# Sistema de Governança de Dados V1.1 - Instruções dos Pacotes

## Dois Pacotes Disponíveis

### 1. PACOTE DE PRODUÇÃO (706KB)

**Arquivo**: `sistema-governanca-producao-v1.1.tar.gz`

**Conteúdo**: Apenas arquivos essenciais para funcionamento 100% da aplicação

- 31 microserviços funcionais (apps/)
- Configuração centralizada (config/)
- Modelo DBML completo (database/modelo_estendido.dbml)
- Scripts SQL de criação e população (database/scripts/)
- Executor principal (main.py)
- Dependências Python 3.13 (requirements.txt)
- Configurações de ambiente (.env, .env.example)
- README de produção

**Uso**: Implementação em produção, desenvolvimento, testes

### 2. PACOTE DE DOCUMENTAÇÃO (80MB)

**Arquivo**: `sistema-governanca-documentacao-v1.1.tar.gz`

**Conteúdo**: Documentação, apresentações e ferramentas auxiliares

- Documentação técnica, funcional e gerencial completa
- Apresentações para diferentes públicos (gerencial, funcional, técnico)
- Diagramas de arquitetura e fluxos (Draw.io)
- Evidências de teste e validação
- Scripts de desenvolvimento e validação
- Dados mock para teste
- Relatórios de análise

**Uso**: Documentação, treinamento, apresentações, desenvolvimento

# Instalação do Pacote de Produção

## 1. Extrair Pacote

Bash

```
tar -xzf sistema-governanca-producao-v1.1.tar.gz
cd GOVERNANCA_PRODUCAO
```

## 2. Pré-requisitos

Bash

```
# Python 3.13
python3.13 --version

# PostgreSQL
sudo apt install postgresql postgresql-contrib
sudo systemctl start postgresql

# Redis
sudo apt install redis-server
sudo systemctl start redis-server
```

## 3. Configurar Banco de Dados

Bash

```
# Criar usuário e banco
sudo -u postgres createuser governance_user
sudo -u postgres createdb governance_system
sudo -u postgres psql -c "ALTER USER governance_user WITH PASSWORD
'governance_pass';"
sudo -u postgres psql -c "GRANT ALL PRIVILEGES ON DATABASE governance_system
TO governance_user;"

# Executar scripts de criação
cd database/scripts
./run_scripts.sh
cd ../..
```

## 4. Instalar Dependências

Bash

```bash
# Instalar dependências Python
pip3.13 install -r requirements.txt

# Configurar ambiente
cp .env.example .env
# Editar .env com suas configurações específicas
```

### 5. Executar Sistema

Bash

```bash
# Iniciar todos os microserviços
python3.13 main.py

# Verificar funcionamento
curl http://localhost:8000/health
open http://localhost:8000/docs
```

## Estrutura do Pacote de Produção

Plain Text

```
GOVERNANCA_PRODUCAO/
├── apps/                          # 31 microserviços
│   ├── api-gateway/               # Gateway principal (8000)
│   ├── identity-service/          # Autenticação (8001)
│   ├── contract-service/          # Contratos de dados (8003)
│   ├── catalog-service/           # Catálogo (8004)
│   ├── quality-service/           # Qualidade (8005)
│   └── ... (26 outros serviços)   # Portas 8006-8030
├── config/
│   └── settings.py                # Configuração centralizada
├── database/
│   ├── modelo_estendido.dbml      # Modelo completo (43 tabelas)
│   └── scripts/                   # Scripts SQL
│       ├── 01_create_database.sql
│       ├── 02_sample_data.sql
│       ├── 03_migration.sql
│       └── run_scripts.sh
├── main.py                        # Executor principal
├── requirements.txt               # Dependências Python 3.13
```

```
├── .env.example                 # Configurações exemplo
└── README_PRODUCAO.md           # Instruções de produção
```

