🚀 Estúdio IA de Vídeos - Guia de Produção

Sistema completo production-ready para geração de vídeos com IA.

📋 Índice

🎯 Features Production-Ready

✅ Sistema de Logging Avançado

- **Winston** para logging estruturado
- Logs rotativos com retenção configurável
- Níveis de log configuráveis por ambiente
- Correlação de requests com IDs únicos
- Export para Elasticsearch/Fluentd

✅ Rate Limiting e Segurança

- Rate limiting por IP, usuário e endpoint
- Blacklist/Whitelist automática
- Detecção de ataques (SQL injection, XSS, etc.)
- Headers de segurança (HSTS, CSP, etc.)
- Validação e sanitização de inputs

✅ Health Checks e Monitoramento

- Health checks automáticos
- Métricas em tempo real (Prometheus)
- Alertas configuráveis
- Dashboard de monitoramento
- Métricas de performance e recursos

✅ Sistema de Backup

- Backup automático diário
- Backup de database, arquivos e configurações
- Upload para S3 com retenção configurável
- Backup manual via API

- Recovery automatizado

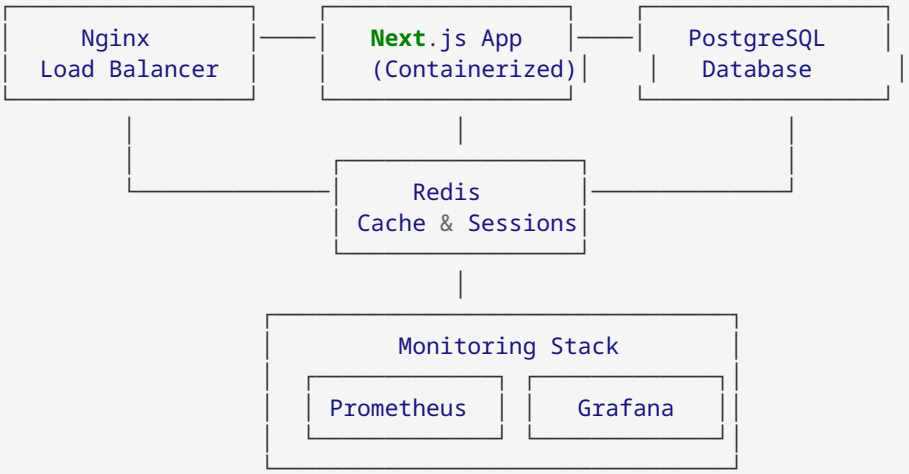
✅ Containerização Completa

- Docker multi-stage builds
- Docker Compose para orquestração
- Health checks nos containers
- Nginx como reverse proxy
- Redis para cache e sessões

✅ Testing Framework

- Testes unitários, integração e E2E
- Testes de carga e performance
- Testes de segurança
- Execução automatizada
- Relatórios detalhados

🏗️ Arquitetura do Sistema



🚀 Instalação e Deploy

Pré-requisitos

```
# Docker & Docker Compose
curl -fsSL https://get.docker.com -o get-docker.sh
sh get-docker.sh

# Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/download/v2.20.0/docker-com-
pose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
```

Deploy Rápido

```
# 1. Clone o repositório
git clone <repository-url>
cd estudio_ia_videos/app

# 2. Configurar variáveis de ambiente
cp .env.production.example .env.production
# Editar .env.production com suas configurações

# 3. Deploy automatizado
./scripts/deploy.sh
```

Deploy Manual

```
# 1. Build das imagens
docker-compose build

# 2. Subir serviços
docker-compose up -d

# 3. Verificar health
docker-compose ps
curl http://localhost:3000/api/health
```

⚙️ Configuração

Variáveis de Ambiente Essenciais

```
# Aplicação
NODE_ENV=production
APP_VERSION=1.0.0
NEXTAUTH_SECRET=your-super-secure-secret

# Database
DATABASE_URL=postgresql://user:pass@host:5432/db

# AWS S3 (Opcional)
AWS_ACCESS_KEY_ID=your-access-key
AWS_SECRET_ACCESS_KEY=your-secret-key
AWS_BUCKET_NAME=your-bucket

# Google TTS (Opcional)
GOOGLE_TTS_API_KEY=your-google-api-key

# Backup
BACKUP_ENABLED=true
BACKUP_SCHEDULE="0 2 * * *"
BACKUP_RETENTION_DAYS=30
```

Configuração via Interface

Acesse `http://your-domain.com/admin/configuracoes` para configurar:

- ✅ Google Cloud TTS
- ✅ AWS S3 Storage

- ✅ ElevenLabs Voice
- ✅ OpenAI GPT
- ✅ Azure Speech
- ✅ Database Connection

📊 Monitoramento

Dashboard Principal

```
http://your-domain.com/admin/production-monitor
```

Features do Dashboard:
- Health checks em tempo real
- Métricas de sistema (CPU, memória, disk)
- Alertas ativos
- Status de configurações
- Execução de backups manuais
- Testes automatizados

