ESPECIFICAÇÃO TÉCNICA DE SOFTWARE

para

PYTHON API REST SWAPI DDD FastAPI

Guia de Explicação e Uso

Preparado por : Luis Fernando Oliveira Belato

5 de fevereiro de 2026

Conteúdo

1 Introdução

1.1 Propósito

O propósito deste documento é detalhar a arquitetura, design e funcionalidades da API REST desenvolvida para consumo e otimização de dados da SWAPI (Star Wars API). Esta aplicação não atua apenas como um proxy, mas implementa uma camada de lógica de negócios baseada em **Domain-Driven Design (DDD)**, oferecendo recursos avançados como cache em memória, resolução de relacionamentos sob demanda e filtros dinâmicos.

A API resolve problemas de latência e redundância de dados ao implementar um *MemoryCache* compartilhado e permite aos clientes (front-end ou outros serviços) solicitar grafos de objetos complexos (ex: um filme com todos os seus personagens e planetas) em uma única requisição.

1.2 Público Alvo

Esta especificação destina-se a desenvolvedores backend, arquitetos de software e QA (Quality Assurance). O documento cobre desde a estrutura de pastas e decisões arquiteturais até a implementação detalhada dos *Query Parameters* e estratégias de teste.

1.3 Escopo do Projeto

O sistema abrange o gerenciamento de leitura de 6 entidades principais do universo Star Wars, além de monitoramento de saúde da aplicação:

- **Films**: Gerenciamento de filmes.
- **People**: Gerenciamento de personagens.
- **Planets**: Gerenciamento de planetas.
- **Species**: Gerenciamento de espécies.
- **Starships**: Gerenciamento de naves espaciais.
- **Vehicles**: Gerenciamento de veículos.
- **Health**: Endpoint para verificação de disponibilidade (‘/health‘).

A aplicação garante isolamento estrito entre camadas, alta cobertura de testes (96%) e performance otimizada via cache.

2 Descrição Geral

2.1 Perspectiva do Produto

A API atua como um *Middleware Inteligente* sobre a SWAPI pública. Diferente de um consumo direto, esta API normaliza os dados, trata erros de indisponibilidade externa e, crucialmente, armazena respostas em cache para evitar tráfego de rede desnecessário.

2.2 Arquitetura e Design (DDD)

O projeto segue o padrão **Domain-Driven Design (DDD)**, garantindo que camadas superiores não acessem camadas inferiores diretamente.

2.2.1 Estrutura de Pastas e Camadas

- **interfaces**: Camada de Apresentação. Contém os *Controllers* (Rotas FastAPI), *DTOs* (Data Transfer Objects) e a lógica de *Query Parameters*.
- **application**: Contém a lógica de negócio e orquestração. Aqui residem os *Services* que herdam de 'BaseSwapiService'.
- **domain**: O coração da aplicação. Contém as Entidades ('FilmEntity', 'PeopleEntity', etc.) que representam o modelo de negócio puro, sem dependências de frameworks externos.
- **infrastructure**: Implementações técnicas ou camada de dados. Inclui a implementação do *MemoryCache*.
- **config**: Configurações globais da aplicação (variáveis de ambiente, constantes).
- **tests**: Suíte de testes automatizados com 'pytest'.

2.3 Diagrama Geral de Infraestrutura

Abaixo é apresentado o diagrama que ilustra a visão geral da arquitetura da solução, destacando a interação entre as camadas e a infraestrutura de suporte.

