Guia de Deploy - Sistema de Gestão de Pessoas

Introdução

Este guia fornece instruções completas para deploy do Sistema de Gestão de Pessoas em ambiente de produção. Inclui configurações para diferentes provedores de cloud e estratégias de deploy.

Opções de Deploy

1. Deploy com Vercel (Frontend) + Railway (Backend)

Frontend no Vercel

1. **Preparação**

```
cd hr-management-frontend
npm run build
```

1. **Deploy via CLI**

```
npm install -g vercel
vercel login
vercel --prod
```

1. **Configuração de Variáveis**

```
VITE_API_URL=https://seu-backend.railway.app/api/v1
VITE_WS_URL=https://seu-backend.railway.app
```

Backend no Railway

1. **Conectar Repositório**
2. Acesse railway.app
3. Conecte seu repositório GitHub

4. Selecione o diretório `backend`

5. **Configurar Variáveis**

```
 DATABASE_URL=sua-string-supabase
JWT_SECRET=seu-jwt-secret-producao
NODE_ENV=production
PORT=3000
```

1. **Deploy Automático**
2. Railway fará deploy automático a cada push

2. Deploy com Netlify (Frontend) + Heroku (Backend)

Frontend no Netlify

1. **Build Settings**

```
 # netlify.toml
[build]
  base = "hr-management-frontend/"
  publish = "hr-management-frontend/dist"
  command = "npm run build"

[build.environment]
  NODE_VERSION = "20"

[[redirects]]
  from = "/*"
  to = "/index.html"
  status = 200
```

Backend no Heroku

1. **Preparação**

```
 # Criar Procfile
echo "web: npm run start:prod" > backend/Procfile

# Configurar package.json
{
  "scripts": {
    "heroku-postbuild": "npm run build"
```

```
  }
}
```

1. **Deploy**

```
heroku create seu-app-backend
heroku config:set DATABASE_URL=sua-string-supabase
heroku config:set JWT_SECRET=seu-jwt-secret
git subtree push --prefix backend heroku main
```

3. Deploy com Docker

Docker Compose Completo

```yaml
version: '3.8'

services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
    ports:
      - "3000:3000"
    environment:
      - DATABASE_URL=${DATABASE_URL}
      - JWT_SECRET=${JWT_SECRET}
      - NODE_ENV=production
    depends_on:
      - redis
    restart: unless-stopped

  frontend:
    build:
      context: ./hr-management-frontend
      dockerfile: Dockerfile
    ports:
      - "80:80"
    depends_on:
      - backend
    restart: unless-stopped

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    restart: unless-stopped

  nginx:
```

```
    image: nginx:alpine
    ports:
      - "443:443"
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf
      - ./ssl:/etc/nginx/ssl
    depends_on:
      - frontend
      - backend
    restart: unless-stopped
```

Nginx Configuration

```
 events {
    worker_connections 1024;
}

http {
    upstream backend {
        server backend:3000;
    }

    upstream frontend {
        server frontend:80;
    }

    server {
        listen 443 ssl http2;
        server_name seu-dominio.com;

        ssl_certificate /etc/nginx/ssl/cert.pem;
        ssl_certificate_key /etc/nginx/ssl/key.pem;

        location /api/ {
            proxy_pass http://backend;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection 'upgrade';
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
            proxy_cache_bypass $http_upgrade;
        }

        location /socket.io/ {
            proxy_pass http://backend;
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
```

```
        proxy_set_header Host $host;
    }

    location / {
        proxy_pass http://frontend;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
  }

  server {
    listen 80;
    server_name seu-dominio.com;
    return 301 https://$server_name$request_uri;
  }
}
```

