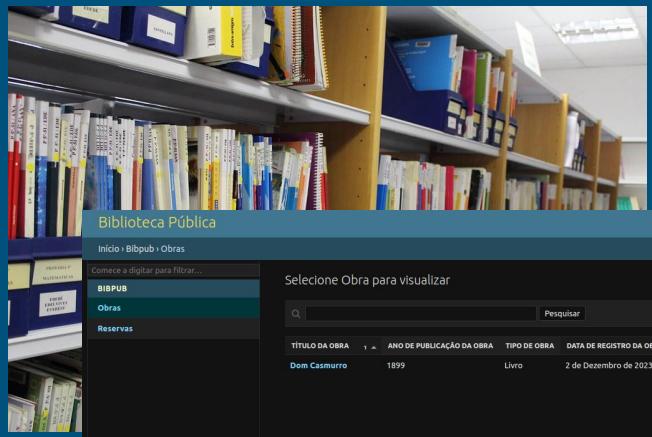
Teste de Software

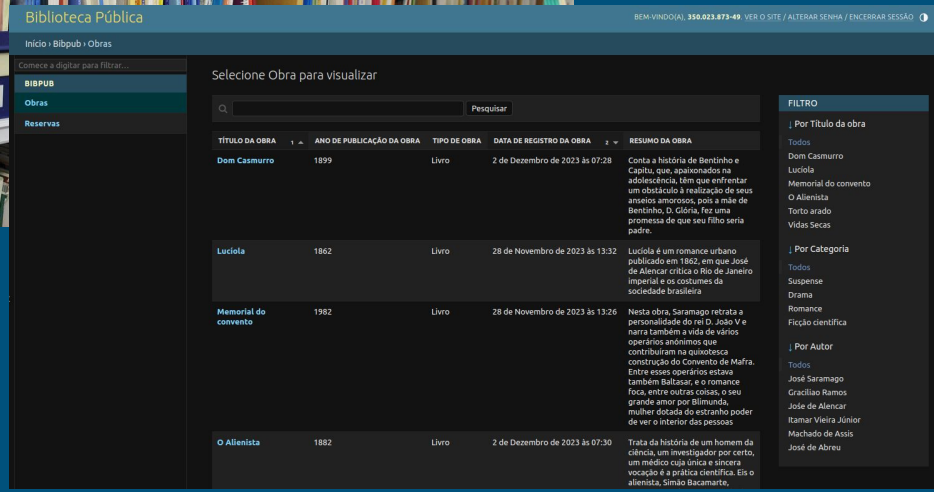
Universidade Federal do Rio Grande do Norte-UFRN
Instituto Metrópole Digital-IMD

Disciplina: PPGTI1011 - Teste de Software
Grupo: Alikson Oliveira & Marcelo Pinto
Prof. Uira Kulesza
2023.2

O sistema escolhido

- Dificuldade de acesso a sistema em desenvolvimento no trabalho
- Ganho de tempo por compartilhar projeto compatível
- Sistema de **Biblioteca Pública**
  - Cadastro de obras (livros e periódicos)
  - Cadastro de usuários
  - Possibilidade de reserva de obras
  - Possibilidade de empréstimo de obras





A linguagem escolhida

- Possibilidade de aprendizado de uma nova linguagem
- Ganho de tempo para viabilizar os projetos
- Possibilidade de compartilhar aprendizado com demais colegas
- Possível comparação com o que está sendo apresentado na disciplina
- **Python & Django & Unitest**
  - Possível utilização no projeto final do mestrado
  - Facilidade de implementação

Testes unitários - Autor

MODEL

```
class Autor(models.Model):
  nome = models.CharField("Nome do
    autor", max_length=200, null=False,
    unique=True)
  nascimento = models.DateField("Data de
    nascimento", null=True)
  biografia = models.TextField("Biografia
    do autor", max_length=1000,
    null=False)

  def data_nascimento_valida(self):
    return self.nascimento <=
      date.today()-timedelta(days=5*365)
```

