Guia de Deploy do Sistema de Login SparkWave

Autor: Manus AI
Data: 06/06/2025
Versão: 1.0

Sumário

1. Introdução

Este guia fornece instruções detalhadas para o deploy do sistema de login da SparkWave Consultoria Empresarial em um ambiente de produção. O sistema consiste em um backend Java com Spring Boot e um frontend HTML/CSS/JavaScript. O guia aborda diferentes opções de deploy, desde a instalação manual até o uso de contêineres Docker.

2. Requisitos de Sistema

Para o deploy do sistema de login da SparkWave, você precisará dos seguintes requisitos:

Backend:

- Java Development Kit (JDK) 17 ou superior
- PostgreSQL 12 ou superior (para produção)
- 2GB de RAM (mínimo)
- 10GB de espaço em disco (mínimo)

Frontend:

- Servidor web (Nginx, Apache, etc.)
- 512MB de RAM (mínimo)
- 1GB de espaço em disco (mínimo)

Para deploy com Docker:

- Docker 20.10 ou superior
- Docker Compose 2.0 ou superior
- 4GB de RAM (mínimo)
- 20GB de espaço em disco (mínimo)

3. Deploy do Backend

3.1. Preparação do Ambiente

Primeiro, certifique-se de que o JDK está instalado no servidor:

```
# Verificar a versão do Java
java -version
```

```
# Se não estiver instalado, instale o JDK (exemplo para Ubuntu/
Debian)
sudo apt update
sudo apt install openjdk-17-jdk
```

3.2. Configuração do Banco de Dados

Para o ambiente de produção, recomendamos o uso do PostgreSQL:

```
# Instalar PostgreSQL (exemplo para Ubuntu/Debian)
sudo apt update
sudo apt install postgresql postgresql-contrib

# Iniciar o serviço
sudo systemctl start postgresql
sudo systemctl enable postgresql

# Acessar o PostgreSQL
sudo -u postgres psql

# Criar banco de dados e usuário
CREATE DATABASE sparkwavedb;
CREATE USER sparkwave WITH ENCRYPTED PASSWORD
'sua_senha_segura';
GRANT ALL PRIVILEGES ON DATABASE sparkwavedb TO sparkwave;
\q

# Verificar se o banco de dados foi criado
sudo -u postgres psql -l
```

3.3. Compilação do Backend

Navegue até o diretório do backend e compile o projeto usando Maven:

```
cd /caminho/para/sparkwave-login/backend

# Compilar o projeto
./mvnw clean package -DskipTests

# O arquivo JAR será gerado na pasta target/
ls -la target/
```

Antes de executar o JAR, crie um arquivo `application-prod.properties` na mesma pasta do JAR com as configurações de produção:

```
# Configurações do servidor
server.port=8080
server.servlet.context-path=/api

# Configurações do PostgreSQL
spring.datasource.url=jdbc:postgresql://localhost:5432/
sparkwavedb
spring.datasource.username=sparkwave
spring.datasource.password=sua_senha_segura
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDi
spring.jpa.hibernate.ddl-auto=update

# Configurações de segurança JWT
jwt.secret=sparkwave_secret_key_muito_segura_e_longa_para_garantir_a_segur
jwt.expiration=86400000

# Configurações CORS
sparkwave.app.cors.allowedOrigins=https://seu-dominio.com

# URL de redirecionamento após login
sparkwave.app.redirect.url=https://school-finance-
wizard.lovable.app/calculos
```

3.4. Deploy como Serviço

Para executar o backend como um serviço do sistema:

```
# Criar um arquivo de serviço systemd
sudo nano /etc/systemd/system/sparkwave-login.service
```

Adicione o seguinte conteúdo:

```
[Unit]
Description=SparkWave Login Backend
After=network.target postgresql.service

[Service]
User=seu_usuario
WorkingDirectory=/caminho/para/sparkwave-login/backend
ExecStart=/usr/bin/java -jar target/login-0.0.1-SNAPSHOT.jar --
spring.config.location=file:./application-prod.properties
SuccessExitStatus=143
TimeoutStopSec=10
Restart=on-failure
RestartSec=5
```

```
[Install]
WantedBy=multi-user.target
```

Inicie e habilite o serviço:

```
sudo systemctl daemon-reload
sudo systemctl start sparkwave-login
sudo systemctl enable sparkwave-login
sudo systemctl status sparkwave-login
```