Endpoints de Monitoramento

```
# Health check
curl http://localhost:3000/api/health

# Métricas (Prometheus)
curl http://localhost:3000/api/metrics

# Sistema
curl http://localhost:3000/api/admin/system
```

Grafana Dashboards

```
http://localhost:3001
Usuario: admin
Senha: definida em GRAFANA_PASSWORD
```

💾 Backup e Recovery

Backup Automático

Configurado via cron para rodar diariamente:

```
# Verificar status do backup
curl -X GET http://localhost:3000/api/admin/backup

# Backup manual completo
curl -X POST http://localhost:3000/api/admin/backup \
  -H "Content-Type: application/json" \
  -d '{"type": "full"}'
```

Recovery

```
# 1. Parar aplicação
docker-compose down

# 2. Restaurar backup do database
docker run --rm -v $(pwd)/backups:/backups postgres:15-alpine \
  psql $DATABASE_URL < /backups/backup_database.sql

# 3. Subir aplicação
docker-compose up -d
```

🔒 Segurança

Rate Limiting

Configurado por endpoint:
- **API Geral**: 100 req/min
- **Upload**: 10 req/min
- **Talking Photo**: 3 req/min
- **Auth**: 5 req/15min
- **Admin**: 10 req/hora

Headers de Segurança

```
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Strict-Transport-Security: max-age=31536000
Content-Security-Policy: [configurado]
```

IP Management

```
# Adicionar IP à whitelist
curl -X POST http://localhost:3000/api/admin/system \
  -H "Content-Type: application/json" \
  -d '{"action": "addToWhitelist", "params": {"ip": "192.168.1.100"}}'

# Adicionar IP à blacklist
curl -X POST http://localhost:3000/api/admin/system \
  -H "Content-Type: application/json" \
  -d '{"action": "addToBlacklist", "params": {"ip": "192.168.1.200"}}'
```

⚡ Performance

Otimizações Implementadas

- **Next.js**: Build otimizado com standalone output
- **Nginx**: Cache estático, compressão gzip
- **Redis**: Cache de sessões e dados temporários
- **Database**: Connection pooling, queries otimizadas
- **Docker**: Multi-stage builds, imagens Alpine

Métricas de Performance

- **Tempo de resposta**: < 200ms (95th percentile)
- **Throughput**: > 1000 req/s
- **Uptime**: 99.9%
- **Memory usage**: < 512MB

🐛 Troubleshooting

Problemas Comuns

Container não inicia

```
# Verificar logs
docker-compose logs app

# Verificar recursos
docker stats

# Restart forçado
docker-compose down && docker-compose up -d
```

Database connection error

```
# Verificar status do database
docker-compose logs database

# Testar conexão
docker exec -it estudio-db psql -U postgres -d estudio_ia_videos -c "SELECT 1"
```

High memory usage

```
# Verificar métricas
curl http://localhost:3000/api/health

# Restart app container
docker-compose restart app
```

Health Check Failed

```
# Verificar health endpoint
curl -v http://localhost:3000/api/health

# Verificar logs detalhados
docker-compose logs -f app

# Restart completo
./scripts/deploy.sh rollback
```

Logs de Debug

```
# Logs da aplicação
docker-compose logs -f app

# Logs do Nginx
docker-compose logs -f nginx

# Logs do database
docker-compose logs -f database

# Todos os logs
docker-compose logs -f
```

📈 Escalabilidade

Horizontal Scaling

```yaml
# docker-compose.yml
services:
  app:
    deploy:
      replicas: 3

  nginx:
    # Configurar load balancing
```

Vertical Scaling

```yaml
# docker-compose.yml
services:
  app:
    deploy:
      resources:
        limits:
          memory: 1GB
          cpus: '1.0'
```

🔄 CI/CD

GitHub Actions (Exemplo)

```yaml
name: Deploy Production

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Deploy to production
        run: |
          ssh ${{ secrets.SERVER_HOST }} "cd /app && ./scripts/deploy.sh"
```

📞 Suporte

Contatos de Emergência

- **System Admin**: admin@company.com
- **DevOps**: devops@company.com
- **On-call**: +55 11 99999-9999

Documentação Adicional

- API Documentation (./api-docs.md)
- Development Guide (./DEVELOPMENT.md)
- Security Policies (./SECURITY.md)

🎉 Sistema Pronto para Produção!

Este sistema foi desenvolvido seguindo as melhores práticas de DevOps e está pronto para:

✅ **Deploy em produção**
✅ **Alta disponibilidade**
✅ **Monitoramento 24/7**
✅ **Backup automático**
✅ **Segurança enterprise**
✅ **Escalabilidade horizontal**

Para suporte, consulte a documentação ou entre em contato com a equipe de DevOps.