4. Deploy na AWS

Usando AWS ECS

1. **Criar Task Definition**

```
{
 "family": "hr-management",
 "networkMode": "awsvpc",
 "requiresCompatibilities": ["FARGATE"],
 "cpu": "512",
 "memory": "1024",
 "executionRoleArn": "arn:aws:iam::account:role/ecsTaskExecutionRole",
 "containerDefinitions": [
  {
   "name": "backend",
   "image": "seu-repo/hr-backend:latest",
   "portMappings": [
    {
     "containerPort": 3000,
     "protocol": "tcp"
    }
   ],
   "environment": [
    {
     "name": "DATABASE_URL",
     "value": "sua-string-supabase"
    }
   ],
   "logConfiguration": {
    "logDriver": "awslogs",
```

```
      "options": {
        "awslogs-group": "/ecs/hr-management",
        "awslogs-region": "us-east-1",
        "awslogs-stream-prefix": "ecs"
      }
    }
  }
 ]
}
```

1. **CloudFormation Template**

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Description: 'HR Management System Infrastructure'

Resources:
  VPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: 10.0.0.0/16
      EnableDnsHostnames: true
      EnableDnsSupport: true

  ECSCluster:
    Type: AWS::ECS::Cluster
    Properties:
      ClusterName: hr-management-cluster

  LoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Name: hr-management-alb
      Scheme: internet-facing
      Type: application
      Subnets:
        - !Ref PublicSubnet1
        - !Ref PublicSubnet2

  ECSService:
    Type: AWS::ECS::Service
    Properties:
      Cluster: !Ref ECSCluster
      TaskDefinition: !Ref TaskDefinition
      DesiredCount: 2
      LaunchType: FARGATE
      NetworkConfiguration:
        AwsvpcConfiguration:
          SecurityGroups:
            - !Ref ECSSecurityGroup
          Subnets:
```

```
        - !Ref PrivateSubnet1
        - !Ref PrivateSubnet2
```

Configuração do Banco de Dados Supabase

Scripts SQL para Execução

1. Schema Principal

Execute o arquivo `database/supabase_schema.sql` no SQL Editor do Supabase:

```
-- Este arquivo contém:
-- - Criação de todas as tabelas
-- - Índices otimizados
-- - Triggers para timestamps
-- - Views para dashboard
-- - Funções utilitárias
```

2. Dados de Exemplo (Opcional)

Execute o arquivo `database/sample_data.sql` para dados de demonstração:

```
-- Este arquivo contém:
-- - Usuário administrador padrão
-- - Departamentos de exemplo
-- - Cargos básicos
-- - Dados para testes
```

3. Configuração de RLS (Row Level Security)

```
-- Habilitar RLS nas tabelas sensíveis
ALTER TABLE users ENABLE ROW LEVEL SECURITY;
ALTER TABLE user_profiles ENABLE ROW LEVEL SECURITY;
ALTER TABLE performance_evaluations ENABLE ROW LEVEL SECURITY;

-- Políticas de acesso
CREATE POLICY "Users can view own data" ON users
  FOR SELECT USING (auth.uid()::text = id::text);

CREATE POLICY "Managers can view team data" ON users
  FOR SELECT USING (
    EXISTS (
      SELECT 1 FROM users u
      WHERE u.id::text = auth.uid()::text
      AND u.role IN ('admin', 'manager')
```

```
  )
 );
```

Configuração de Autenticação

Configurar Auth no Supabase

1. Vá para Authentication → Settings
2. Configure:
3. **Site URL**: https://seu-dominio.com
4. **Redirect URLs**: https://seu-dominio.com/auth/callback
5. **JWT expiry**: 3600 (1 hora)

Integração com Backend

```
// supabase.service.ts
import { createClient } from '@supabase/supabase-js';

@Injectable()
export class SupabaseService {
  private supabase = createClient(
    process.env.SUPABASE_URL,
    process.env.SUPABASE_ANON_KEY
  );

  async verifyToken(token: string) {
    const { data, error } = await this.supabase.auth.getUser(token);
    if (error) throw new UnauthorizedException('Token inválido');
    return data.user;
  }
}
```

Configuração de Domínio e SSL

1. Configuração de DNS

Registros DNS Necessários