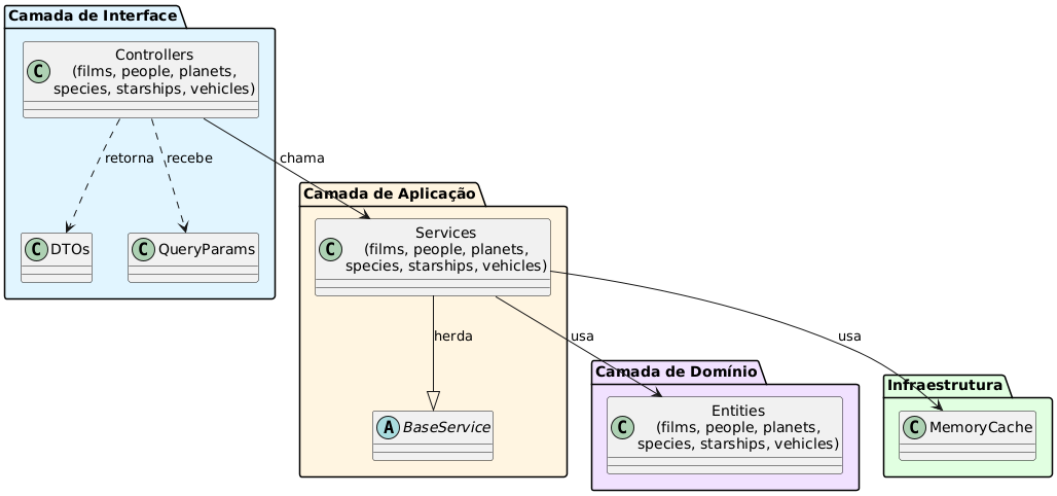

Figura 2.1: Visão Geral da Arquitetura e Infraestrutura

2.4 Fluxo de Requisição

Abaixo é apresentado o diagrama de sequência que ilustra o fluxo de uma requisição para o endpoint de filmes (‘/films‘), demonstrando a interação entre Controller, Service, Cache e a API Externa.

O fluxo segue a lógica: 1. O Cliente faz a requisição. 2. O Controller valida os *Query Parameters*. 3. O Service verifica se o dado existe no *MemoryCache*. 4. Se **Cache Hit**: Retorna imediatamente. 5. Se **Cache Miss**: Consulta a SWAPI externa, serializa para Entidade de Domínio, salva no Cache e retorna.

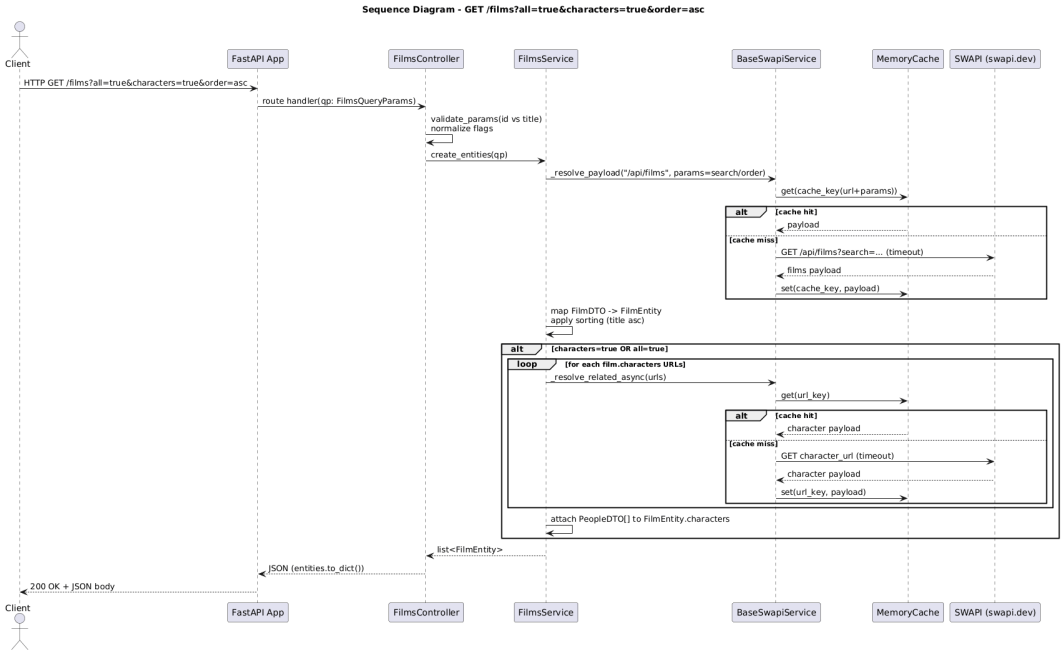

Figura 2.2: Fluxo de Sequência: Requisição GET /films

2.5 Ambiente de Operação

A aplicação é construída em Python (FastAPI) e é agnóstica de sistema operacional, podendo rodar em contêineres Docker (Linux) ou nativamente em Windows/macOS.

