Migração da Aplicação To-Do List para Django

Documento gerado em: 03/10/2025 às 22:27

■ Resumo da Migração

A aplicação **To-Do List** foi migrada com sucesso de **FastAPI + SQLAlchemy** para **Django + Django REST Framework**. Esta migração mantém todas as funcionalidades existentes enquanto aproveita os benefícios do framework Django.

■ Mudanças Realizadas

■ *Tecnologias Removidas*

• FastAPI - Framework web assíncrono
• SQLAlchemy - ORM
• Alembic - Sistema de migrações
• Pydantic - Validação de dados

■ *Tecnologias Implementadas*

• Django 5.0.1 - Framework web Python
• Django REST Framework 3.14.0 - API REST
• Django CORS Headers 4.3.1 - Configuração CORS
• psycopg2-binary 2.9.9 - Driver PostgreSQL
• python-decouple 3.8 - Gerenciamento de variáveis

■ Nova Estrutura de Arquivos

```
backend/ ■■■ manage.py # Script de gerenciamento Django ■■■ requirements.txt
# Dependências Python ■■■ env.example # Exemplo de configuração ■■■
todolist/ # Projeto principal Django ■ ■■■ __init__.py ■ ■■■ settings.py #
Configurações do projeto ■ ■■■ urls.py # URLs principais ■ ■■■ wsgi.py #
Configuração WSGI ■ ■■■ asgi.py # Configuração ASGI ■■■ tasks/ # App Django
para tarefas ■■■ __init__.py ■■■ apps.py # Configuração do app ■■■
models.py # Modelo Task (Django ORM) ■■■ views.py # Views da API REST ■■■
serializers.py # Serializers para validação ■■■ urls.py # URLs do app ■■■
admin.py # Interface administrativa
```

■■ Arquivos Criados

1. requirements.txt

```
Django==5.0.1 djangorestframework==3.14.0 django-cors-headers==4.3.1
psycopg2-binary==2.9.9 python-decouple==3.8
```

2. tasks/models.py - Modelo Task migrado para Django ORM

```
class Task(models.Model): STATUS_CHOICES = [ ('pendente', 'Pendente'),
('concluida', 'Concluída'), ] titulo = models.TextField() descricao =
models.TextField(blank=True, null=True) status = models.CharField(
max_length=20, choices=STATUS_CHOICES, default='pendente' ) created_at =
models.DateTimeField(auto_now_add=True) updated_at =
models.DateTimeField(auto_now=True)
```

■ Rotas da API

A API mantém exatamente as mesmas rotas que o FastAPI:

```
GET /tasks/ Lista todas as tarefas GET /tasks/{id}/ Busca tarefa por ID POST
/tasks/ Cria nova tarefa PUT /tasks/{id}/ Atualiza tarefa completa PATCH
/tasks/{id}/status/ Atualiza apenas status DELETE /tasks/{id}/ Exclui tarefa
```

■■ Configuração do Ambiente

1. Instalar Dependências

```
cd backend
pip install -r requirements.txt
```

2. Configurar Variáveis de Ambiente

```
SECRET_KEY=django-insecure-change-me-in-production DEBUG=True
DATABASE_URL=postgresql://appuser:apppass@127.0.0.1:5432/todolist
```

3. Executar Migrações

```
python manage.py makemigrations
python manage.py migrate
```

4. Iniciar Servidor

```
python manage.py runserver
```

■ Benefícios da Migração

■ Vantagens do Django

• ORM Poderoso - Django ORM é mais intuitivo e completo • Admin Interface - Interface administrativa automática • Sistema de Migrações - Migrações integradas ao framework • Segurança - Proteções de segurança built-in • Documentação - Documentação extensa e comunidade ativa • Escalabilidade - Framework maduro e testado

■ *Funcionalidades Mantidas*

• Todas as rotas da API • Validação de dados • Configuração CORS • Compatibilidade com frontend React • Funcionalidades de CRUD completas

■ Comandos Django Úteis

```
python manage.py makemigrations # Criar migrações python manage.py migrate # Aplicar migrações python manage.py createsuperuser # Criar superusuário python manage.py runserver # Iniciar servidor python manage.py shell # Acessar shell Django python manage.py check # Verificar configuração
```

■ Próximos Passos

1. Testar a aplicação - Verificar se todas as funcionalidades estão funcionando 2. Configurar produção - Ajustar configurações para ambiente de produção 3. Implementar testes - Adicionar testes unitários e de integração 4. Documentar API - Usar ferramentas como drf-spectacular para documentação 5. Otimizar performance - Implementar cache e otimizações de banco

■ Conclusão

A migração foi realizada com sucesso, mantendo a compatibilidade total com o frontend React existente. A aplicação agora utiliza o Django como framework backend, aproveitando seus recursos avançados de ORM, sistema de migrações e interface administrativa.

Data da Migração: 03/10/2025 às 22:27
Versão Django: 5.0.1
Status: ■ Concluída com sucesso