Vítor Milanez **RA:804319**

Descrição da aplicação escolhida para a prática de Devops

- O sistema apresentado neste repositório tem como objetivo cadastrar locações de clientes em locadoras, armazenando no banco de dados informações relevantes como a data e hora da locação, além de identificar qual cliente alugou com qual locadora.
- O projeto conta com um sistema de login com diferentes níveis de acesso: clientes, locadoras e administradores. Cada perfil possui permissões específicas para execução de suas respectivas tarefas.
- Idiomas suportados: Português-BR, Inglês e Alemão.

Funcionalidades

- O projeto teve foco em cobrir os requisitos do seguinte arquivo: Requisitos A 1 Servlets.pdf

Requisitos, Instalação e Execução

Antes da containerização, era necessário ter o Java (preferencialmente versão 8 ou superior) e o gerenciador de dependências Maven instalados localmente para executar a aplicação com o seguinte comando:

```
mvn clean package tomcat7:run.
```

Após a adoção do Docker, era necessário ter tanto o Docker quanto o Docker Compose instalados para subir a aplicação com:

```
docker-compose up --build
```

Agora, com a adoção do Kubernets junto com o Helm Chart, é necessário:

- Docker instalado (https://docs.docker.com/get-docker/))
- Kubernetes (Minikube) instalado (https://minikube.sigs.k8s.io/docs/start)
- Helm instalado (https://helm.sh/docs/intro/install/)
- Arquivo .env no diretório bikes-rent com o seguinte conteúdo:

```
DB_HOST=db
DB_USER=user
DB_PASSWORD=password
DB_PORT=3306
DB_NAME=bikeRentSystem
```

Para executar a aplicação:

1. Clone o repositório:

```
git clone https://github.com/VMila/Pratica-Devops-k8s.git
cd Pratica-Devops-k8s/bikes-rent
```

2. Executar o script de automação:

O script build-local.sh foi criado para automatizar todo o processo de preparação e implantação.

```
chmod +x build-local.sh
./build-local.sh
```

3. Acesse a aplicação em:
   http://k8s.local para a aplicação base
   http://mail.k8s.local para o serviço de e-mail Mailhog

Alternativamente, em sistemas Windows só consegui executar a aplicação utilizando os comandos:

```
kubectl port-forward svc/bike-release-bike-app-service 8080:8080
kubectl port-forward svc/bike-release-mailhog-service 8025:8025
```

Para depois acessar a aplicação em http://localhost:8080 e http://localhost:8025

Arquitetura com Kubernets

A arquitetura atual utiliza o Kubernetes para orquestrar os contêineres que compõem a aplicação. Cada componente do sistema antigo foi mapeado para objetos nativos do Kubernetes.

- **bike-app :** Aplicação base, do sistema de locação de bicicletas, gerido por um **Deployment**, que garante que um número desejado de réplicas da aplicação esteja sempre a ser executado. É exposto internamente por um **Service**.
- **db (MySQL):** Gerido por um **Deployment** e um **Service**. Os dados são persistidos através de um **PersistentVolumeClaim (PVC)**, que solicita armazenamento do cluster, garantindo que os dados não se perdem. As senhas são geridas de forma segura por um **Secret**, e a estrutura inicial das tabelas é criada através de um **ConfigMap** que contém o script SQL de inicialização.
- **mailhog (Servidor de E-mail):** Gerido por um **Deployment** e um **Service**, operando de forma isolada para capturar os e-mails enviados pela aplicação.
- **nginx:** A função de proxy reverso e roteamento de tráfego foi substituída por um recurso de **Ingress**. O Ingress define as regras de acesso externo (como k8s.local e mail.k8s.local) e direciona as requisições para os serviços corretos dentro do cluster.

Tecnologias Utilizadas

- Java EE | Servlets
- Tailwind CSS
- JavaScript
- Docker (com Docker Compose)
- Kubernetes (Minikube)
- Helm