Reestruturação Frontend/Backend - Sentinela IA

Nova Estrutura Proposta



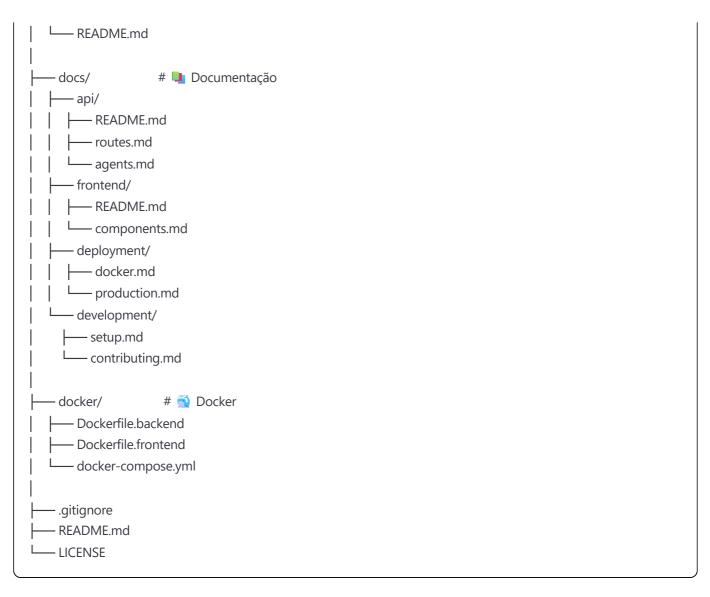
```
train_inteligente.py
                    # Scripts utilitários
    scripts/
      - __init__.py
      popular_banco.py
      migrate_structure.py
                   # Testes
   - tests/
      __init__.py
      test_agent_system.py
      test_routes.py
    — test_services.py
   - requirements.txt
                        # Dependências Python
   - .env.example
                    # Ponto de entrada do backend
   run.py
                    # Para produção (Gunicorn/uWSGI)
   - wsgi.py
- frontend/
                     # # Frontend Web
  — public/
                    # Arquivos estáticos públicos
      index.html
      favicon.ico
      – manifest.json
                  # Código fonte do frontend
      - assets/
                    # Assets (CSS, imagens, fontes)
         css/
          — main.css
            components.css
         images/
          - fonts/
                  # JavaScript
          - main.js
          components/
             consulta.js
            – analise.js
            – analise_IA.js

nova_ocorrencia.js

          services/ # Cliente para APIs
             - api.js
            - consulta-service.js

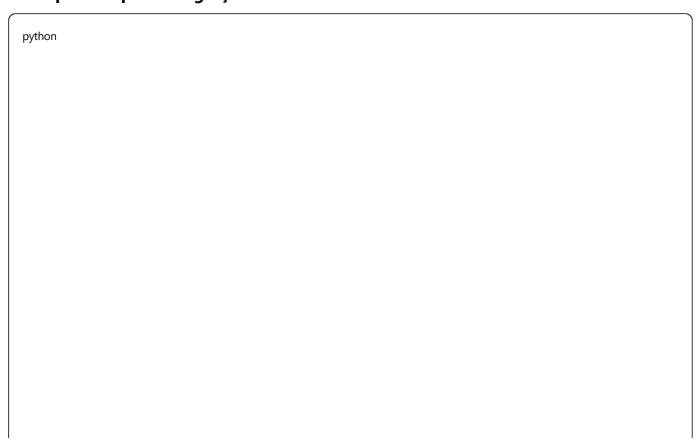
analise-service.js

         - utils/
                   # Utilitários JS
        formatters.js
        └── validators.js
       templates/
                      # Templates HTML
         - base.html
         - consulta.html
         - nova_ocorrencia.html
         - analise.html
        - analise_IA.html
    package.json
                     # Se usar npm/yarn
    webpack.config.js # Se usar webpack
```



