Universidade Catolica De Moçambique e

Faculdade de Gestao de Turismos e informatica

Documentação de Estudos de língua Portuguesa

Nome: Pascoal Bernardo Sualehe

1 Introdução

Esta documentação descreve a aplicação frontend da plataforma "Estudo de Português", detalhando suas funcionalidades, estrutura, tecnologias utilizadas, instruções de uso e manutenção. A aplicação é parte de um sistema educacional que oferece materiais de estudo interativos, chat de dúvidas e ferramentas administrativas para aprendizado de português.

2 Informações Gerais

Nome da Aplicação: Estudo de Português

Objetivo Principal: Fornecer uma plataforma web para aprendizado de português com materiais (vídeos, áudios, PDFs) organizados por níveis (Inicial, Intermédio, Avançado), chat interativo para dúvidas e painel de administração para monitoramento.

Público-Alvo: Estudantes de português (iniciantes a avançados), professores e administradores educacionais.

Tecnologias Utilizadas:

HTML5: Estrutura das páginas.

CSS3: Estilização, com uso de Bootstrap (implicado pelo uso de classes como navbar, card, btn-primary).

JavaScript: Interatividade (não fornecido, mas inferido para integração com APIs e Socket.IO).

Socket.IO (client-side): Comunicação em tempo real para chat e notificações.

Bootstrap: Framework CSS para design responsivo e componentes prontos.

3. Instalação e Configuração

Pré-requisitos

Node.js (v14 ou superior) e npm para rodar o backend.

Navegador web moderno (Chrome, Firefox, Edge).

Diretório public com os arquivos HTML, CSS e JS do frontend.

Backend configurado (conforme app.js, routes.js)

4. Estrutura do Projeto

A aplicação está organizada na pasta raiz /estudos, que contém duas subpastas principais: public para o frontend e server para o backend.

/estudos/public:

Contém os arquivos do frontend, servidos estaticamente pelo Express.

Arquivos HTML: Cada um representa uma página da aplicação, com navegação consistente via barra de navegação (navbar).

estilo.css: Arquivo CSS centralizado com estilos para componentes, responsividade e transições.

/files: Armazena os arquivos de mídia referenciados em data.json (e.g., intro_gramatica.mp4, gramatica_basica.pdf).

JavaScript: Arquivos JS do frontend não foram fornecidos, mas são esperados para integração com a API (/api) e Socket.IO.

/estudos/server:

Contém os arquivos do backend, responsáveis pela lógica do servidor, APIs e comunicação em tempo real.

app.js: Configura o servidor, serve os arquivos de /public e gerencia eventos Socket.IO.

routes.js: Define rotas para gerenciar usuários, materiais, dúvidas e relatórios.

data.json: Armazena dados de forma persistente, usado diretamente pelo backend.

Essa estrutura separa o frontend (interface do usuário) do backend (lógica e dados), facilitando a manutenção e escalabilidade.

5. Funcionalidades

A aplicação oferece as seguintes funcionalidades no frontend:

5.1. Página Inicial (index.html)

Apresenta a plataforma com uma mensagem de boas-vindas e convite para login/cadastro.

Navegação para outras seções (Início, Materiais, Admin, Login).

5.2. Login (login.html)

Funcionalidade: Permite login e cadastro de usuários.

Formulários:

Login: Campo para nome de usuário, integrado com POST /api/users/login.

Cadastro: Campos para nome de usuário, tipo (Estudante/Professor), nível (Inicial/Intermédio/Avançado) e localização, integrado com POST /api/users.

Integração: Envia dados ao backend e atualiza o status online do usuário.

5.3. Materiais (materials.html)

Funcionalidade: Exibe materiais educacionais (vídeos, áudios, PDFs) categorizados por tipo.

Filtros: Materiais podem ser filtrados por nível ou tópico via GET /api/materials?level|topic.

Chat de Dúvidas: Interface para envio de mensagens, integrado com Socket.IO (userMessage).

Feedback: Modal para envio de feedback.

