Trabalho 1 - Devops

Nataly Cristina da Silva - 812719

Descrição da aplicação:

Visão geral da aplicação
A aplicação é um sistema web desenvolvido em Spring Boot que permite a empresas cadastrar vagas de emprego e profissionais se candidatarem a essas vagas. Ela oferece funcionalidades de autenticação, CRUD de empresas e vagas, e gerenciamento de inscrições. Um de seus diferenciais é a capacidade de notificar profissionais por e-mail sobre o status de suas candidaturas, como a chamada para entrevistas, através de um serviço de e-mail dedicado.
A aplicação foi containerizada usando Docker e gerenciada com Docker Compose, tornando fácil a sua instalação e execução em qualquer ambiente compatível com Docker.

Contêineres

1. Contêiner spring_app (Backend)

- **Funcionalidade:** Este contêiner hospeda a lógica de negócios principal da aplicação, desenvolvida em **Spring Boot (Java 17)**. É responsável por gerenciar as requisições HTTP, interagir com o banco de dados (MySQL) para persistir e recuperar informações sobre empresas, vagas e inscrições, e orquestrar o fluxo de dados. Ele também atua como o **consumidor do serviço de e-mail** para enviar notificações.
- **Importância na Aplicação:** É o *core* da aplicação, fornecendo as APIs e a interface web (via Thymeleaf) que os usuários (empresas e profissionais) interagem. Sem ele, a aplicação não existiria.
- **Tecnologias:** Spring Boot, Java 17, Maven.
- **Dockerfile:**
    - FROM maven:3.8.4-openjdk-17-slim AS build-stage: Utiliza uma imagem Maven com JDK 17 para compilar o código fonte.
    - FROM openjdk:17-jdk-slim AS production-stage: Cria uma imagem final leve com apenas o JRE 17 para executar a aplicação.
    - COPY --from=build-stage /app/target/*.jar app.jar: Copia o arquivo JAR compilado da etapa de build.
    - CMD ["java", "-jar", "app.jar"]: Define o comando para iniciar a aplicação Spring Boot.

2. Contêiner mysql_db (Banco de Dados)

- **Funcionalidade:** Este contêiner provê o **serviço de banco de dados relacional MySQL (versão 8.0)**. Ele é responsável por armazenar todos os

dados da aplicação, como informações de usuários (empresas, profissionais), detalhes das vagas e registros de inscrições.
- **Importância na Aplicação:** Essencial para a persistência dos dados. Sem ele, a aplicação não conseguiria armazenar e recuperar informações, tornando-a inoperante. Em um ambiente de produção, garantir a disponibilidade e a integridade dos dados é crucial.
- **Tecnologias:** MySQL 8.0.
- **Dockerfile:**
    - FROM mysql:8.0: Utiliza a imagem oficial do MySQL 8.0.
    - ENV MYSQL_ROOT_PASSWORD=sua_senha_aqui: Define a senha do usuário root (será sobrescrita pelo docker-compose).
    - ENV MYSQL_DATABASE=EmpresaVaga: Define o nome do banco de dados a ser criado na inicialização.

3. Contêiner email_service (Serviço de E-mail)

- **Funcionalidade:** Este contêiner é um **microsserviço Spring Boot dedicado exclusivamente ao envio de e-mails**. Ele recebe requisições do contêiner spring_app com os detalhes do e-mail (destinatário, assunto, corpo, etc.) e utiliza o protocolo SMTP para enviá-los. Essa separação permite que a funcionalidade de e-mail seja escalada independentemente e mantenha o backend focado em suas responsabilidades principais.
- **Importância na Aplicação:** Adiciona uma funcionalidade crítica para a experiência do usuário, permitindo notificações automáticas sobre o status de inscrições, o que é comum em sistemas de produção. A separação em um microsserviço garante resiliência e manutenibilidade.
- **Tecnologias:** Spring Boot, Java 17, Maven, SMTP.
- **Dockerfile:**
    - Similar ao backend, utiliza etapas de build e produção para criar uma imagem leve.
    - EXPOSE 8081: Indica que o serviço escuta na porta 8081.
    - HEALTHCHECK: Inclui um health check para verificar a disponibilidade do serviço, crucial em ambientes de produção.

Relacionamentos entre os Contêineres
A comunicação entre os contêineres é estabelecida através da **rede Docker interna (devops-net)**, utilizando os **nomes dos serviços** definidos no docker-compose.yml. Isso garante que a aplicação possa ser facilmente orquestrada e que os serviços se descubram sem depender de mapeamentos de porta para o host local.

- O contêiner **spring_app (Backend)** acessa o **banco de dados MySQL (db)** através do nome do serviço db na URL de conexão (ex: jdbc:mysql://db:3306/EmpresaVaga). O spring_app depende da disponibilidade do db para iniciar, conforme a condição service_healthy.

- Similarmente, o contêiner **spring_app (Backend)** envia requisições HTTP (POST para /api/email/send) para o **serviço de e-mail (email-service)** através do nome do serviço email-service (como visto no código http://email-service:8081/api/email/send). O spring_app depende da inicialização do email-service para começar, conforme a condição service_started.

Não há comunicação direta entre os contêineres mysql_db e email_service; ambos são utilizados separadamente pelo spring_app, que centraliza a lógica da aplicação.

Tecnologias utilizadas
- Java 17
- Spring Boot
- Maven
- MySQL 8
- Docker
- Docker Compose
- SMTP (para envio de e-mails)

Manual de Instalação e Execução
Pré-requisitos

- Docker
- Docker Compose
- Acesso à internet para baixar as imagens base.

Como Executar o Projeto com Docker

1. Clone o repositório
   - git clone (https://github.com/natycristina/Trabalho1-devops.git)

2. Abra a pasta onde o repositório foi clonado

   No meu caso é:

   cd C:\Users\Nataly\Trabalho1-devops>

3. Configurar Credenciais (MySQL e Email Service):

   Edite o arquivo docker-compose.yml e substitua os placeholders pelas suas credenciais

   Exemplo:

- MySQL:

```
services:
  app:
    environment:
      SPRING_DATASOURCE_USERNAME: seu_username_aqui
      SPRING_DATASOURCE_PASSWORD: sua_senha_aqui
```

Altere as linhas indicadas no serviço app para refletir suas credenciais do MySQL. Suponha que seu MySQL usa root para usuário e senha, você as usaria aqui:

```
SPRING_DATASOURCE_USERNAME: root
SPRING_DATASOURCE_PASSWORD: root
```

- Email Service:

```
email-service:
  environment:
    SPRING_MAIL_USERNAME: seu_email_que_gostaria_de_receber_o_gmail # Substitua pelo seu e-mail
    SPRING_MAIL_PASSWORD: sua_senha_do_gmail_para_app # Substitua pela senha de app
```

Para o GMail, é necessário gerar uma "senha de app" se você tiver a verificação em duas etapas ativada, ou permitir acesso a aplicativos menos seguros nas configurações da sua conta Google.

4. Execute os contêineres

   No diretório raiz do projeto (onde está o docker-compose.yml), execute:

   - docker-compose up --build

5. Acesse a aplicação
   - Aplicação Spring Boot: http://localhost:8080

6. Parar os contêineres
    - docker-compose down