Link repositorio: https://github.com/MarcosVini222/Sprint4-devops.git

Integrantes: Pedro Cardoso - rm555160, Heitor Ortega - rm557825, Marcos Lourenço - rm556496

Link Video Devops: https://youtu.be/3QE1Y75fh4M

Componentes usados:

🔹 1. GitHub

- **Repositório:** MarcosVini222/Sprint4-devops
- **Função:** Armazena o código-fonte do projeto (Java Spring Boot + Dockerfile + pipeline YAML).
- **Integração:** Conectado ao Azure DevOps — um *push* para a branch `main` aciona automaticamente a pipeline CI/CD.

🔹 2. Azure DevOps

- **Projeto:** `LorAch`
- **Service Connections:**
  - `sc-azure-lorch` → conexão com a subscription e Resource Group no Azure.
  - `sc-acr-lorch` → conexão com o Azure Container Registry (ACR).
- **Função:**
  - **CI (Continuous Integration):**
    - Faz *build* do projeto Spring Boot.
    - Executa testes automatizados (quando configurados).
    - Constrói a imagem Docker.
    - Faz *push* da imagem para o ACR.
  - **CD (Continuous Deployment):**

        - Executa comandos Azure CLI (`az container create`) para realizar o deploy automático no Azure Container Instance (ACI).

- **Arquivo principal:** `azure-pipelines.yml`

🔹 3. Azure Container Registry (ACR)

- **Nome:** `acrlorch`
- **Função:** Armazena as imagens Docker geradas pela pipeline CI.
- **Exemplo de imagem:** `acrlorch.azurecr.io/motoblu:latest`
- **Autenticação:** via `sc-acr-lorch` e variáveis seguras (`ACR_USERNAME`, `ACR_PASSWORD`).

🔹 4. Azure Container Instance (ACI)

- **Nome:** `aci-motoblu`
- **Localização:** `brazilsouth`
- **Função:** Executa o container da aplicação Java a partir da imagem armazenada no ACR.
- **Configuração:**
    - Porta exposta: `8080`
    - Variáveis de ambiente:
        - `DB_HOST`
        - `DB_NAME`
        - `DB_USER`
        - `DB_PASS`

        - `GITHUB_CLIENT_ID`
        - `GITHUB_CLIENT_SECRET`

Comando de deploy automático:

```
az container create \
  --resource-group rg_sprint \
  --name aci-motoblu \
  --image acrlorch.azurecr.io/motoblu:latest \
  --cpu 1 --memory 1.5 \
  --ports 8080 \
  --dns-name-label motoblu556496 \
  --environment-variables ...
```

        -

🔹 5. Azure Database for MySQL – Flexible Server

- **Nome:** `mysqlmotoblu556496`
- **Database:** `motoblu`
- **Usuário:** `motoblu@mysqlmotoblu556496`
- **Função:** Banco de dados relacional da aplicação.
- **Conexão:** realizada via variáveis de ambiente no container (`spring.datasource.*`).

Exemplo de URL:

```
jdbc:mysql://mysqlmotoblu556496.mysql.database.azure.com:3306/motoblu?useSSL=true&serverTimezone=UTC
```

-

🔹 6. Resource Group

- **Nome:** `rg_sprint`
- **Função:** Agrupa todos os recursos da solução:
    - ACR (`acrlorch`)
    - ACI (`aci-motoblu`)
    - Banco de dados MySQL (`mysqlmotoblu556496`)
- **Localização:** `Brazil South`

🔹 7. Azure Key Vault (opcional – recomendado para nota máxima)

- **Função:** Armazenar de forma segura variáveis sensíveis (DB_PASS, Client Secrets etc.).
- **Status:** Ainda pode ser adicionado para demonstrar boas práticas de segurança.

🔹 8. Docker

- **Função:** Containeriza a aplicação Spring Boot.
- **Arquivo:** `Dockerfile`

Exemplo de imagem gerada:

`acrlorch.azurecr.io/motoblu:latest`

-
- **Etapas principais:**
    1. Build da aplicação com Maven/Gradle.
    2. Copia o `.jar` para o container.
    3. Expõe a porta `8080`.
    4. Define o entrypoint para o Spring Boot.

🔹 9. Spring Boot Application

- **Versão:** Java 21, Spring Boot 3.4.4
- **Pacotes:** `spring-boot-starter-web`, `spring-boot-starter-data-jpa`, `mysql-connector-j`, `spring-boot-starter-security`, `flyway-core`.
- **Função:** API REST conectada ao MySQL, com autenticação via OAuth2 (GitHub).
- **Configuração:** via `application.properties` usando variáveis de ambiente.