Scripts de Migração

1. Script Principal de Migração



```
#!/usr/bin/env python3
migrate_to_structure.py
Script para migrar para nova estrutura frontend/backend
import os
import shutil
from pathlib import Path
def create_new_structure():
  """Cria nova estrutura de diretórios"""
  directories = [
     # Backend
    "backend/app/models",
    "backend/app/services/agents",
    "backend/app/routes",
    "backend/app/utils",
    "backend/config/agents",
    "backend/ml_models/trained",
    "backend/ml_models/training",
    "backend/scripts",
    "backend/tests",
     # Frontend
    "frontend/public",
    "frontend/src/assets/css",
    "frontend/src/assets/images",
    "frontend/src/assets/fonts",
    "frontend/src/js/components",
    "frontend/src/js/services",
    "frontend/src/js/utils",
    "frontend/src/templates",
     # Docs e Docker
    "docs/api",
    "docs/frontend",
    "docs/deployment",
    "docs/development",
    "docker"
  for directory in directories:
     Path(directory).mkdir(parents=True, exist_ok=True)
     # Criar __init__.py para módulos Python
    if ("backend/" in directory and
       not any(x in directory for x in ["scripts", "tests", "config"])):
       init_file = Path(directory) / "__init__.py"
```

```
init_file.touch()
  print(" ✓ Nova estrutura de diretórios criada")
def migrate_files():
  """Migra arquivos para nova estrutura"""
  file_mappings = {
    # Backend
    "app/main.py": "backend/run.py",
    "app/__init__.py": "backend/app/__init__.py",
    "app/models/database.py": "backend/app/models/database.py",
    "app/services/placa_service.py": "backend/app/services/placa_service.py",
    "app/services/semantic_service.py": "backend/app/services/semantic_service.py",
    "app/services/enhanced_placa_service.py": "backend/app/services/enhanced_placa_service.py",
    "app/services/agents/": "backend/app/services/agents/",
    "app/routes/main_routes.py": "backend/app/routes/main_routes.py",
    "app/routes/analise_routes.py": "backend/app/routes/analise_routes.py",
    "app/routes/agent_routes.py": "backend/app/routes/agent_routes.py",
    "config/": "backend/config/",
    "ml models/": "backend/ml models/",
    "scripts/": "backend/scripts/",
    "tests/": "backend/tests/",
    "requirements.txt": "backend/requirements.txt",
    ".env.example": "backend/.env.example",
    # Frontend
    "app/templates/": "frontend/src/templates/",
    "static/js/": "frontend/src/js/",
    "static/css/": "frontend/src/assets/css/",
    "static/images/": "frontend/src/assets/images/"
  for source, destination in file_mappings.items():
    if os.path.exists(source):
       try:
         dest_dir = os.path.dirname(destination)
         Path(dest_dir).mkdir(parents=True, exist_ok=True)
         if source.endswith('/'): # Directory
            if os.path.exists(destination):
              shutil.rmtree(destination)
            shutil.copytree(source, destination)
         else: # File
            shutil.copy2(source, destination)
         print(f" ✓ Migrado: {source} → {destination}")
       except Exception as e:
         print(f" X Erro ao migrar {source}: {e}")
```

```
def create_new_configs():
  """Cria novos arquivos de configuração"""
  # Backend: run.py
  backend_run = ""#!/usr/bin/env python3
Backend do Sistema Sentinela IA
from app import create_app
from config.settings import criar_tabelas
import os
app = create_app()
def init_database():
  """Inicializa banco de dados"""
    criar_tabelas()
    print(" ✓ Banco de dados inicializado")
  except Exception as e:
    print(f" X Erro ao inicializar banco: {e}")
if __name__ == '__main__':
  init_database()
  # Configuração do servidor
  host = os.getenv('HOST', '0.0.0.0')
  port = int(os.getenv('PORT', 5000))
  debug = os.getenv('FLASK_ENV', 'development') == 'development'
  app.run(host=host, port=port, debug=debug)
  # Backend: wsgi.py (para produção)
  wsgi_py = """"
WSGI Entry point for production deployment
from app import create_app
from config.settings import criar_tabelas
# Inicializar banco
criar_tabelas()
# Criar aplicação
application = create_app()
```

```
if __name__ == "__main__":
  application.run()
  # Frontend: package.json
  package_json = "'{
 "name": "sentinela-ia-frontend",
 "version": "2.0.0",
 "description": "Frontend do Sistema de Análise de Placas",
 "main": "src/js/main.js",
 "scripts": {
  "start": "webpack serve --mode development",
  "build": "webpack --mode production",
  "watch": "webpack --mode development --watch",
  "lint": "eslint src/js/**/*.js",
  "format": "prettier --write src/js/**/*.js"
 },
 "dependencies": {
  "axios": "^1.5.0",
  "plotly.js": "^2.26.0"
 },
 "devDependencies": {