3 Funcionalidades do Sistema

3.1 Endpoints REST

A API expõe 7 endpoints principais. Cada endpoint de entidade suporta operações de listagem e busca detalhada por ID.

- ‘GET /films/‘: Retorna lista ou detalhes de filmes.
- ‘GET /people/‘: Retorna lista ou detalhes de personagens.
- ‘GET /planets/‘: Retorna lista ou detalhes de planetas.
- ‘GET /species/‘: Retorna lista ou detalhes de espécies.
- ‘GET /starships/‘: Retorna lista ou detalhes de naves.
- ‘GET /vehicles/‘: Retorna lista ou detalhes de veículos.
- ‘GET /health‘: Retorna status ‘200 OK‘ se a API estiver operacional.

3.2 Query Parameters e Relacionamentos

Esta é uma das funcionalidades centrais da API. Os parâmetros de consulta permitem filtrar resultados e, mais importante, resolver relacionamentos automaticamente.

3.2.1 Estrutura dos Parâmetros

Os parâmetros são divididos em três categorias:

1. **Filtros Nativos (Search):** Parâmetros que filtram a busca na fonte de dados (ex: ‘name‘, ‘title‘, ‘model‘). Variam conforme a entidade.
2. **Parâmetros de Controle:** Comuns a todas as entidades (ex: ‘id‘, ‘page‘, ‘order‘).
3. **Parâmetros de Relacionamento (Boolean flags):** Indicam se a API deve buscar os dados das URLs contidas na resposta (ex: ‘films=true‘ em uma rota de personagem traz os dados completos dos filmes, não apenas a URL).

3.2.2 Implementação Técnica (Dataclasses)

Utiliza-se ‘dataclasses‘ com um método ‘__post_init__‘ para converter os descritores do FastAPI (‘Query‘) em valores reais acessíveis pela camada de serviço. O parâmetro ‘all=true‘ é um atalho para ativar todos os relacionamentos disponíveis.

Exemplo da estrutura para *Starships*:

```python
@dataclass
class StarshipsQueryParams:
    # Parametros nativos da api SWAPI (Search)
    id: Optional[int] = Query(None, description="ID da nave")
    name: Optional[str] = Query(None, description="Busca por nome")
    model: Optional[str] = Query(None, description="Busca por modelo")

    # Parametros de Relacionamento (Lazy Loading)
    films: Optional[bool] = Query(None, description="Inclui filmes relacionados")
    pilots: Optional[bool] = Query(None, description="Inclui pilotos relacionados")

    # Controle Global
    all: Optional[bool] = Query(None, description="Inclui todos os relacionamentos")
    order: Optional[str] = Query(None, description="Ordenacao (ASC/DESC)")
```

Listing 3.1: Exemplo de Query Params para Starships

3.3 Sistema de Cache (MemoryCache)

Para otimizar a performance e respeitar os limites da API externa, foi implementado um sistema de cache em memória na camada de infraestrutura.

- **Estratégia:** Armazena o mapeamento ‘URL (chave) → Objeto JSON (valor)‘.
- **Compartilhamento:** O cache é singleton e compartilhado entre todos os services através da herança da classe ‘BaseSwapiService‘.
- **Reuso em Relacionamentos:** Se um filme for carregado na rota ‘/films‘, e posteriormente solicitado como relacionamento na rota ‘/people‘, o sistema usará a versão em cache, evitando nova chamada HTTP.

4 Requisitos Não Funcionais e Qualidade

4.1 Testes e Cobertura

A qualidade do software é garantida por uma suíte de testes robusta utilizando ‘pytest‘.

- **Cobertura:** 96% do código fonte.
- **Estratégia:**
  - *Unitários:* Testes isolados de Services e Entidades.
  - *Integração:* Testes dos Endpoints verificando códigos de status HTTP e estrutura do JSON.
  - *Mocks:* Uso de ‘unittest.mock‘ para simular as respostas da SWAPI externa, garantindo que os testes não dependam de internet ou da disponibilidade de terceiros.
- **Fixtures:** Implementação de ‘clear_cache‘ para garantir isolamento entre testes que utilizam o MemoryCache.

