Descrição da Infraestrutura de Implantação

Sistema de Gestão de Feiras (SGF)

Higor Roger de Freitas Santos 221006440
Victor Eneias Oliveira 221038364

Engenharia de Software CIC0105 Turma 01 2025.1

30 de junho de 2025

Conteúdo

1 Visão Geral

O Sistema de Gestão de Feiras (SGF) foi projetado com uma arquitetura modular que permite implantação tanto em ambiente de desenvolvimento local quanto em produção. Este documento descreve os requisitos de hardware, software e serviços necessários para ambos os cenários.

2 Ambiente de Desenvolvimento

2.1 Hardware - Desenvolvimento

2.1.1 Requisitos Mínimos

- **CPU:** 1 core, 1.0 GHz
- **RAM:** 2 GB
- **Armazenamento:** 5 GB livres
- **Rede:** Conexão com internet para instalação de dependências

2.1.2 Requisitos Recomendados

- **CPU:** 2+ cores, 2.0+ GHz
- **RAM:** 4+ GB
- **Armazenamento:** 10+ GB livres (SSD preferível)
- **Rede:** Banda larga estável

2.2 Software - Desenvolvimento

2.2.1 Sistema Operacional

- Windows 10/11
- Linux (Ubuntu 20.04+, CentOS 8+, Debian 10+)
- macOS 10.15+

2.2.2 Dependências Necessárias

- Python 3.8+ (recomendado 3.9+)
- pip (gerenciador de pacotes Python)
- Navegador web moderno (Chrome 90+, Firefox 88+, Edge 90+)
- Editor de código (opcional: VS Code, PyCharm)

2.2.3 Bibliotecas Python

Conforme arquivo `requirements.txt`:

```
fastapi==0.68.0
uvicorn[standard]==0.15.0
sqlalchemy==1.4.23
pydantic==1.8.2
python-jose[cryptography]==3.3.0
passlib[bcrypt]==1.7.4
python-multipart==0.0.5
```

2.3 Execução - Desenvolvimento

2.3.1 Instalação

1. Clonar o repositório do projeto
2. Instalar Python 3.8+ no sistema
3. Executar: `pip install -r requirements.txt`
4. Executar backend: `uvicorn main:app --reload --port 8000`
5. Servir frontend: `python -m http.server 3000` (pasta frontend)
6. Acessar: `http://localhost:3000`

2.3.2 Estrutura de Arquivos - Desenvolvimento

- `main.py` - Aplicação principal FastAPI
- `frontend/` - Arquivos da interface (HTML/CSS/JS)
- `routers/` - Endpoints da API REST
- `models.py` - Modelos SQLAlchemy
- `requirements.txt` - Dependências Python
- `esw.db` - Banco SQLite (criado automaticamente)

3 Ambiente de Produção

3.1 Hardware - Produção

3.1.1 Servidor de Aplicação

- **CPU:** 4+ cores, 2.4+ GHz (Intel Xeon ou AMD EPYC)
- **RAM:** 8+ GB (16 GB recomendado)
- **Armazenamento:** 50+ GB SSD NVMe
- **Rede:** 1 Gbps, IP público estático
- **Redundância:** Fonte redundante, RAID 1 para dados críticos

3.1.2 Servidor de Banco de Dados (Opcional)

Para alta disponibilidade, separar o banco em servidor dedicado:

- **CPU:** 2+ cores, 2.0+ GHz
- **RAM:** 4+ GB (8 GB recomendado)
- **Armazenamento:** 100+ GB SSD, backup automatizado
- **Rede:** Conexão privada com servidor de aplicação

3.2 Software - Produção

3.2.1 Sistema Operacional

- **Recomendado:** Ubuntu Server 22.04 LTS
- **Alternativas:** CentOS Stream 9, Debian 11, RHEL 9
- **Configurações:** Firewall ativo, atualizações automáticas de segurança

3.2.2 Servidor Web e Proxy Reverso

- **Nginx** (recomendado) ou Apache HTTP Server
- **Função:** Proxy reverso, SSL/TLS, servir arquivos estáticos
- **Configuração:** Rate limiting, compressão gzip

3.2.3 Servidor de Aplicação

- **Gunicorn** com workers Uvicorn para produção
- **Configuração:** Múltiplos workers, auto-restart
- **Monitoramento:** Logs estruturados, métricas de performance