  "webpack": "^5.88.0",
  "webpack-cli": "^5.1.0",
  "webpack-dev-server": "^4.15.0",
  "css-loader": "^6.8.0",
  "style-loader": "^3.3.0",
  "html-webpack-plugin": "^5.5.0",
  "eslint": "^8.47.0",
  "prettier": "^3.0.0"
},
 "author": "Sentinela IA Team",
 "license": "MIT"
}'''
  # Docker Compose
  docker_compose = "'version: '3.8'
services:
 postgres:
  image: postgres:15-alpine
  environment:
   POSTGRES_DB: sentinela_db
   POSTGRES_USER: postgres
   POSTGRES_PASSWORD: Jmkjmk.00
  ports:
   - "5432:5432"
  volumes:
   - postgres_data:/var/lib/postgresql/data
```

```
healthcheck:
   test: ["CMD-SHELL", "pg_isready -U postgres"]
   timeout: 5s
   retries: 5
 backend:
  build:
   context: ./backend
   dockerfile: ../docker/Dockerfile.backend
  ports:
   - "5000:5000"
  environment:
   - FLASK_ENV=production
   - DB_HOST=postgres
   - DB_PORT=5432
  depends_on:
   postgres:
    condition: service_healthy
  volumes:
   - ./backend:/app
  command: gunicorn --bind 0.0.0.0:5000 wsgi:application
 frontend:
  build:
   context: ./frontend
   dockerfile: ../docker/Dockerfile.frontend
  ports:
   - "3000:80"
  depends_on:
   - backend
  volumes:
   - ./frontend/dist:/usr/share/nginx/html:ro
volumes:
 postgres_data:
  # Escrever arquivos
  files_to_create = {
    "backend/run.py": backend_run,
    "backend/wsgi.py": wsgi_py,
    "frontend/package.json": package_json,
    "docker/docker-compose.yml": docker_compose
  for filepath, content in files_to_create.items():
    with open(filepath, 'w', encoding='utf-8') as f:
       f.write(content)
```

```
print(f" ✓ Criado: {filepath}")
def create_dockerfiles():
  """Cria Dockerfiles para backend e frontend"""
  # Dockerfile Backend
  dockerfile_backend = "'FROM python:3.11-slim
WORKDIR /app
# Instalar dependências do sistema
RUN apt-get update && apt-get install -y \\
  gcc \\
  g++\\
  && rm -rf /var/lib/apt/lists/*
# Copiar requirements e instalar dependências Python
COPY requirements.txt.
RUN pip install --no-cache-dir -r requirements.txt
# Baixar modelo spaCy
RUN python -m spacy download pt_core_news_sm
# Copiar código
COPY..
# Expor porta
EXPOSE 5000
# Comando padrão
CMD ["python", "run.py"]
  # Dockerfile Frontend
  dockerfile_frontend = "'FROM node:18-alpine as builder
WORKDIR /app
# Copiar package files
COPY package*.json ./
# Instalar dependências
RUN npm ci --only=production
# Copiar código fonte
COPY..
# Build da aplicação
RUN npm run build
```

```
# Stage de produção com nginx
FROM nginx:alpine
# Copiar build
COPY --from=builder /app/dist /usr/share/nginx/html
# Copiar configuração do nginx
COPY nginx.conf /etc/nginx/nginx.conf
# Expor porta
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
  # Nginx config
  nginx_conf = ""events {
  worker_connections 1024;
}
http {
  include /etc/nginx/mime.types;
  default_type application/octet-stream;
  server {
     listen 80;
     root /usr/share/nginx/html;
     index index.html;
     location / {
       try_files $uri $uri/ /index.html;
    }
     location /api/ {
       proxy_pass http://backend:5000;
       proxy_set_header Host $host;
       proxy_set_header X-Real-IP $remote_addr;
    }
  }
}
  files = {
     "docker/Dockerfile.backend": dockerfile_backend,
     "docker/Dockerfile.frontend": dockerfile_frontend,
     "frontend/nginx.conf": nginx_conf
```

```
for filepath, content in files.items():
    with open(filepath, 'w', encoding='utf-8') as f:
       f.write(content)
    def update_imports():
  """Atualiza imports nos arquivos migrados"""
  print(" <a> Atualizando imports...")</a>
  # Arquivos que precisam ter imports atualizados
  files_to_update = [
    "backend/app/routes/main_routes.py",
    "backend/app/routes/analise_routes.py",
    "backend/app/routes/agent_routes.py",
    "backend/app/services/placa_service.py",
    "backend/run.py"
  import_replacements = {
    "from app.models.database": "from app.models.database",
    "from app.services.": "from app.services.",
    "from config.settings": "from config.settings",
    "from config.": "from config."
  }
  for filepath in files_to_update:
    if os.path.exists(filepath):
       try:
         with open(filepath, 'r', encoding='utf-8') as f:
            content = f.read()
         for old, new in import_replacements.items():
            content = content.replace(old, new)
         with open(filepath, 'w', encoding='utf-8') as f:
            f.write(content)
         print(f" ✓ Imports atualizados: {filepath}")
       except Exception as e:
         print(f" X Erro ao atualizar {filepath}: {e}")
def create_documentation():
  """Cria documentação básica"""
  readme_main = ""# Sistema Sentinela IA v2.0
```