4.2 Performance

A aplicação foi desenhada para reduzir latência. A conversão de objetos é feita via DTOs otimizados e o cache em memória elimina o *round-trip* time de requisições repetidas.

5 Exemplos de Uso e Parâmetros

Esta seção detalha os 6 endpoints principais da API. Todos os endpoints compartilham a URL base:

`https://star-wars-gateway-6sjbufg5.uc.gateway.dev/`

Para facilitar a leitura e execução, os exemplos abaixo utilizam a quebra de linha do terminal (\). Ao copiar, certifique-se de copiar o bloco inteiro.

5.1 Filmes (/films/)

Gerenciamento de filmes da saga Star Wars.

Parâmetros Suportados:

Parâmetro	Tipo	Descrição
`id`	int	ID do filme na SWAPI.
`title`	string	Busca pelo título do filme.
`characters`	bool	Inclui lista completa de personagens.
`planets`	bool	Inclui lista completa de planetas.
`starships`	bool	Inclui lista completa de naves.
`vehicles`	bool	Inclui lista completa de veículos.
`species`	bool	Inclui lista completa de espécies.
`all`	bool	Carrega todos os relacionamentos.
`order`	string	Ordenação: "ASC"ou "DESC".

Exemplo de Requisição:

```
curl -X GET \
"https://star-wars-gateway-6sjbufg5.uc.gateway.dev//films/?id=1&characters=true&planets=
    true" \
-H "accept: application/json"
```

5.2 Personagens (/people/)

Gerenciamento de personagens (People).

Parâmetros Suportados:

Parâmetro	Tipo	Descrição
`id`	int	ID do personagem na SWAPI.
`name`	string	Busca pelo nome do personagem.
`films`	bool	Inclui lista completa de filmes.
`species`	bool	Inclui lista completa de espécies.
`starships`	bool	Inclui lista completa de naves.
`vehicles`	bool	Inclui lista completa de veículos.
`all`	bool	Carrega todos os relacionamentos.
`order`	string	Ordenação: "ASC"ou "DESC".

Exemplo de Requisição:

```
curl -X GET \
"https://star-wars-gateway-6sjbufg5.uc.gateway.dev//people/?name=Luke&order=asc" \
-H "accept: application/json"
```

5.3 Planetas (/planets/)

Gerenciamento de planetas.

Parâmetros Suportados:

Parâmetro	Tipo	Descrição
`id`	int	ID do planeta na SWAPI.
`name`	string	Busca pelo nome do planeta.
`residents`	bool	Inclui lista completa de residentes.
`films`	bool	Inclui lista completa de filmes.
`all`	bool	Carrega todos os relacionamentos.
`order`	string	Ordenação: "ASC"ou "DESC".

Exemplo de Requisição:

```
curl -X GET \
"https://star-wars-gateway-6sjbufg5.uc.gateway.dev//planets/?name=Tatooine&residents=
    true" \
-H "accept: application/json"
```

5.4 Espécies (/species/)

Gerenciamento de espécies alienígenas.

Parâmetros Suportados:

Parâmetro	Tipo	Descrição
`id`	int	ID da espécie na SWAPI.
`name`	string	Busca pelo nome da espécie.
`people`	bool	Inclui lista de pessoas desta espécie.
`films`	bool	Inclui lista de filmes com esta espécie.
`all`	bool	Carrega todos os relacionamentos.
`order`	string	Ordenação: "ASC"ou "DESC".

Exemplo de Requisição:

```
curl -X GET \
"https://star-wars-gateway-6sjbufg5.uc.gateway.dev//species/?name=Wookie&all=true" \
-H "accept: application/json"
```

5.5 Naves Espaciais (/starships/)

Gerenciamento de naves espaciais.

Parâmetros Suportados:

Parâmetro	Tipo	Descrição
`id`	int	ID da nave na SWAPI.
`name`	string	Busca pelo nome da nave.
`model`	string	Busca pelo modelo da nave.
`films`	bool	Inclui lista de filmes.
`pilots`	bool	Inclui lista de pilotos.
`all`	bool	Carrega todos os relacionamentos.
`order`	string	Ordenação: "ASC"ou "DESC".