```
 A    @         IP-do-servidor
A    www       IP-do-servidor
CNAME api      seu-backend.railway.app
CNAME www      seu-frontend.vercel.app
```

2. Certificado SSL

Let's Encrypt (Gratuito)

```
# Instalar Certbot
sudo apt install certbot python3-certbot-nginx

# Obter certificado
sudo certbot --nginx -d seu-dominio.com -d www.seu-dominio.com

# Renovação automática
sudo crontab -e
# Adicionar: 0 12 * * * /usr/bin/certbot renew --quiet
```

CloudFlare (Recomendado)

1. Adicione seu domínio ao CloudFlare
2. Configure DNS proxy (nuvem laranja)
3. SSL/TLS → Full (strict)
4. Edge Certificates → Always Use HTTPS

Monitoramento e Logs

1. Configuração de Logs

Winston Logger (Backend)

```
// logger.config.ts
import { WinstonModule } from 'nest-winston';
import * as winston from 'winston';

export const loggerConfig = WinstonModule.createLogger({
  transports: [
    new winston.transports.Console({
      format: winston.format.combine(
        winston.format.timestamp(),
        winston.format.colorize(),
        winston.format.simple()
      ),
    }),
    new winston.transports.File({
      filename: 'logs/error.log',
      level: 'error',
      format: winston.format.combine(
        winston.format.timestamp(),
        winston.format.json()
```

```
      ),
    }),
    new winston.transports.File({
      filename: 'logs/combined.log',
      format: winston.format.combine(
        winston.format.timestamp(),
        winston.format.json()
      ),
    }),
  ],
});
```

2. Health Checks

Endpoint de Health

```
// health.controller.ts
@Controller('health')
export class HealthController {
  constructor(
    private prisma: PrismaService,
    private redis: RedisService
  ) {}

  @Get()
  async check() {
    const checks = await Promise.allSettled([
      this.checkDatabase(),
      this.checkRedis(),
      this.checkExternalServices()
    ]);

    const status = checks.every(check => check.status === 'fulfilled')
      ? 'healthy' : 'unhealthy';

    return {
      status,
      timestamp: new Date().toISOString(),
      checks: {
        database: checks[0].status,
        redis: checks[1].status,
        external: checks[2].status
      }
    };
  }

  private async checkDatabase() {
    await this.prisma.$queryRaw`SELECT 1`;
    return 'healthy';
  }
```

```
  private async checkRedis() {
    await this.redis.ping();
    return 'healthy';
  }

  private async checkExternalServices() {
    // Verificar APIs externas se houver
    return 'healthy';
  }
}
```

3. Métricas com Prometheus

Configuração

```
// metrics.module.ts
import { PrometheusModule } from '@willsoto/nestjs-prometheus';

@Module({
  imports: [
    PrometheusModule.register({
      path: '/metrics',
      defaultMetrics: {
        enabled: true,
      },
    }),
  ],
})
export class MetricsModule {}

// Custom metrics
@Injectable()
export class MetricsService {
  private readonly httpRequestsTotal = new Counter({
    name: 'http_requests_total',
    help: 'Total number of HTTP requests',
    labelNames: ['method', 'route', 'status'],
  });

  incrementHttpRequests(method: string, route: string, status: number) {
    this.httpRequestsTotal.inc({ method, route, status: status.toString() });
  }
}
```

Backup e Recuperação

1. Backup Automatizado

Script de Backup

```
#!/bin/bash
# backup.sh

BACKUP_DIR="/var/backups/hr-system"
DATE=$(date +%Y%m%d_%H%M%S)
SUPABASE_URL="sua-url-supabase"
SUPABASE_KEY="sua-chave-supabase"

