

Relatório: Testes Realizados no Projeto Jornada FAST ADOS

Introdução

Durante o desenvolvimento do projeto, foi aplicada uma abordagem de testes automatizados para verificar o bom funcionamento de partes do sistema. Os testes foram realizados utilizando a ferramenta pytest e a interface Swagger, visando garantir maior confiabilidade na análise e tratamento dos dados relacionados às ordens de serviço (OS) da fábrica.



Natureza dos Testes Realizados

Testes com Dados Válidos

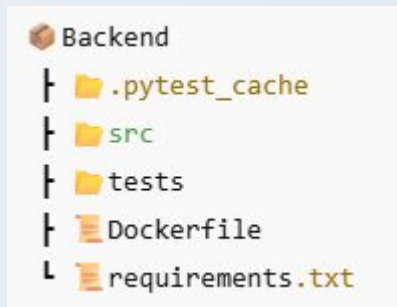
- Criação de ordens de serviço (OS) com todos os campos obrigatórios preenchidos corretamente.
- Envio de arquivos CSV corretamente formatados, contendo colunas esperadas e dados válidos.
- Teste de carga leve: criação de múltiplas OS em sequência.
- Envio de CSVs grandes para avaliação de performance.

Testes com Dados Inválidos ou Incompletos

- Envio de OS com campos obrigatórios ausentes, nulos ou vazios.
- Campos com tipos de dados incorretos (ex: string no lugar de número).
- Envio de CSVs com colunas trocadas, ausentes ou com valores nulos em campos obrigatórios.
- Arquivos CSV com dados de tipos incorretos (ex: texto em vez de número).
- Upload de arquivos com formatos inválidos, como .txt, .xlsx, arquivos vazios ou corrompidos.

Ferramenta Utilizada: Pytest

Pytest foi a ferramenta escolhida por sua leveza, simplicidade e integração com sistemas de CI (integração contínua) como o GitHub Actions. Todos os testes automatizados foram escritos em arquivos `test_*.py`, organizados dentro da pasta `tests/`.



Estrutura dos Testes automatizados

```
def test_upload_csv_arquivo_errado():  
    """  
    Testa o endpoint de upload de CSV com um arquivo de tipo incorreto (ex: .txt).  
    Espera uma resposta de status 400 (Bad Request).  
    """  
  
    file_path = os.path.join(TEST_FILES_DIR, "arquivo_errado.txt")  
    with open(file_path, "rb") as file:  
        response = client.post("/fast/OS/upload_csv", files={"file": ("arquivo_errado.txt", file, "text/plain")})  
    print("❌ Resposta API (Arquivo Não é CSV):", response.status_code, response.json())  
    assert response.status_code == 400  
  
    assert "0 arquivo deve ser um CSV." in response.json().get("detail", "")
```

1 passed, 1 warning in 1.77s

Testes Manuais via Swagger

Além dos testes automatizados, foram realizados testes manuais utilizando majoritariamente a interface Swagger.

file ★ required
string(\$binary)

documentos.pdf

Error: Bad Request









Response body

```
{  
  "detail": "O arquivo deve ser um CSV."  
}
```


Github Actions

test-backend


succeeded 49 minutes ago in 29s

- >  Set up job
- >  Checkout repo
- >  Setup Python
- >  Install dependencies
- >  Run backend tests
- >  Post Setup Python
- >  Post Checkout repo
- >  Complete job

```
===== test session starts =====  
platform linux -- Python 3.10.18, pytest-8.4.0, pluggy-1.6.0  
rootdir: /home/runner/work/Jornada_Fast_ADOS/Jornada_Fast_ADOS  
plugins: anyio-4.9.0  
collected 2 items  
  
Backend/tests/test_upload_csv.py .. [100%]
```

 test

Run Backend Python Tests #30: Commit [860748a](#) pushed by DaniloCossioDias

 Update python-tests.yml

Run Backend Python Tests #27: Commit [977d6f8](#) pushed by DaniloCossioDias

Resultados e Conclusões

Os testes demonstraram que a API responde corretamente na maioria dos cenários de uso comum e que existem tratativas de erro funcionando em diversos casos. No entanto, o principal problema identificado está no tratamento de erros inesperados. Em várias situações, os erros não informam claramente o que ocorreu, dificultando o diagnóstico. É necessário aprimorar essa parte para tornar o sistema mais robusto.

Próximos Passos

Escrever mais testes automatizados utilizando pytest para aumentar a cobertura e garantir maior robustez do sistema.

Focar em automatizar a execução dos testes integrando-os ao GitHub Actions, para garantir que todos os testes sejam executados automaticamente a cada alteração no código.

Expandir os testes para incluir cenários de erro mais complexos, melhorando a confiabilidade da aplicação.