🔹 10. Ferramentas de Apoio

- **Postman / Swagger:** Testes da API REST.
- **Azure CLI:** Deploy e monitoramento de containers.
- **Git / GitHub Desktop:** Versionamento de código e integração com Azure DevOps.

Criação do banco

```
$RG = "rg_sprint"
$LOCATION = "brazilsouth"
$MYSQL_SERVER = "mysqlmotoblu556496"
$ADMIN_USER = "motoblu"
$ADMIN_PASS = "SenhaForte!123"
$DB_NAME = "motoblu_db"

az mysql flexible-server create `
  --resource-group $RG `
  --name $MYSQL_SERVER `
  --location $LOCATION `
  --admin-user $ADMIN_USER `
  --admin-password $ADMIN_PASS `
  --sku-name Standard_B1ms `
  --public-access 0.0.0.0 `
  --tier Burstable


az mysql flexible-server db create `
```

```
  --resource-group $RG `
  --server-name $MYSQL_SERVER `
  --database-name $DB_NAME
```

```yaml
trigger:
 branches:
   include:
     - main
     - master

pool:
 vmImage: 'ubuntu-latest'

variables:
 appName: 'motoblu556496'
 acrName: 'acrrm556496'
 acrLoginServer: 'acrrm556496.azurecr.io'
 imageRepository: 'motoblu'
 dockerfilePath: 'Dockerfile'
 tag: 'latest'
 resourceGroup: 'rg_sprint'
 containerGroup: 'aci-motoblu'
 location: 'brazilsouth'

stages:

 - stage: Build
   displayName: 'Build and Push Image to ACR'
   jobs:
     - job: BuildAndPush
       displayName: '🔨 Build and Push Docker Image'
       steps:
         - task: Docker@2
           displayName: 'Build and Push Docker Image to ACR'
           inputs:
             containerRegistry: 'SC-FIAP-556496'
             repository: '$(imageRepository)'
             command: 'buildAndPush'
             Dockerfile: '$(dockerfilePath)'
             tags: |
               $(tag)

 - stage: Deploy
   displayName: 'Deploy Image to Azure Container Instance (ACI)'
```

```
    dependsOn: Build
    condition: succeeded()
    jobs:
      - job: DeployToACI
        displayName: '🚀 Deploy to Azure ACI'
        steps:
          - task: AzureCLI@2
            displayName: 'Deploy to Azure Container Instance'
            inputs:
              azureSubscription: 'SC-AZURE-RM-556496'
              scriptType: 'bash'
              scriptLocation: 'inlineScript'
              inlineScript: |
                echo "🔹 Iniciando deploy no Azure Container
Instance..."

                az acr login --name $(acrName)

                az container delete \
                  --resource-group $(resourceGroup) \
                  --name $(containerGroup) \
                  --yes || true

                echo "🔹 Criando novo container..."
                az container create \
                  --resource-group $(resourceGroup) \
                  --name $(containerGroup) \
                  --image
$(acrLoginServer)/$(imageRepository):$(tag) \
                  --os-type Linux \
                  --cpu 1 \
                  --memory 1.5 \
                  --registry-login-server $(acrLoginServer) \
                  --registry-username $(ACR_USERNAME) \
                  --registry-password $(ACR_PASSWORD) \
                  --dns-name-label $(appName) \
                  --restart-policy Always \
                  --ports 8080 \
                  --environment-variables \

DB_HOST="mysqlmotoblu556496.mysql.database.azure.com" \
                      DB_NAME="motoblu" \
                      DB_USER="motoblu@mysqlmotoblu556496" \
                      DB_PASS="$(DB_PASS)" \
                      GITHUB_CLIENT_ID="$(GITHUB_CLIENT_ID)" \

GITHUB_CLIENT_SECRET="$(GITHUB_CLIENT_SECRET)" \
                  --location $(location)
```

```
                echo "✅ Deploy concluído!"
                echo "🌐 Acesse em:
http://$(appName).$(location).azurecontainer.io:8080"


DockerFILe


FROM gradle:8.7.0-jdk21 AS build
WORKDIR /usr/app
COPY . .
RUN chmod +x ./gradlew
RUN ./gradlew clean build -x test

FROM eclipse-temurin:21-jre
WORKDIR /app
COPY --from=build /usr/app/build/libs/*.jar app.jar

EXPOSE 8080

ENV DB_USER=""
ENV DB_PASS=""
ENV GITHUB_CLIENT_ID=""
ENV GITHUB_CLIENT_SECRET=""

ENTRYPOINT ["java", "-jar", "app.jar"]
```