3.5. Deploy com Docker

Alternativamente, você pode usar Docker para o deploy do backend. Crie um arquivo `Dockerfile` no diretório do backend:

```
FROM openjdk:17-jdk-slim

WORKDIR /app

COPY target/login-0.0.1-SNAPSHOT.jar app.jar
COPY application-prod.properties application.properties

EXPOSE 8080

ENTRYPOINT ["java", "-jar", "app.jar", "--
spring.config.location=file:./application.properties"]
```

Construa e execute a imagem Docker:

```
# Construir a imagem
docker build -t sparkwave-login-backend .

# Executar o contêiner
docker run -d -p 8080:8080 --name sparkwave-backend sparkwave-
login-backend
```

4. Deploy do Frontend

4.1. Preparação dos Arquivos

Antes de fazer o deploy do frontend, você precisa atualizar a URL da API no arquivo `script.js`:

```
// Configuração da API
const API_URL = 'https://seu-dominio.com/api';
const AUTH_ENDPOINT = '/auth/signin';
```

4.2. Configuração do Servidor Web

Você pode usar qualquer servidor web para hospedar os arquivos estáticos do frontend. Neste guia, abordaremos o Nginx e o Apache.

4.3. Deploy com Nginx

Instale o Nginx:

```
# Instalar Nginx (exemplo para Ubuntu/Debian)
sudo apt update
sudo apt install nginx

# Iniciar o serviço
sudo systemctl start nginx
sudo systemctl enable nginx
```

Crie um arquivo de configuração para o site:

```
sudo nano /etc/nginx/sites-available/sparkwave-login
```

Adicione o seguinte conteúdo:

```
server {
    listen 80;
    server_name seu-dominio.com www.seu-dominio.com;
    root /var/www/sparkwave-login;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://localhost:8080;
        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;
```

```
    }
}
```

Ative o site e reinicie o Nginx:

```
sudo ln -s /etc/nginx/sites-available/sparkwave-login /etc/
nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

Copie os arquivos do frontend para o diretório do servidor web:

```
sudo mkdir -p /var/www/sparkwave-login
sudo cp -r /caminho/para/sparkwave-login/frontend/* /var/www/
sparkwave-login/
sudo chown -R www-data:www-data /var/www/sparkwave-login
```

4.4. Deploy com Apache

Instale o Apache:

```
# Instalar Apache (exemplo para Ubuntu/Debian)
sudo apt update
sudo apt install apache2

# Iniciar o serviço
sudo systemctl start apache2
sudo systemctl enable apache2
```

Crie um arquivo de configuração para o site:

```
sudo nano /etc/apache2/sites-available/sparkwave-login.conf
```

Adicione o seguinte conteúdo:

```
<VirtualHost *:80>
    ServerName seu-dominio.com
    ServerAlias www.seu-dominio.com
    DocumentRoot /var/www/sparkwave-login

    <Directory /var/www/sparkwave-login>
        Options Indexes FollowSymLinks
        AllowOverride All
        Require all granted
```

```
    </Directory>

    ProxyPreserveHost On
    ProxyPass /api http://localhost:8080/api
    ProxyPassReverse /api http://localhost:8080/api

    ErrorLog ${APACHE_LOG_DIR}/sparkwave-error.log
    CustomLog ${APACHE_LOG_DIR}/sparkwave-access.log combined
</VirtualHost>
```

Ative o site e os módulos necessários:

```
sudo a2enmod proxy proxy_http
sudo a2ensite sparkwave-login.conf
sudo systemctl restart apache2
```

Copie os arquivos do frontend para o diretório do servidor web:

```
sudo mkdir -p /var/www/sparkwave-login
sudo cp -r /caminho/para/sparkwave-login/frontend/* /var/www/
sparkwave-login/
sudo chown -R www-data:www-data /var/www/sparkwave-login
```

4.5. Deploy com Docker

Você também pode usar Docker para o deploy do frontend. Crie um arquivo `Dockerfile` no diretório do frontend:

```
FROM nginx:alpine

COPY . /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 80

CMD ["nginx", "-g", "daemon off;"]
```

Crie um arquivo `nginx.conf`:

```
server {
    listen 80;
    server_name localhost;
    root /usr/share/nginx/html;
    index index.html;
```

```
    location / {
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://backend:8080;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
    }
}
```

Construa e execute a imagem Docker:

```
# Construir a imagem
docker build -t sparkwave-login-frontend .

