Estrutura do Projeto

Visão Geral

O projeto está organizado da seguinte forma:

```
clinicalmf-producao/
├── frontend/           # Aplicação Next.js
├── backend/            # API FastAPI
│   ├── config/         # Configurações
│   ├── repositories/   # Acesso a dados
│   ├── services/       # Lógica de negócios
│   ├── scripts/        # Scripts utilitários
│   └── app.py          # Aplicação principal
├── docs/               # Documentação geral
├── instrucoes/         # Guias e instruções
├── sql/                # Scripts SQL
├── Unimed_Scraping/    # Scripts de integração com Unimed
└── templates/          # Templates e modelos
```

Backend (backend/)

Estrutura de Diretórios do Backend

```
backend/
├── config/                 # Configurações do projeto
│   ├── __init__.py
│   └── config.py          # Configurações globais e cliente Supabase
├── repositories/          # Camada de acesso a dados
│   ├── __init__.py
│   ├── database_supabase.py
│   ├── auditoria_repository.py
│   ├── paciente.py        # Repositório de pacientes
│   └── ...
├── services/             # Lógica de negócios
│   ├── __init__.py
│   ├── auditoria_service.py
│   ├── importacao_service.py  # Serviço de importação de dados externos
│   └── ...
├── scripts/              # Scripts utilitários
│   ├── __init__.py
│   ├── gerar_dados_antigo.py
│   ├── gerar_dados_teste.py
│   ├── gerar_dados_de_testes.py
│   └── importar_pacientes.py  # Script de importação de pacientes
└── app.py               # Aplicação FastAPI principal
```

Componentes Principais

Configurações (`config/`)

- `config.py`: Configurações globais usando Pydantic Settings
- Gerencia conexão com Supabase e variáveis de ambiente
- Centraliza todas as configurações em um único local

Repositórios (`repositories/`)

- Camada de acesso a dados
- Implementa operações CRUD com o Supabase
- Separa lógica de banco de dados da lógica de negócios

Serviços (`services/`)

- Implementa a lógica de negócios
- Utiliza os repositórios para operações de dados
- Processa regras de negócio e validações
- Inclui serviços de importação de dados externos (MySQL)

Scripts (`scripts/`)

- Scripts utilitários para testes e desenvolvimento
- Geração de dados de teste
- Ferramentas de manutenção
- Scripts de importação de dados de sistemas externos

Frontend (`frontend/`)

```
frontend/
├── src/
│   ├── app/              # Páginas e rotas
│   ├── components/       # Componentes reutilizáveis
│   ├── hooks/           # Hooks personalizados
│   ├── services/        # Serviços e APIs
│   ├── styles/          # Estilos globais
│   └── utils/           # Utilitários
├── public/              # Arquivos estáticos
├── .env.example         # Exemplo de variáveis de ambiente
└── package.json         # Dependências e scripts
```

Documentação (`docs/`)

```
docs/
├── visao_geral.md            # Visão geral do sistema
├── scripts-sql.md            # Documentação SQL
├── database/                 # Documentação do Banco de Dados
│   ├── schema.md             # Diagrama de Relacionamentos
│   └── fluxo_dados.md        # Fluxo de Dados
├── sistema_auditoria/        # Documentação do Sistema de Auditoria
│   ├── visao_geral.md        # Visão Geral
│   ├── tipos_divergencias.md # Tipos de Divergências
│   ├── processo_auditoria.md # Processo de Auditoria
│   └── ...
├── faturamento/              # Documentação da Tabela de Faturamento
│   ├── estrutura_tabela.md  # Estrutura da Tabela
│   ├── processo_preenchimento.md # Processo de Preenchimento
│   └── ...
├── fichas_processamento/     # Documentação do Fluxo de Fichas
│   ├── visao_geral.md     # Visão Geral do Processo
│   ├── captura_execucoes.md # Captura de Execuções
│   └── ...
└── melhorias_futuras.md      # Recomendações para Melhorias
```

Instruções (`instrucoes/`)

```
instrucoes/
├── instrucoes_backend.md    # Guia do backend
├── instrucoes_frontend.md   # Guia do frontend
├── instrucoes_utils.md      # Guia de utilitários
├── instrucoes_testes.md     # Guia de testes
├── instrucoes_deploy.md     # Guia de deploy
└── instrucoes_sql.md        # Instruções SQL
```

SQL (`sql/`)

```
sql/
├── 00_reset_database.sql        # Reseta o banco de dados (cuidado!)
├── 01_tipos_enums.sql           # Define tipos ENUM personalizados
├── 02_funcoes_auxiliares.sql    # Funções auxiliares (ex: update_updated_at)
├── 03_criar_tabelas.sql         # Criação das tabelas principais
├── 04_indices.sql               # Criação de índices para otimização
├── 05_funcoes_negocio.sql       # Funções de negócio (ex: RPCs)
├── 06_triggers.sql              # Criação de triggers
├── 07_views.sql                 # Criação de views
├── 08_seguranca_permissoes.sql  # Configuração de RLS e permissões
├── scripts/                     # Scripts SQL diversos (ex: migrações antigas,
testes)
├── migrations/                  # Migrações incrementais
└── ...                          # Outros arquivos SQL específicos ou antigos
```

Execução de Scripts

Para executar os scripts de teste, use o seguinte formato:

```
# Executar scripts do backend
python -m backend.scripts.gerar_dados_teste
python -m backend.scripts.gerar_dados_de_testes

# Executar a aplicação principal
python -m backend.app
```

Boas Práticas

1. **Organização de Código**
    - Mantenha os arquivos organizados em suas respectivas pastas
    - Use imports relativos para referências internas
    - Siga o padrão de nomenclatura estabelecido
2. **Configurações**
    - Use variáveis de ambiente para configurações sensíveis
    - Mantenha as configurações centralizadas em `backend/config/config.py`
    - Não hardcode valores sensíveis no código
3. **Documentação**
    - Mantenha a documentação atualizada
    - Documente alterações importantes
    - Siga o padrão de documentação estabelecido
4. **Testes**
    - Use os scripts de teste apropriados
    - Mantenha os dados de teste consistentes
    - Documente casos de teste importantes
5. **Segurança**
    - Configure RLS adequadamente
    - Use políticas de acesso
    - Proteja dados sensíveis
6. **Performance**
    - Otimize queries
    - Use índices estratégicos

    - Monitore performance

Módulos Específicos

Sistema de Auditoria

Implementado nos arquivos `auditoria_service.py` e `auditoria_repository.py`.

Sistema de Importação de Dados

- Importação de pacientes do MySQL via SSH tunnel
- Mapeamento e validação de dados entre sistemas
- Controle de importação incremental