DOCUMENTAÇÃO COMPLETA — DentalShow

---

1. Visão Geral do Projeto

- **Nome:** DentalShow
- **Propósito:** Gestão clínica odontológica.
- **Stack:** Python (Flask) + SQLite → PostgreSQL/MySQL + React + TypeScript + Tailwind CSS.
- **API RESTful:** Comunicação frontend/backend.
- **Segurança:** JWT e validação no backend.

---

2. Backend

- **Python 3.11+ / Flask**
- **Banco:** SQLite (pode migrar)
- **Endpoints RESTful:**
  `/pacientes`, `/agendamentos`, `/receituarios`, `/prontuarios`, `/exames`, `/pagamentos`, `/produtosservicos`
- **Autenticação:** JWT Token, obrigatório em rotas protegidas.
- **Validações:** Backend sempre confere dados.
- **Organização:** Um arquivo principal centraliza tudo, fácil modularizar depois.

---

3. Frontend

- **React + TypeScript + Vite**
- **Tailwind CSS** para estilização
- **Organização:**
  - Cada módulo: Page, Form, List, Detalhe, Service
  - Rotas em `App.tsx`
  - Tipos em `types.ts`
- **Autenticação:**
  - Token JWT no localStorage
  - Protege rotas
  - Logout limpa token

---

4. Arquitetura dos Arquivos

Por módulo (exemplo: Pacientes):

- `PacientesPage.tsx`: Página principal/lista
- `FormPaciente.tsx`: Cadastro/edição
- `PacienteDetalhe.tsx`: Visualização individual
- `ListPacientes.tsx`: Lista/tabulação

- `pacientes.ts` (service): Integração API

---

5. Segurança

- **JWT**: usado em todas as requisições.
- **Rotas privadas**: só com token.
- **Validação de dados**: front e back.
- **Tratamento de erro**: feedback ao usuário.

---

6. Como o login funciona

- Usuário entra/login → backend responde com token.
- Front salva token e protege páginas.
- Logout = limpa token e volta ao login.

---

7. Escalabilidade

- Backend facilmente migrável (ORM, container/Docker, cloud).
- Frontend pode ser hospedado em Vercel/Netlify.
- API desacoplada, só mudar URL.
- Cada módulo independente (fácil adicionar/remover).

---

8. Ideias para melhoria

- PDF/Excel automático
- Permissões avançadas (admin/usuário)
- Log/auditoria de operações
- Integração Google Agenda
- Dashboard de indicadores
- Notificações e UX

---

9. Resumo dos arquivos

Page: tela principal do módulo
Form: cadastro/edição
List: listagem
Detalhe: visualização individual
Service: integrações API
types.ts: tipagem global

---

10. Migração e evolução

- Troca banco? Só mudar driver/URI no backend.
- Backend ou front em nuvem? Só ajustar build/deploy.
- Precisa novo módulo? Siga a estrutura atual: crie Page, Form, List, Service e tipos.

---

11. Dica de uso

- Use esta documentação como **norte**.
- Antes de novo módulo, revise aqui a estrutura.
- Ao criar arquivos, siga a separação por módulo/funcionalidade.
- Sempre pense na separação entre backend (dados) e frontend (apresentação).