Resumo - Backend FastAPI

Visão Geral

Sistema backend desenvolvido em **FastAPI** para integração com múltiplas APIs governamentais brasileiras, oferecendo busca unificada, cache inteligente e documentação automática.

Tecnologias Principais

- **FastAPI 0.104+** - Framework web moderno e rápido
- **Redis 7.0+** - Sistema de cache em memória
- **Uvicorn** - Servidor ASGI de alta performance
- **Pydantic** - Validação automática de dados
- **httpx** - Cliente HTTP assíncrono

Arquitetura

Estrutura de Diretórios

```
backend/
├── app/
│   ├── main.py              # Aplicação principal
│   ├── config.py            # Configurações
│   ├── api/v1/              # Endpoints da API
│   ├── services/            # Integração com APIs externas
│   ├── models/              # Modelos Pydantic
│   ├── core/                # Cache, logging, segurança
│   └── utils/               # Utilitários
├── requirements.txt
└── Dockerfile
```

APIs Integradas

1. **Portal da Transparência** - 80+ endpoints (despesas, servidores, licitações)
2. **Brasil API** - CEP, CNPJ, bancos, municípios
3. **Câmara dos Deputados** - Deputados, proposições, votações
4. **Senado Federal** - Senadores, matérias legislativas
5. **IBGE** - Dados geográficos e demográficos
6. **Banco Central** - Cotações e dados financeiros

Funcionalidades Principais

1. Busca Unificada

```python
@router.post("/search")
async def unified_search(request: SearchRequest):
    # Busca em múltiplas APIs simultaneamente
    # Agregação e ranking de resultados
    # Cache inteligente de respostas
```

2. Sistema de Cache

- **Redis** para cache de consultas
- **TTL configurável** por tipo de dados
- **Invalidação automática** de cache expirado
- **Compressão** de dados para otimização

3. Rate Limiting

- **100 requisições/minuto** por IP
- **Burst limit** de 100 requisições
- **Proteção** contra abuso de APIs
- **Monitoramento** de uso em tempo real

4. Validação Automática

```python
class SearchRequest(BaseModel):
    query: str = Field(..., min_length=1, max_length=500)
    apis: Optional[List[str]] = None
    categories: Optional[List[str]] = None
    page: int = Field(default=1, ge=1)
    limit: int = Field(default=20, ge=1, le=100)
```

Endpoints Principais

Endpoint	Método	Descrição
`/api/v1/search`	POST	Busca unificada em todas as APIs
`/api/v1/apis`	GET	Lista APIs disponíveis e status
`/api/v1/categories`	GET	Lista categorias de dados
`/health`	GET	Health check do sistema
`/docs`	GET	Documentação Swagger automática

Modelos de Dados

Requisição de Busca

```json
{
  "query": "despesas educação",
  "apis": ["portal_transparencia", "camara_deputados"],
  "categories": ["despesas", "educacao"],
  "date_start": "2024-01-01",
  "date_end": "2024-12-31",
  "page": 1,
  "limit": 20
}
```

Resposta de Busca

```
{
  "results": [
    {
      "id": "unique_id",
      "source": "Portal da Transparência",
      "category": "despesas",
      "title": "Despesas com Educação",
      "description": "Gastos do MEC em 2024",
      "data": {...},
      "relevance": 0.95,
      "timestamp": "2024-09-11T10:00:00Z"
    }
  ],
  "total": 1250,
  "page": 1,
  "has_more": true
}
```

Segurança e Monitoramento

Medidas de Segurança

- **Rate limiting** por IP
- **Validação rigorosa** de entrada
- **Sanitização** de dados
- **Headers de segurança** (CORS, CSP)
- **Logs estruturados** para auditoria

Monitoramento

- **Health checks** automáticos
- **Métricas** de performance
- **Alertas** de indisponibilidade
- **Logs centralizados** com estrutlog

Performance

Otimizações

- **Programação assíncrona** nativa
- **Connection pooling** para APIs externas
- **Cache em múltiplas camadas**
- **Compressão gzip** automática
- **Paginação eficiente** de resultados

Métricas Esperadas

- **Latência**: < 200ms para consultas em cache
- **Throughput**: 1000+ req/s com cache
- **Disponibilidade**: 99.9% uptime
- **Cache hit ratio**: > 80%

Deployment

Docker

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt
COPY . .
CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0"]
```

Variáveis de Ambiente

```
PORTAL_TRANSPARENCIA_API_KEY=sua_chave_aqui
REDIS_URL=redis://localhost:6379
LOG_LEVEL=INFO
RATE_LIMIT_PER_MINUTE=100
```

Vantagens do FastAPI

1. **Performance Superior** - Baseado em Starlette, muito rápido
2. **Documentação Automática** - Swagger UI e ReDoc gerados automaticamente
3. **Validação Nativa** - Pydantic para validação de tipos
4. **Async/Await** - Suporte nativo a programação assíncrona
5. **Type Hints** - Excelente suporte a tipagem Python
6. **Padrões Modernos** - OpenAPI 3.0 e JSON Schema
7. **IDE Support** - Autocompletar e verificação de tipos

Próximos Passos

1. Implementar autenticação JWT
2. Adicionar métricas Prometheus
3. Implementar circuit breaker
4. Adicionar testes de carga
5. Configurar CI/CD pipeline