Exemplo de Requisição:

```
curl -X GET \
"https://star-wars-gateway-6sjbufg5.uc.gateway.dev//starships/?model=Star%20Destroyer" \
-H "accept: application/json"
```

5.6 Veículos (/vehicles/)

Gerenciamento de veículos.

Parâmetros Suportados:

Parâmetro	Tipo	Descrição
`id`	int	ID do veículo na SWAPI.
`name`	string	Busca pelo nome do veículo.
`model`	string	Busca pelo modelo do veículo.
`films`	bool	Inclui lista de filmes.
`pilots`	bool	Inclui lista de pilotos.
`all`	bool	Carrega todos os relacionamentos.
`order`	string	Ordenação: "ASC"ou "DESC".

Exemplo de Requisição:

```
curl -X GET \
"https://star-wars-gateway-6sjbufg5.uc.gateway.dev//vehicles/?id=4&pilots=true" \
-H "accept: application/json"
```

6 Infraestrutura e Deploy na Google Cloud

A aplicação foi implantada na Google Cloud Platform (GCP) utilizando uma arquitetura serverless para garantir escalabilidade, segurança e eficiência de custos. Abaixo são apresentadas as evidências da configuração dos serviços principais em execução.

6.1 Cloud Run (Serviço de Contêineres)

O backend da API está hospedado no Google Cloud Run, que gerencia o ciclo de vida do contêiner Docker da aplicação. A imagem abaixo demonstra o serviço ativo e operante.

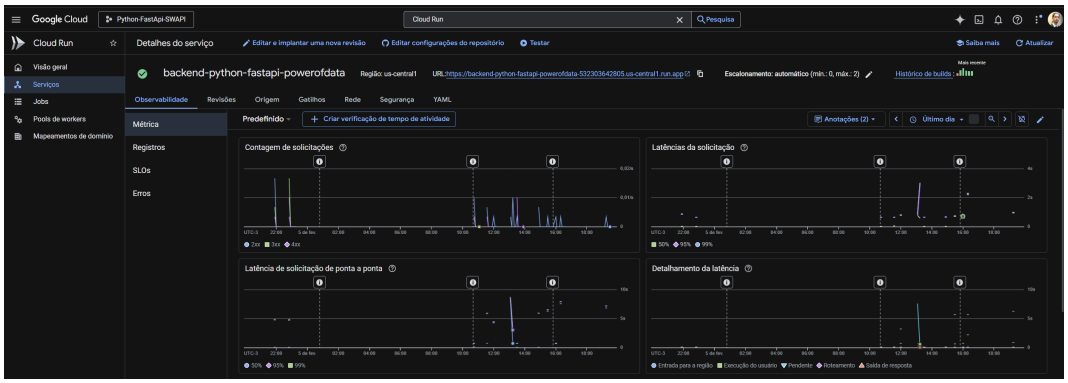

Figura 6.1: Console do Google Cloud Run - Status do Serviço

6.2 API Gateway (Gerenciamento de Acesso)

O acesso público à API é gerenciado pelo Google API Gateway, que serve como porta de entrada segura, roteando as requisições externas para o serviço do Cloud Run.

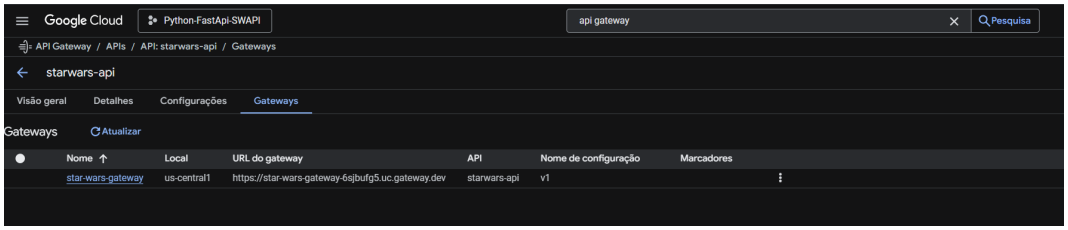

Figura 6.2: Console do Google API Gateway - Configuração do Gateway