TEST

```
class AutorModelTests(TestCase):
  def test_data_nascimento_futura(self):
    nascimento = date.today() +
      timedelta(days=30)
    autor = Autor(nascimento=nascimento)
    self.assertIs(autor.data_nascimento_v
      alida(), False)

  def test_data_nascimento_recente(self):
    nascimento = date.today() -
      timedelta(days=5*200)
    autor = Autor(nascimento=nascimento)
    self.assertIs(autor.data_nascimento_v
      alida(), False)
```

Testes unitários - Empréstimo

MODEL

```
class Emprestimo(models.Model):
  pessoa = models.ForeignKey (Pessoa,
    verbose_name=("Pessoa"),
    on_delete=models.CASCADE,)
  obras = models.ManyToManyField (Obra)
  dataemprest = models.DateTimeField ("Data
    do empréstimos",auto_now_add=True,
    null=False)
  prazo =
    models.PositiveSmallIntegerField("Prazo
    do empréstimo em dias", default=10)
  datadevol = models.DateTimeField("Data da
    efetiva devolução", null=True)

  def prazo_valido(self):
    return self.prazo >= 30 and self.prazo
      <= 90
```

TEST

```
class EmprestimoModelTests(TestCase):
  def test_prazo_inferior(self):
    prazo = 20
    emprestimo = Emprestimo(prazo=prazo)
    self.assertIs(emprestimo.prazo > 30,
      False)

  def test_prazo_superior(self):
    prazo = 200
    emprestimo = Emprestimo(prazo=prazo)
    self.assertIs(emprestimo.prazo < 90,
      False)
```

Execução de testes unitários com sucesso

```
(venv) mmpinto@desktop51:~/git/bibpub-imd$
(venv) mmpinto@desktop51:~/git/bibpub-imd$ python manage.py test bibpub
Found 14 test(s).
Creating test database for alias 'default'...
/home/mmpinto/git/bibpub-imd/venv/lib/python3.10/site-packages/django/db/models/fields/__init__.py:159
5: RuntimeWarning: DateTimeField Obra.dataatualizacao received a naive datetime (2023-11-25 18:40:55.3
46000) while time zone support is active.
  warnings.warn(
System check identified no issues (0 silenced).
.............../home/mmpinto/git/bibpub-imd/venv/lib/python3.10/site-packages/django/db/models/fields/__
init__.py:1595: RuntimeWarning: DateTimeField Reserva.datareserva received a naive datetime (2023-12-0
9 19:48:44.694584) while time zone support is active.
  warnings.warn(
.
----------------------------------------------------------------------
Ran 14 tests in 0.166s

OK
Destroying test database for alias 'default'...
(venv) mmpinto@desktop51:~/git/bibpub-imd$
```

Execução de testes unitários com falha

```
(venv) mmpinto@desktop51:~/git/bibpub-imd$
(venv) mmpinto@desktop51:~/git/bibpub-imd$ python manage.py test bibpub
Found 14 test(s).
Creating test database for alias 'default'...
/home/mmpinto/git/bibpub-imd/venv/lib/python3.10/site-packages/django/db/models/fields/__init__.py:1595: RuntimeWarning: DateTimeField Obra.dataatualizacao received a naive datetime
 (2023-11-25 18:40:55.346000) while time zone support is active.
  warnings.warn(
System check identified no issues (0 silenced).
.....FF...../home/mmpinto/git/bibpub-imd/venv/lib/python3.10/site-packages/django/db/models/fields/__init__.py:1595: RuntimeWarning: DateTimeField Reserva.datareserva received a na
ive datetime (2023-12-09 19:59:03.907183) while time zone support is active.
  warnings.warn(
.
======================================================================
FAIL: test_prazo_inferior (bibpub.tests.EmprestimoModelTests)
Prazo do empréstimo é de no mínimo 30 dias, devendo retornar False para prazo inferior a este limite
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/mmpinto/git/bibpub-imd/bibpub/tests.py", line 59, in test_prazo_inferior
    self.assertIs(emprestimo.prazo > 30, False)
AssertionError: True is not False

======================================================================
FAIL: test_prazo_superior (bibpub.tests.EmprestimoModelTests)
Prazo do empréstimo é de no máximo 90 dias, devendo retornar False para prazo superior a este limite
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/home/mmpinto/git/bibpub-imd/bibpub/tests.py", line 65, in test_prazo_superior
    self.assertIs(emprestimo.prazo < 90, False)
AssertionError: True is not False

----------------------------------------------------------------------
Ran 14 tests in 0.170s

FAILED (failures=2)
Destroying test database for alias 'default'...
(venv) mmpinto@desktop51:~/git/bibpub-imd$
```