# Funcionalidades Incluídas

## Microserviços (31)

- API Gateway, Identity, Audit, Contract
- Catalog, Quality, Governance, Lineage
- Metadata, Discovery, Validation, Notification
- Workflow, Security, Privacy, Compliance
- Integration, Analytics, Monitoring, Backup
- Versioning, Tag Management, Data Masking
- Transformation, Scheduler, Config, Reporting
- External Integration, Stewardship, Performance
- Message Queue

## Endpoints (1042)

- APIs REST completas
- Documentação OpenAPI automática
- Autenticação JWT
- Rate limiting
- Versionamento

## Modelo de Dados (43 Tabelas)

- Campos de auditoria (created_at, updated_at)
- Tipos otimizados (text, timestamptz)
- 10 grupos funcionais organizados
- Comentários detalhados no DBML

## Integrações Nativas

- Databricks Unity Catalog
- Informatica Axon

- Azure Active Directory
- PostgreSQL + Redis
- Azure Service Bus + Event Hubs

# Configuração de Produção

## Arquivo .env

Plain Text

```
# Banco de Dados
DATABASE_URL=postgresql://governance_user:governance_pass@localhost:5432/gove
rnance_system
REDIS_URL=redis://localhost:6379/0

# Segurança
SECRET_KEY=your-production-secret-key
JWT_SECRET=your-production-jwt-secret

# Azure (opcional)
AZURE_CLIENT_ID=your-azure-client-id
AZURE_CLIENT_SECRET=your-azure-client-secret
AZURE_TENANT_ID=your-azure-tenant-id

# Databricks (opcional)
DATABRICKS_HOST=your-databricks-host
DATABRICKS_TOKEN=your-databricks-token

# Informatica Axon (opcional)
AXON_HOST=your-axon-host
AXON_USERNAME=your-axon-username
AXON_PASSWORD=your-axon-password
```

## Portas dos Microserviços

- 8000: API Gateway (principal)
- 8001: Identity Service
- 8002: Audit Service
- 8003: Contract Service
- 8004: Catalog Service
- 8005: Quality Service

- 8006-8030: Demais serviços especializados

# Verificação da Instalação

## Health Check

Bash

```bash
# Verificar API Gateway
curl http://localhost:8000/health

# Verificar documentação
curl http://localhost:8000/docs

# Verificar microserviços específicos
curl http://localhost:8001/health  # Identity
curl http://localhost:8003/health  # Contract
curl http://localhost:8005/health  # Quality
```

## Logs

Bash

```bash
# Logs do sistema
tail -f logs/system.log

# Logs de microserviços
tail -f logs/microservices.log
```

# Uso do Pacote de Documentação

## Extrair Documentação

Bash

```bash
tar -xzf sistema-governanca-documentacao-v1.1.tar.gz
cd GOVERNANCA_DOCUMENTACAO
```

## Conteúdo Disponível

- **docs/**: Documentação técnica, funcional e gerencial

- **CAMADA_GOVERNANCA_DADOS/**: Apresentações executivas
- **fluxos_drawio/**: Diagramas para Draw.io
- **evidencias_teste/**: Relatórios de teste
- **mock_data/**: Dados de exemplo
- **Scripts auxiliares**: Ferramentas de desenvolvimento

# Suporte

## Requisitos Mínimos

- **CPU**: 4 cores
- **RAM**: 8GB (16GB recomendado)
- **Disco**: 20GB livres
- **Python**: 3.13 (versão exata)
- **PostgreSQL**: 13+
- **Redis**: 6+

## Troubleshooting

1. Verificar versão Python: `python3.13 --version`
2. Verificar serviços: `systemctl status postgresql redis-server`
3. Verificar logs: `tail -f logs/*.log`
4. Verificar portas: `netstat -tulpn | grep :8000`

## Próximos Passos

1. Implementação Fase 1: Contratos + Unity Catalog
2. Configuração de monitoramento
3. Treinamento da equipe
4. Evolução para Event-Driven Architecture

---

**Sistema de Governança de Dados V1.1**

*Validado para Python 3.13*

*Pronto para produção imediata*