gerenciamento de tarefas, notificações e chats, além de WebSockets (Socket.IO) para comunicação em tempo real.
- **Chat em Tempo Real**: Suporte a mensagens entre alunos, administradores e bot, com notificações instantâneas.

6. Documentação de Código

- **Front-end**: O código JavaScript nos arquivos HTML contém comentários inline explicando a lógica de autenticação, chamadas à API e manipulação do DOM. Os arquivos HTML são estruturados com classes Bootstrap para consistência.
- **Back-end**: Os arquivos app.js, routes.js, e server.js contêm comentários explicando a lógica de rotas, autenticação JWT, e integração com Socket.IO.

- **Boas Práticas**:
    - Uso de CSS modular em estilo.css com classes reutilizáveis.
    - Tratamento de erros em chamadas à API e WebSocket.

7. Testes

- **Tipos de testes**: Não há testes automatizados implementados no Front-end ou Back-end.
- **Testes manuais**: O sistema foi testado manualmente para:
    - Autenticação (login/cadastro).

- Criação, edição e exclusão de tarefas.
- Funcionamento do chat com bot e administradores.
- Responsividade em diferentes dispositivos.

- **Ferramentas recomendadas para testes futuros**:
  - Jest e React Testing Library (caso o Front-end evolua para React).

- **Como executar testes**: Atualmente, testes são manuais. Para testes futuros, configurar um script como npm test.

8. Hospedagem

- **Plataforma**: A aplicação está hospedada no Render.
- **Passos para deploy no Render**:
  1. Crie um novo serviço Web no Render.
  2. Conecte o repositório do projeto ao Render.
  3. Configure as variáveis de ambiente no painel do Render (ex.: PORT=3000).
  4. Defina o comando de inicialização como:
  5. node app.js
  6. Deploy o serviço, garantindo que o diretório raiz seja servido para arquivos estáticos.
- **Link de acesso**: https://a-sanli-teste2.onrender.com

9. Conclusão

O Sistema de Tarefas Acadêmicas foi desenvolvido e hospedado com sucesso no Render, oferecendo uma interface intuitiva e responsiva para alunos e administradores gerenciarem tarefas acadêmicas. A integração com WebSockets garante comunicação em tempo real, e o bot de suporte melhora a experiência do usuário. **Lições aprendidas**:

- Desafios na configuração de hospedagem no Render para suportar WebSockets.
- Necessidade de testes automatizados para garantir robustez. **Sugestões para melhorias**:

- Migrar o Front-end para React ou Vue.js para melhor gerenciamento de estado.
- Adicionar criptografia para senhas no db.json.