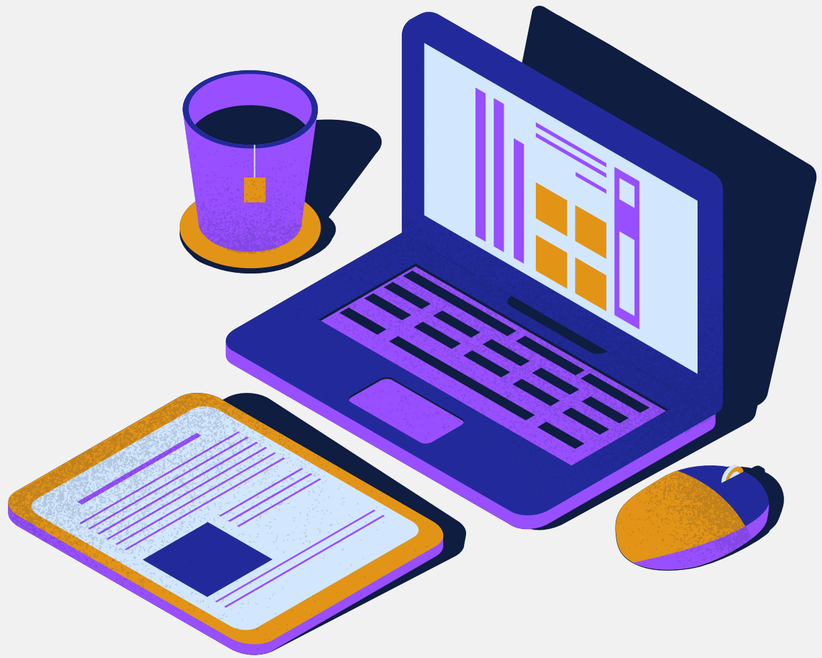


TESTES DE INTEGRAÇÃO CI/CD

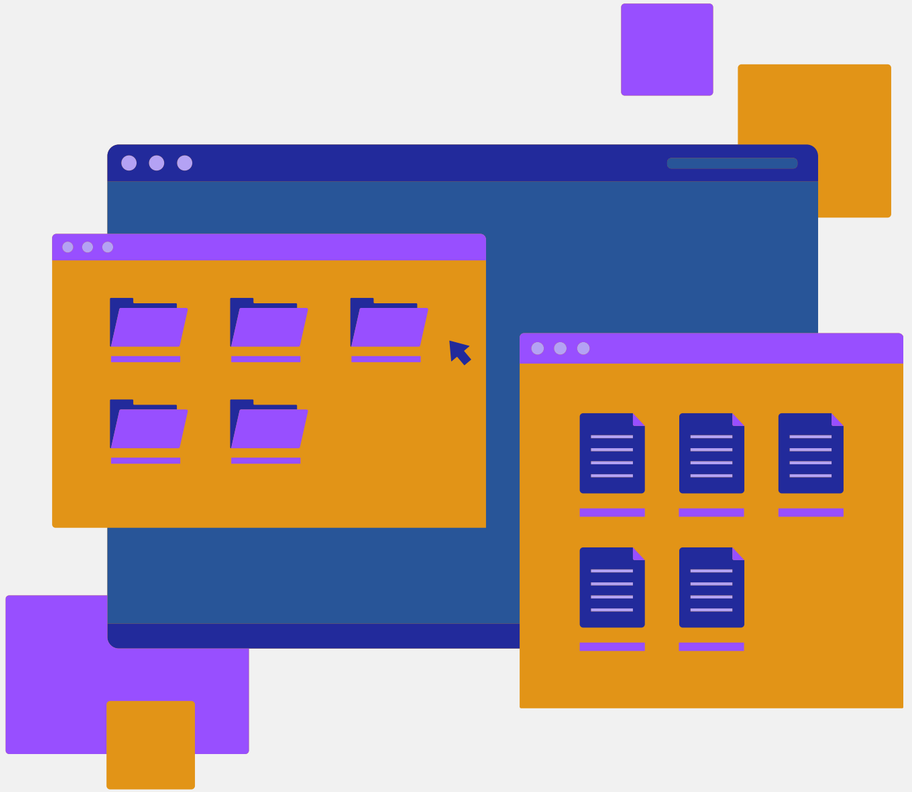
Ariane Silveira Correa - 20212014040001
Bruno Lins dos Santos Viana - 20221014040010
Cibele Regina Barros Diniz - 20221014040018
Marcus Vinicius Cadete Spencer Chaves - 20221014040001

O que é CI/CD?

- **Continuous Integration** (integração contínua) e **Continuous Delivery** (implantação/entrega contínua).

- São práticas que automatizam a integração, teste e entrega de código.

- **Objetivo**: melhorar a qualidade do código e eficiência no desenvolvimento.