5.4. Tópicos (lessons.html)

Funcionalidade: Lista tópicos de estudo organizados por nível (Inicial, Intermédio, Avançado).

Integração: Usa GET /api/materials?level para carregar tópicos.

Feedback: Modal para envio de feedback.

5.5. Chat de Dúvidas (chat.html)

Funcionalidade: Interface dedicada para interação em tempo real com professores/administradores.

Seleção de Material: Dropdown para escolher o material relacionado à dúvida.

Mensagens: Envio de mensagens via Socket.IO (userMessage) e recebimento de respostas (adminMessage)

Feedback: Modal para envio de feedback.

5.6. Painel de Administração (admin.html)

Funcionalidade: Ferramenta para administradores monitorarem a plataforma.

Autenticação: Campo para chave de administrador, integrado com POST /api/admin/auth.

Monitoramento:

Usuários Online: Lista usuários ativos (userStatus).

Atividade do Sistema: Registra eventos como logins, acessos a materiais e mensagens (systemActivity).

Conversas do Chat: Exibe mensagens de dúvidas e permite respostas (adminMessage).

Integração: Usa Socket.IO para atualizações em tempo real.

5.7. Responsividade

Design parcialmente responsivo, com uso de Bootstrap e estilos em estilo.css (e.g., .material-video, .navbar).

Transições visuais (e.g., card:hover) e suporte a diferentes formatos de mídia.

5.8. Integração com APIs

Endpoints Utilizados:

POST /api/users, POST /api/users/login, POST /api/users/logout: Gerenciamento de usuários.

GET /api/materials, GET /api/materials/:id: Listagem e acesso a materiais.

POST /api/doubts, PUT /api/doubts: Criação e atualização de dúvidas.

POST /api/admin/auth: Autenticação de administradores.

GET /api/reports/doubts, GET /api/reports/usage: Relatórios de uso.

Socket.IO:

Eventos como newUser, userLogin, userMessage, adminMessage para comunicação em tempo real.

6. Documentação de Código

Comentários: Os arquivos HTML e CSS não contêm comentários extensivos, mas a estrutura é clara, com classes semânticas (e.g., .material-section, .chatMessages).

Boas Práticas:

Uso de classes consistentes para estilização (e.g., btn-primary, card).

Organização modular das páginas em arquivos HTML separados.

CSS centralizado em estilo.css para manutenção fácil.

7. Testes

Estado Atual: Não há testes implementados no frontend.

Recomendações:

Testes Unitários: Usar Jest para testar funções JavaScript (e.g., validação de formulários).

Testes de Integração: Usar React Testing Library (se o frontend migrar para React) ou Cypress para testar fluxos (e.g., login, envio de mensagem).

Testes End-to-End: Usar Cypress para simular navegação e interações completas.

8. Hospedagem

Estado Atual: A aplicação está configurada para rodar localmente (http://localhost:3000).

Passos para Deploy (recomendados):

Backend:

Hospede o servidor Node.js em plataformas como Heroku, Render ou AWS.

Configure um banco de dados (e.g., MongoDB Atlas) para substituir data.json.

Use um serviço de armazenamento (e.g., AWS S3) para os arquivos de mídia.

Frontend:

Hospede os arquivos estáticos (/public) em Netlify, Vercel ou GitHub Pages.

Configure o backend como proxy para chamadas API (e.g., /api redirecionado para o servidor).

9. Conclusão

A aplicação "Estudo de Português" é uma plataforma educacional robusta, com um frontend bem estruturado que suporta aprendizado interativo e administração eficiente. As principais conquistas incluem:

Interface amigável com navegação consistente.

Integração com backend para gerenciamento de usuários, materiais e chat em tempo real.

Design visual atraente com suporte a diferentes formatos de mídia.

Lições Aprendidas:

A importância de planejar a integração frontend-backend desde o início.

A necessidade de testes para garantir robustez.

Limitações do armazenamento baseado em arquivos (data.json) para escalabilidade.