Descrição do Projeto

Este projeto tem como objetivo o desenvolvimento de uma API RESTful em Java com Spring Boot, para gerenciar a entrada, saída e relacionamento entre Moto, Chaveiro, Funcionário e Pátio. A aplicação possui estrutura modularizada e segue boas práticas como:

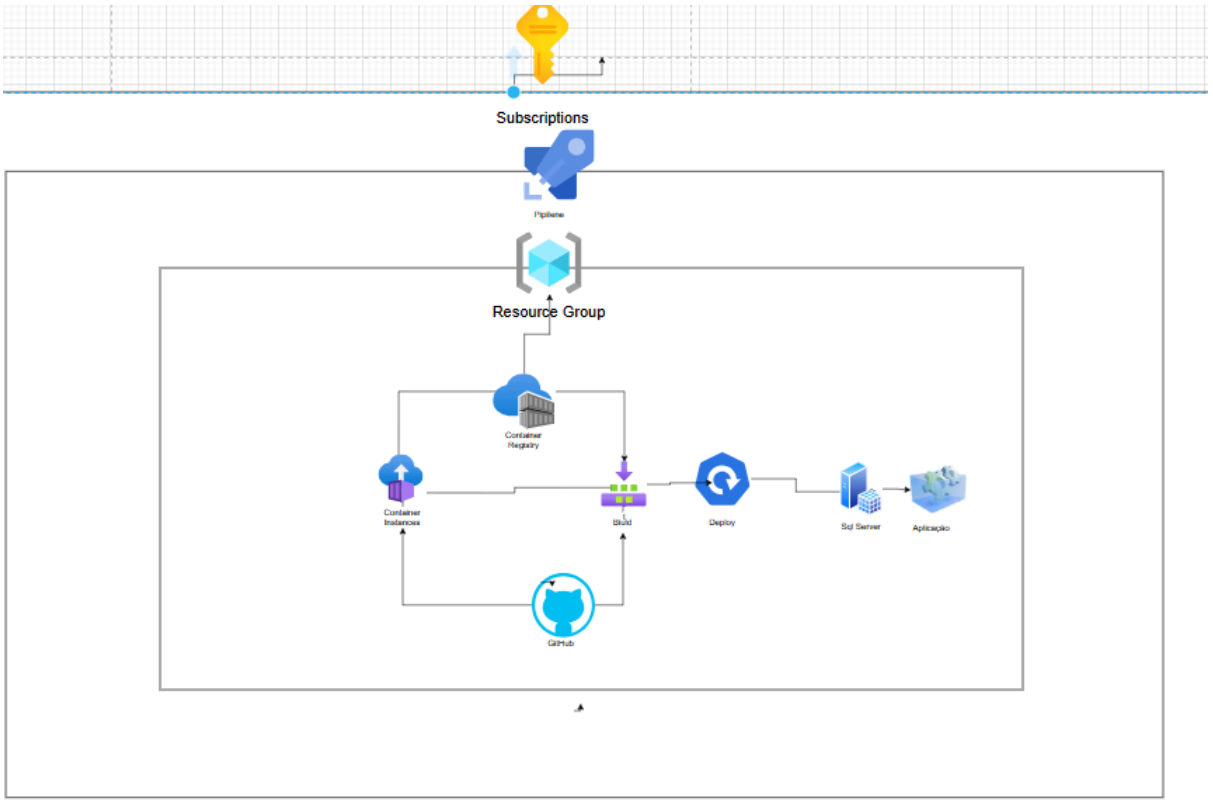
- Uso de DTOs para controle de entrada/saída de dados
- Validações com Bean Validation
- Paginação e ordenação nos endpoints de listagem
- Cache com Spring Cache para otimização de desempenho
- Manipulação de exceções com um Exception Handler global

FUNCIONALIDADES

- Banco de dados Oracle
- Spring web
- Spring boot
- Sptring data JPA

- Gradle
- Flyway (migrations)
- Spring Security (OAuth 2)
- Thymeleaf (MVC)

Para as próximas entregas, planejamos conectar uma API de bluetooth, para conseguirmos linka-las com as motos, que é a ideia do nosso projeto. Basicamente, o funcioanrio conseguiria localizar sua moto através de conexões por bluetooth, que a partir dai, mostraria a localização da moto no galpão. Obviamente, isso ainda não foi implementado mas está sendo planejado pelos desenvolvedores do projeto.



Diagrama