Integração Contínua (CI)

- Processo de automação para integrar código em um repositório compartilhado.
- Execução automática de testes para detectar erros cedo.
- Benefícios: menos bugs, melhor padronização e qualidade do código.

Implantação Contínua (CD)

- Extensão da CI, focada em automatizar a construção, teste e implantação do código.
- Reduz a complexidade e o tempo necessário para colocar uma nova versão em produção.

Por que os testes de CI/CD devem ser automatizados?

- **Automatização**: Reduz trabalho manual e minimiza erros humanos.
- **Detecção precoce de erros**: Captura problemas logo no início.
- **Implantações mais rápidas**: Facilita a liberação de novas versões de software.
- **Feedback contínuo**: Permite melhorias constantes e ágeis.

Como funciona o CI/CD na prática?

01 Formatação do código, sem alterar seu comportamento, apenas melhorando a legibilidade para os seus mantenedores

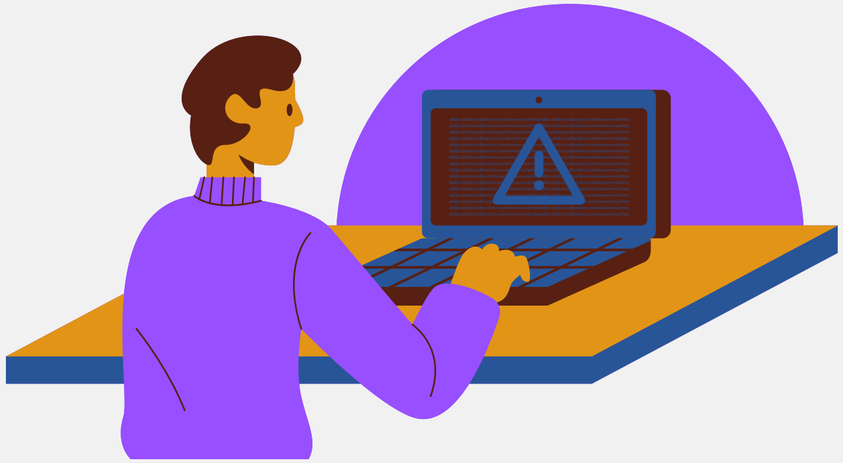
02 Execução dos testes automatizados para identificar erros, sem alterar o código, mas indicando possíveis problemas nele

03 Construção da aplicação: compilar e criar uma imagem do software

04 Análise de código estática, garantindo que o código siga boas práticas e esteja livre de erros básicos

05 Construção de uma nova versão do software, pronta para ser implantada

06 Implantação da versão criada na CD, tornando-a acessível aos usuários



Benefícios do CI/CD

- Código mais confiável e livre de bugs.
- Economia de tempo e redução de custos.
- Ciclo contínuo de desenvolvimento e atualizações do software.
- Integração contínua entre membros da equipe.
- Melhor rastreamento e transparência das mudanças.

Exemplo prático

CI

.github > workflows > ! django.yml

```yaml
name: Django CI

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

jobs:
  build:

    runs-on: ubuntu-latest
    strategy:
      max-parallel: 4
      matrix:
        python-version: [3.12.3, ]

    steps:
    - uses: actions/checkout@v4
    - name: Set up Python ${{ matrix.python-version }}
      uses: actions/setup-python@v3
      with:
        python-version: ${{ matrix.python-version }}
    - name: Install Dependencies
      run: |
        python -m pip install --upgrade pip
        pip install -r mysite/requirements.txt
    - name: Run Tests
      run: |
        python mysite/manage.py test

```

CD

```yaml
  workflow_dispatch:

jobs:
  build:…

  deploy:
    runs-on: ubuntu-latest
    needs: build
    environment:
      name: 'Production'
      url: ${{ steps.deploy-to-webapp.outputs.webapp-url }}
    permissions:
      id-token: write

    steps:
      - name: Download artifact from build job
        uses: actions/download-artifact@v4
        with:
          name: python-app

      - name: Unzip artifact for deployment
        run: unzip release.zip


      - name: Login to Azure
        uses: azure/login@v2
        with:
          client-id: ${{ secrets.AZUREAPPSERVICE_CLIENTID_B4C824BE984E4E609BA37CF9F5F697B3 }}
          tenant-id: ${{ secrets.AZUREAPPSERVICE_TENANTID_C4D5A36E833B466E9FCEFD1727B0B83E }}
          subscription-id: ${{ secrets.AZUREAPPSERVICE_SUBSCRIPTIONID_FA05D7D0D1CC4DAE897937174EDEE341 }}

      - name: 'Deploy to Azure Web App'
        uses: azure/webapps-deploy@v3
        id: deploy-to-webapp
        with:
          app-name: 'solarte'
          slot-name: 'Production'
```

https://github.com/arianesc/SolArte

Perguntas?