# Criar diretório de backup
mkdir -p $BACKUP_DIR

# Backup do banco via Supabase CLI
supabase db dump --db-url $SUPABASE_URL > $BACKUP_DIR/database_$DATE.sql

# Backup de arquivos de configuração
tar -czf $BACKUP_DIR/config_$DATE.tar.gz \
  /path/to/hr-system/.env \
  /path/to/hr-system/docker-compose.yml \
  /etc/nginx/sites-available/hr-system

# Backup de logs
tar -czf $BACKUP_DIR/logs_$DATE.tar.gz /var/log/hr-system/

# Limpeza de backups antigos (manter 30 dias)
find $BACKUP_DIR -name "*.sql" -mtime +30 -delete
find $BACKUP_DIR -name "*.tar.gz" -mtime +30 -delete

# Upload para S3 (opcional)
aws s3 cp $BACKUP_DIR/database_$DATE.sql s3://hr-backups/database/
aws s3 cp $BACKUP_DIR/config_$DATE.tar.gz s3://hr-backups/config/

echo "Backup completed: $DATE"
```

Cron Job

```
# Adicionar ao crontab
0 2 * * * /path/to/backup.sh >> /var/log/backup.log 2>&1
```

2. Plano de Recuperação

Procedimento de Restore

```
 #!/bin/bash
# restore.sh

BACKUP_FILE=$1
SUPABASE_URL="sua-url-supabase"

if [ -z "$BACKUP_FILE" ]; then
  echo "Uso: ./restore.sh backup_file.sql"
  exit 1
fi

echo "Iniciando restauração do backup: $BACKUP_FILE"

# Confirmar operação
read -p "Isso irá sobrescrever o banco atual. Continuar? (y/N): " confirm
if [ "$confirm" != "y" ]; then
  echo "Operação cancelada"
  exit 1
fi

# Restaurar banco
psql $SUPABASE_URL < $BACKUP_FILE

echo "Restauração concluída"
```

Configuração de Produção

1. Variáveis de Ambiente

Backend (.env.production)

```
 # Database
DATABASE_URL="postgresql://postgres:[PASSWORD]@db.[PROJECT].supabase.co:
5432/postgres"

# JWT
JWT_SECRET="jwt-secret-super-seguro-producao-com-64-caracteres-minimo"
JWT_EXPIRES_IN="1h"
JWT_REFRESH_EXPIRES_IN="7d"

# Server
NODE_ENV="production"
PORT=3000
```

```
# CORS
CORS_ORIGIN="https://seu-dominio.com"

# Logs
LOG_LEVEL="info"
LOG_FILE="/var/log/hr-system/app.log"

# Cache
REDIS_URL="redis://localhost:6379"

# Monitoring
SENTRY_DSN="https://your-dsn@sentry.io/project-id"

# External Services
N8N_WEBHOOK_URL="https://n8n.seu-dominio.com/webhook"
SMTP_HOST="smtp.gmail.com"
SMTP_PORT=587
SMTP_USER="noreply@seu-dominio.com"
SMTP_PASS="senha-do-email"

# Features
CHAT_BOT_ENABLED=true
ANALYTICS_ENABLED=true
```

Frontend (.env.production)

```
 # API
VITE_API_URL=https://api.seu-dominio.com/v1
VITE_WS_URL=https://api.seu-dominio.com

# App
VITE_APP_NAME="Sistema de Gestão de Pessoas"
VITE_APP_VERSION="1.0.0"
VITE_APP_DESCRIPTION="Sistema completo de gestão de RH"