Sistema de análise inteligente de placas veiculares com arquitetura moderna frontend/backend.

```
- **Backend**: Python/Flask com sistema de agentes especializados
- **Frontend**: HTML/CSS/JS moderno com build system
- **Database**: PostgreSQL
- **ML**: Modelos de classificação semântica e análise de rotas

### Com Docker (Recomendado)

"bash
docker-compose -f docker/docker-compose.yml up
```

Desenvolvimento Local

Backend

```
bash

cd backend/
pip install -r requirements.txt
python run.py
```

Frontend

bash

cd frontend/
npm install
npm start

Documentação

- API Backend
- Frontend
- <u>Deploy</u>

Desenvolvimento

Ver docs/development/setup.md "

```
backend_readme = ""# Backend - Sentinela IA
```

API Flask com sistema de agentes especializados para análise de placas.

🏃 Execução

bash

pip install -r requirements.txt python run.py

Estrutura

- (app/) Aplicação Flask
- (config/) Configurações
- (ml_models/) Modelos de ML
- scripts/) Scripts utilitários
- (tests/) Testes

Legacy API

- (GET /api/consulta_placa/{placa}) Consulta por placa
- (GET /api/analise_placa/{placa}) Análise de risco

Agents API v2

- (GET /api/v2/analyze/{placa}) Análise completa
- (GET /api/v2/analyze/{placa}/fast) Análise rápida
- (POST /api/v2/analyze/batch) Análise em lote

🥓 Testes

bash

python -m pytest tests/

111

frontend_readme = ""# Frontend - Sentinela IA

Interface web para o sistema de análise de placas.

Desenvolvimento

bash

npm install

npm start



bash

npm run build

Estrutura

- (src/templates/) Templates HTML
- (src/js/) JavaScript
- (src/assets/) CSS, imagens, fontes
- (public/) Arquivos estáticos

Componentes

Páginas

- Consulta Busca por placa/CPF
- Nova Ocorrência Registro de ocorrências
- Análise Dashboard analítico
- Análise IA Análise por ML

Serviços

- API Client
- Formatação de dados
- Validações "' files = { "README.md": readme_main, "backend/README.md": backend_readme,
 "frontend/README.md": frontend_readme } for filepath, content in files.items(): with open(filepath,
 'w', encoding='utf-8') as f: f.write(content) print(f" Documentação criada: {filepath}")

def main():

```
"""Executa migração completa"""
print(" 
Migrando para estrutura Frontend/Backend")
print("=" * 50)
```

```
create_new_structure()
migrate_files()
create_new_configs()
create_dockerfiles()
update_imports()
create_documentation()
print("\n 🎉 Migração concluída!")
print("\n > Próximos passos:")
print("1. cd backend && pip install -r requirements.txt")
print("2. cd frontend && npm install")
print("3. Teste: docker-compose -f docker/docker-compose.yml up")
print("4. Acesse: http://localhost:3000")
```

```
if name == "main": main()
  ## 6 Benefícios da Nova Estrutura
  ### <a> Separação Clara</a>
  - Backend focado em API e lógica de negócio
  - Frontend focado em interface de usuário
  - Facilita desenvolvimento em equipe
  ### Z Escalabilidade
  - Backend pode ser deployado independentemente
  - Frontend pode usar CDN
  - Múltiplas instâncias de cada parte
  ### <a href="#"> Tecnologia Moderna</a>
  - Build system para frontend (Webpack)
  - Docker para containerização
  - CI/CD mais simples
  ### Manutenibilidade
  - Código mais organizado
```

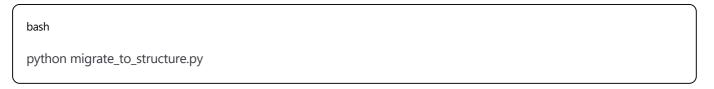
2. Execute o script de migração

- Testes separados por camada - Documentação estruturada

K Como Executar a Migração

1. **Faça backup do projeto atual**

cp -r . ../sentinela-ia-backup



3. Teste a nova estrutura

```
bash

# Backend

cd backend/
pip install -r requirements.txt
python run.py

# Frontend (novo terminal)
cd frontend/
npm install
npm start
```

4. Ou use Docker

bash

docker-compose -f docker/docker-compose.yml up

Checklist de Migração

- Backup do projeto atual
- Executar script de migração
- Testar backend isoladamente
- Configurar build do frontend
- Testar integração completa
- Atualizar documentação
- ☐ Configurar CI/CD (opcional)

Esta estrutura permitirá um desenvolvimento mais organizado e facilitará futuras evoluções do sistema!