3.2.4 Banco de Dados

Opção 1 - SQLite (Pequena escala):

- Adequado para até 1000 usuários simultâneos
- Backup diário automatizado
- Replicação para servidor secundário

Opção 2 - PostgreSQL (Escala média/alta):

- Adequado para 1000+ usuários simultâneos
- Configuração master-slave para alta disponibilidade
- Backup incremental e point-in-time recovery

3.3 Serviços de Produção

3.3.1 Segurança

- **SSL/TLS:** Certificado válido (Let's Encrypt ou comercial)
- **Firewall:** Apenas portas 80, 443 e 22 (SSH) abertas
- **Fail2Ban:** Proteção contra ataques de força bruta
- **Backup:** Backup diário automatizado com retenção de 30 dias

3.3.2 Monitoramento

- **Logs:** Centralizados via rsyslog ou ELK Stack
- **Métricas:** CPU, RAM, disco, rede via Prometheus/Grafana
- **Uptime:** Monitoramento externo (UptimeRobot, Pingdom)
- **Alertas:** Notificações via email/SMS para problemas críticos

3.3.3 Escalabilidade

- **Load Balancer:** Nginx ou HAProxy para múltiplas instâncias
- **CDN:** CloudFlare ou AWS CloudFront para arquivos estáticos
- **Cache:** Redis para sessões e cache de aplicação
- **Auto-scaling:** Kubernetes ou Docker Swarm (opcional)

4 Configuração de Produção

4.1 Variáveis de Ambiente

```
# Configuracoes de producao
ENVIRONMENT=production
SECRET_KEY=<chave-secreta-forte-256-bits>
DATABASE_URL=postgresql://user:pass@localhost/sgf_prod
ALLOWED_HOSTS=sgf.exemplo.com,www.sgf.exemplo.com
SSL_REDIRECT=true
DEBUG=false
LOG_LEVEL=INFO
```

4.2 Configuração do Nginx

```
server {
    listen 80;
    server_name sgf.exemplo.com www.sgf.exemplo.com;
    return 301 https://$server_name$request_uri;
}
```

```
server {
    listen 443 ssl http2;
    server_name sgf.exemplo.com www.sgf.exemplo.com;

    ssl_certificate /etc/letsencrypt/live/sgf.exemplo.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/sgf.exemplo.com/privkey.
       pem;

    location / {
        proxy_pass http://127.0.0.1:8000;
        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;
    }

    location /static/ {
        alias /var/www/sgf/frontend/;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
```

4.3 Configuração do Gunicorn

```
# gunicorn.conf.py
bind = "127.0.0.1:8000"
workers = 4
worker_class = "uvicorn.workers.UvicornWorker"
worker_connections = 1000
max_requests = 1000
max_requests_jitter = 100
preload_app = True
keepalive = 5
```

5 Procedimento de Deploy

5.1 Deploy Inicial

1. Provisionar servidor com especificações mínimas
2. Instalar e configurar sistema operacional
3. Configurar firewall e segurança básica
4. Instalar Python, Nginx, PostgreSQL (se aplicável)
5. Clonar código da aplicação
6. Configurar variáveis de ambiente
7. Instalar dependências Python
8. Configurar banco de dados e executar migrações

9. Configurar Nginx e SSL
10. Configurar monitoramento e backup
11. Testar funcionamento completo

5.2 Deploy de Atualizações

1. Backup do banco de dados atual
2. Baixar nova versão do código
3. Instalar novas dependências (se houver)
4. Executar migrações de banco (se houver)
5. Reiniciar serviços (Gunicorn, Nginx)
6. Verificar funcionamento
7. Rollback em caso de problemas

6 Estimativas de Custo

6.1 Hospedagem em Cloud (mensal)

- **AWS EC2 t3.medium:** $30-50/mês
- **DigitalOcean Droplet 4GB:** $24/mês
- **Google Cloud e2-standard-2:** $35-60/mês
- **Azure B2s:** $30-45/mês

6.2 Serviços Adicionais (mensal)

- **Certificado SSL:** Gratuito (Let's Encrypt)
- **Backup:** $5-15/mês
- **Monitoramento:** $10-25/mês
- **CDN:** $5-20/mês
- **Total estimado:** $50-150/mês