# Features
VITE_CHAT_ENABLED=true
VITE_ANALYTICS_ENABLED=true
VITE_SENTRY_ENABLED=true

# Monitoring
VITE_SENTRY_DSN="https://your-dsn@sentry.io/project-id"

# External
VITE_GOOGLE_ANALYTICS_ID="GA-XXXXXXXXX"
```

2. Configuração de Segurança

Helmet.js (Backend)

```
// main.ts
import helmet from 'helmet';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);

  app.use(helmet({
    contentSecurityPolicy: {
      directives: {
        defaultSrc: ["'self'"],
        styleSrc: ["'self'", "'unsafe-inline'"],
        scriptSrc: ["'self'"],
        imgSrc: ["'self'", "data:", "https:"],
      },
    },
    hsts: {
      maxAge: 31536000,
      includeSubDomains: true,
      preload: true
    }
  }));

  await app.listen(3000);
}
```

Rate Limiting

```
// rate-limit.config.ts
import { ThrottlerModule } from '@nestjs/throttler';

@Module({
  imports: [
    ThrottlerModule.forRoot({
      ttl: 60,
      limit: 100,
    }),
  ],
})
export class AppModule {}
```

Checklist de Deploy

Pré-Deploy

- [ ] Testes unitários passando
- [ ] Testes de integração passando
- [ ] Build de produção funcionando
- [ ] Variáveis de ambiente configuradas
- [ ] Banco de dados configurado
- [ ] SSL configurado
- [ ] Domínio apontando corretamente

Deploy

- [ ] Backup do ambiente atual
- [ ] Deploy do backend
- [ ] Deploy do frontend
- [ ] Verificação de health checks
- [ ] Testes de fumaça
- [ ] Verificação de logs

Pós-Deploy

- [ ] Monitoramento ativo
- [ ] Alertas configurados
- [ ] Backup automático funcionando
- [ ] Performance dentro do esperado
- [ ] Usuários conseguem acessar
- [ ] Funcionalidades críticas testadas

Troubleshooting

Problemas Comuns

1. Erro de CORS

```
 // Configuração CORS
app.enableCors({
  origin: process.env.CORS_ORIGIN?.split(',') || ['http://localhost:5173'],
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH'],
```

```
  allowedHeaders: ['Content-Type', 'Authorization'],
});
```

2. Erro de Conexão com Banco

```
# Verificar conectividade
psql "postgresql://postgres:[PASSWORD]@db.[PROJECT].supabase.co:5432/
postgres" -c "SELECT 1;"

# Verificar variáveis
echo $DATABASE_URL
```

3. Erro de Build

```
# Limpar cache
npm cache clean --force
rm -rf node_modules package-lock.json
npm install

# Verificar versão do Node
node --version
npm --version
```

4. Erro de SSL

```
# Verificar certificado
openssl s_client -connect seu-dominio.com:443 -servername seu-dominio.com

# Renovar Let's Encrypt
sudo certbot renew --dry-run
```

Logs Importantes

Localização dos Logs

```
# Application logs
tail -f /var/log/hr-system/app.log

# Nginx logs
tail -f /var/log/nginx/access.log
tail -f /var/log/nginx/error.log

# System logs
journalctl -u hr-backend -f
```

Análise de Performance

```
# CPU e Memória
htop

# Conexões de rede
netstat -tulpn | grep :3000

# Espaço em disco
df -h

# Processos do Node
ps aux | grep node
```

Conclusão

Este guia fornece uma base sólida para deploy do Sistema de Gestão de Pessoas em produção. A escolha da estratégia de deploy depende dos requisitos específicos de infraestrutura, orçamento e expertise da equipe.

Recomendações Finais

1. **Comece simples**: Use Vercel + Railway para MVPs
2. **Escale gradualmente**: Migre para AWS/GCP conforme necessário
3. **Monitore sempre**: Configure alertas desde o início
4. **Faça backups**: Automatize backups diários
5. **Documente tudo**: Mantenha runbooks atualizados

Suporte Contínuo

- Monitore métricas de performance
- Mantenha dependências atualizadas
- Realize testes de carga periodicamente
- Revise logs de segurança regularmente
- Mantenha plano de recuperação de desastres

Guia de Deploy - Sistema de Gestão de Pessoas

Versão 1.0 - Desenvolvido por Manus AI