GAC116-14A - Programação Web

Relatório Final

Professor: Professor Raphael Winckler de Bettio
Aluno(s): Carlos Eduardo Borges de Sousa

Lavras-MG
2025

1. Introdução

Este relatório tem como objetivo apresentar o desenvolvimento de um projeto prático para a disciplina de Programação Web, cujo desafio consistiu na construção de uma API RESTful para um sistema de blog. A proposta envolveu duas etapas principais: inicialmente, a implementação utilizando o framework Django com Django REST Framework (DRF), adotado na disciplina; e, posteriormente, a reimplementação do mesmo sistema com uma tecnologia de livre escolha — neste caso, o framework FastAPI.

O sistema desenvolvido permite que usuários se cadastrem, criem postagens, editem ou deletem tanto suas informações quanto seus próprios conteúdos. Ambas as versões do backend foram construídas com foco na organização, segurança e clareza das rotas, além da persistência local dos dados utilizando o banco de dados SQLite.

Foi elaborado uma comparação entre as duas tecnologias analisando as semelhanças, diferenças, vantagens e limitações observadas no uso de Django/DRF e FastAPI, destacando os critérios que podem orientar a escolha entre um ou outro em diferentes contextos de desenvolvimento.

2. Objetivo

O objetivo deste trabalho é desenvolver uma API RESTful para um sistema de blog, utilizando duas tecnologias distintas: Django/DRF e FastAPI. Além da implementação funcional, busca-se realizar uma comparação entre os frameworks, analisando suas principais diferenças em termos de estrutura, desempenho, facilidade de uso e adequação a diferentes contextos de desenvolvimento.

3. Visão Geral da Implementação

3.1. Django

O backend do projeto foi desenvolvido utilizando Django e Django REST Framework (DRF), duas tecnologias amplamente reconhecidas para construção de APIs robustas e seguras em Python. O sistema oferece funcionalidades de cadastro e gerenciamento de usuários, além do CRUD completo de posts, expostos via endpoints RESTful.

Estrutura do Projeto
blog/: Aplicação principal, contendo modelos, views, serializers, testes e rotas.
project_blog/: Configurações globais do projeto Django.
db.sqlite3: Banco de dados SQLite utilizado para persistência local.
manage.py: Script utilitário para comandos administrativos.

3.1. FastAPI

O backend deste projeto foi desenvolvido utilizando FastAPI, um framework moderno e de alto desempenho para construção de APIs em Python. O sistema oferece funcionalidades de cadastro e gerenciamento de usuários, além do CRUD completo de posts, expostos via endpoints RESTful.

Estrutura do Projeto
fastapi_backend/app/: Contém os módulos principais do backend, incluindo modelos, esquemas (schemas), operações CRUD, rotas (routers) e configuração do banco de dados.
blog.db: Banco de dados SQLite utilizado para persistência local.
requirements.txt: Lista de dependências do projeto.
main.py: Ponto de entrada da aplicação FastAPI.

4. Diferenças entre as tecnologias

Durante o desenvolvimento do projeto "Blog API", foi possível explorar dois frameworks distintos para a construção de APIs RESTful em Python: **Django com Django REST Framework (DRF)** e **FastAPI**. Ambos oferecem soluções completas para a construção de aplicações web, mas com abordagens, facilidades e pontos fortes diferentes.

4.1. Arquitetura e Organização

- **Django/DRF**: Segue o padrão **MTV** (Model-Template-View), com uma estrutura fortemente integrada e orientada a aplicações web completas. Possui um ecossistema robusto que cobre ORM, rotas, views, autenticação, admin, etc.

- **FastAPI**: É mais **modular e minimalista**, com foco exclusivo em APIs. A separação entre lógica de negócio, rotas, modelos e validações é feita de forma explícita e altamente customizável.

4.2. Interface Administrativa

- **Django**: Possui uma interface administrativa pronta, útil para testes e gestão de dados durante o desenvolvimento.

- **FastAPI**: **Não possui admin nativo**, sendo necessário implementar ou integrar manualmente qualquer painel administrativo.

4.3. ORM e Banco de Dados

- **Django**: Utiliza seu próprio ORM integrado e sistema de migração.

- **FastAPI**: Usa o **SQLAlchemy**, que é poderoso, mas exige configuração manual e uso de ferramentas externas (como Alembic) para migrações.

4.4. Validação de Dados

- **Django/DRF**: Utiliza **Serializers** para validação e transformação de dados.

- **FastAPI**: Utiliza os **Schemas com Pydantic**, que oferecem validação automática e integrada ao código das rotas.

4.5. Desempenho e Concorrência

- **Django/DRF**: Funciona por padrão com WSGI e chamadas **síncronas**. Suporta chamadas assíncronas com ASGI, mas exige configuração adicional.

- **FastAPI**: É **assíncrono por padrão**, suportando alta concorrência, ideal para aplicações modernas e de alto desempenho.

4.6. Documentação

- **Django**: A documentação da API pode ser adicionada com DRF ou pacotes extras.

- **FastAPI**: Gera **documentação automática e interativa** (Swagger/OpenAPI) diretamente a partir do código.

4.7. Escalabilidade e Extensibilidade

- **Django**: Suporta boa escalabilidade, principalmente em arquitetura monolítica com múltiplos apps.

- **FastAPI**: É ideal para **microserviços** e arquiteturas modulares, com excelente suporte à concorrência.

4.8. Curva de Aprendizado e Comunidade

- **Django/DRF**: Muito utilizado no mercado, com grande comunidade e vasta documentação.

- **FastAPI**: Comunidade menor, mas em rápida expansão. A curva de aprendizado é baixa para quem já domina Python moderno (tipagens, async, etc).

4.9. Tabela Comparativa: Django/DRFvs FastAPI

Aspecto	Django/DRF	FastAPI
Admin	Interface pronta e poderosa	Não possui admin nativo
ORM	Integrado (Django ORM)	SQLAlchemy (externo)
Validação	Serializers do DRF	Pydantic Schemas
Desempenho	Bom, mas síncrono por padrão	Muito alto, assíncrono por padrão
Documentação	Manual ou via DRF	Automática e interativa (Swagger/OpenAPI)
Migrações	Integradas	Externas (Alembic)
Middlewares	Simples de adicionar	Simples de adicionar
Escalabilidade	Bom, mas tradicionalmente WSG	Excelente, nativamente ASGI
Curva de aprendizado	Baixa para quem conhece Python	Muito baixa, sintaxe moderna
Comunidade	Muito grande e madura	Crescente, mas menor

5. Conclusão

O backend em FastAPI oferece uma solução moderna, de alto desempenho, com validação robusta e documentação automática para APIs RESTful. É ideal para projetos que buscam performance, facilidade de integração e desenvolvimento rápido, especialmente em cenários assíncronos ou de alta concorrência. Em comparação ao Django/DRF, FastAPI se destaca pelo desempenho e simplicidade, mas exige mais implementação manual para funcionalidades administrativas e de migração. A escolha entre as tecnologias deve considerar o perfil do projeto, requisitos de desempenho, facilidade de manutenção e experiência da equipe.

Recomendações:
Usar FastAPI quando: Performance é crítica, você precisa de APIs assíncronas, ou quer documentação automática.
Usar Django/DRF quando: Você precisa de uma solução completa com admin, ou tem uma equipe mais familiarizada com Django.