



Trabalho Final

TCP, Etapa 3


Com Github Actions

Integrantes: André Mainardi Klarmann, João Gabriel Rau
Wendt, Juan Marcelo Trabaina, Lucas Leal Meregalli





Como configuramos o Github Actions



Para configurar o GitHub Actions para um projeto Java primeiro precisamos saber qual ferramentas de compilação usamos no projeto.

No nosso caso usamos Maven, então criamos um arquivo de fluxo de trabalho (workflow) que será armazenado no diretório `.github/workflows/` do repositório.

O próprio github já oferece diversos templates de arquivos de configuração com diferentes ferramentas de compilação, então basta selecionar e modificar o necessário

O nome do arquivo se chama `main.yml` e possui a seguinte configuração

```
name: CI/CD

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Set up JDK 23
        uses: actions/setup-java@v4
        with:
          java-version: '23'
          distribution: 'temurin'
          cache: maven

      - name: Clean and compile project
        run: mvn clean compile
      - name: Run tests
        run: mvn test
      # Push to production branch after successful build and tests
      - name: Push to production branch
        if: success()
        run: |
          git config --global user.name "GitHub Actions"
          git config --global user.email "actions@github.com"
          git checkout -b production
          git add .
          git commit -m "Deploy changes to production"
          git push origin production
    env:
      GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN }
```



Apresentação de integração com falha e êxito em vídeo

Link: <https://youtu.be/tc86NWBUIrA>

Dificuldades na realização:

■ Nosso grupo conseguiu realizar a integração com certa facilidade, porém houve um empecilho na configuração que fazia o github actions “não rodar os testes”, porém, após um tempo descobrimos que faltava uma dependência* no arquivo de configuração do projeto que o Maven usa.

Então, bastava adicionar as seguintes linhas no arquivo `pom.xml` e tudo deu certo:

```
<plugin>
<groupId>org.apache.maven.plugins</groupId>
<artifactId>maven-surefire-plugin</artifactId>
<version>3.0.0-M8</version>
</plugin>
```

*O **Surefire** é um plugin do Maven utilizado principalmente para executar testes unitários em projetos Java. Ele faz parte da arquitetura de construção e teste do Maven