# Executar o contêiner
docker run -d -p 80:80 --name sparkwave-frontend sparkwave-
login-frontend
```

5. Configuração de HTTPS

Para garantir a segurança do sistema, é altamente recomendável configurar HTTPS.

5.1. Obtenção de Certificado SSL

Você pode obter um certificado SSL gratuito usando o Let's Encrypt:

```
# Instalar Certbot (exemplo para Ubuntu/Debian)
sudo apt update
sudo apt install certbot

# Para Nginx
sudo apt install python3-certbot-nginx

# Para Apache
sudo apt install python3-certbot-apache
```

5.2. Configuração no Nginx

Obtenha e configure o certificado para Nginx:

```
sudo certbot --nginx -d seu-dominio.com -d www.seu-dominio.com
```

O Certbot atualizará automaticamente a configuração do Nginx para usar HTTPS.

5.3. Configuração no Apache

Obtenha e configure o certificado para Apache:

```
sudo certbot --apache -d seu-dominio.com -d www.seu-dominio.com
```

O Certbot atualizará automaticamente a configuração do Apache para usar HTTPS.

6. Monitoramento e Manutenção

6.1. Logs do Sistema

Para monitorar os logs do backend:

```
# Se estiver usando systemd
sudo journalctl -u sparkwave-login -f

# Se estiver usando Docker
docker logs -f sparkwave-backend
```

Para monitorar os logs do servidor web:

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

# Apache
sudo tail -f /var/log/apache2/sparkwave-access.log
sudo tail -f /var/log/apache2/sparkwave-error.log
```

6.2. Backup do Banco de Dados

Crie backups regulares do banco de dados:

```
# Backup do PostgreSQL
pg_dump -U sparkwave -W -F t sparkwavedb > sparkwavedb_backup_$
(date +%Y%m%d).tar

# Restaurar o backup
```

```
pg_restore -U sparkwave -W -d sparkwavedb
sparkwavedb_backup_20250606.tar
```

6.3. Atualização do Sistema

Para atualizar o backend:

1. Faça backup do banco de dados
2. Compile a nova versão do backend
3. Pare o serviço atual
4. Substitua o arquivo JAR
5. Inicie o serviço novamente

```
# Parar o serviço
sudo systemctl stop sparkwave-login

# Substituir o arquivo JAR
cp target/login-0.0.1-SNAPSHOT.jar /caminho/para/deploy/

# Iniciar o serviço
sudo systemctl start sparkwave-login
```

Para atualizar o frontend:

1. Atualize os arquivos no diretório do servidor web
2. Limpe o cache do navegador (se necessário)

```
# Atualizar os arquivos
sudo cp -r /caminho/para/sparkwave-login/frontend/* /var/www/
sparkwave-login/
sudo chown -R www-data:www-data /var/www/sparkwave-login
```

7. Solução de Problemas

Problema: O backend não inicia

Possíveis causas e soluções:

1. **Porta em uso:** ```bash # Verificar se a porta está em uso sudo lsof -i :8080

Encerrar o processo que está usando a porta sudo kill -9 PID ```

1. **Erro de conexão com o banco de dados:** ```bash # Verificar se o PostgreSQL está em execução sudo systemctl status postgresql

Verificar as credenciais no arquivo application-prod.properties ```

1. **Erro de permissão:** ```bash # Verificar as permissões do arquivo JAR ls -la /caminho/para/login-0.0.1-SNAPSHOT.jar

Corrigir as permissões chmod 755 /caminho/para/login-0.0.1-SNAPSHOT.jar ```

Problema: O frontend não carrega

Possíveis causas e soluções:

1. **Erro de configuração do servidor web:** ```bash # Verificar a sintaxe da configuração do Nginx sudo nginx -t

Verificar a sintaxe da configuração do Apache sudo apachectl configtest ```

1. **Erro de permissão:** ```bash # Verificar as permissões dos arquivos do frontend ls -la /var/www/sparkwave-login

Corrigir as permissões sudo chown -R www-data:www-data /var/www/sparkwave-login ```

1. **Erro de CORS:** `bash # Verificar a configuração CORS no backend # Editar application-prod.properties sparkwave.app.cors.allowedOrigins=https://seu-dominio.com`

Problema: Erro de autenticação

Possíveis causas e soluções:

1. **Credenciais incorretas:** `bash # Verificar as credenciais no banco de dados sudo -u postgres psql -d sparkwavedb -c "SELECT username FROM users;"`

2. **Erro de JWT:** `bash # Verificar a configuração JWT no backend # Editar application-prod.properties jwt.secret=sparkwave_secret_key_muito_segura_e_longa_para_garantir_a_se`

8. Referências

1. Documentação do Spring Boot
2. Documentação do Nginx
3. Documentação do Apache
4. Documentação do PostgreSQL
5. Documentação do Docker
6. Let's Encrypt
7. Certbot