Caso de teste - Cadastro de pessoas

Caso de teste - Cadastro de pessoas

CT_CadastroPessoa_001	
Descrição: Verificar o cadastro de uma pessoa com todos os campos preenchidos corretamente.	
Requisitos associados: Cadastro de Pessoa	Procedimentos associados: Acessar a página de cadastro, preencher todos os campos corretamente e confirmar o cadastro.
Pré-Condições: • Nenhuma	Pós-Condições: • A pessoa deve ser cadastrada com sucesso no sistema.
Passos: • Acesse a página de cadastro de pessoas. • Preencha todos os campos obrigatórios corretamente. • Confirme o cadastro.	
Critério de sucesso: • A pessoa é cadastrada corretamente no sistema.	

CPF Válido: CPF que segue o formato padrão (XXX.XXX.XXX-XX) e passa no algoritmo de validação. 11 dígitos

CPF Inválido: CPF que não segue o formato padrão ou não passa no algoritmo de validação. < 11 dígitos

Email Válido: E-mail que segue o formato padrão, devendo conter o @.

Email Inválido: E-mail que não segue o formato padrão ou não tem o @.

Entre outros

```python
class CadastroPessoaTestCase(TestCase):
    def test_cadastro_pessoa_valida(self):
        response = self.client.post('/cadastrar_pessoa/', {
          'nome': 'Maria',
          'nascimento': '1990-01-01',
          'cpf': '123.456.789-09',
          'sexo': 'F',
          'email': 'maria@imd.com',
          'cep': '12345678',
          'endereco': 'Rua Mossoro, 23',
          'cidade': 'Natal',
          'uf': 'RN',
          'origem': 'INTERNET',
        })
        self.assertEqual(response.status_code, 302)

        # Verifica se a pessoa foi cadastrada no banco de dados
        pessoa_cadastrada = Pessoa.objects.get(cpf='123.456.789-09')
        self.assertEqual(pessoa_cadastrada.situacaocadastro, 'PENDENTE')
```

```python
def test_cadastro_pessoa_invalida(self):
    # Tenta cadastrar uma pessoa com dados inválidos
    response = self.client.post('/cadastrar_pessoa/', {
        'nome': '',
        'nascimento': '',
        'cpf': '123.456.789-09',
        'sexo': 'M',
        'email': 'usuario@gmail.com',
        'cep': '12345678',
        'endereco': 'Rua mossoro, 1234',
        'cidade': 'Natal',
        'uf': 'RN',
        'origem': 'INTERNET',
    })
    # Verifica se "erro" está presente no conteúdo da resposta
    self.assertIn('erro', response.content.decode('utf-8').lower())

    # Verifica se a pessoa não foi cadastrada no banco
    pessoa_cadastrada = Pessoa.objects.filter(cpf='12345678909').first()
    self.assertIsNone(pessoa_cadastrada)
```

CT_ConsultaObra_002	
Descrição:Testar a consulta de informações sobre uma obra no sistema.	
Requisitos associados: O sistema deve permitir a consulta de informações sobre obras.	Procedimentos associados: Acessar a página de consulta de obras, inserir o título da obra desejada, realizar a busca.
Pré-Condições: • Ter pelo menos uma obra cadastrada no sistema	Pós-Condições: • Nenhuma alteração nas obras do sistema
Passos: • Acesse a página de consulta de obras. • Insira o título da obra desejada. • Realize a busca.	
Critério de sucesso: • O sistema exibe corretamente as informações sobre a obra consultada.	

```python
class ObraTestCase(TestCase):
    def setUp(self):
        categoria = Categoria.objects.create(descricao='Categoria Teste')
        autor = Autor.objects.create(nome='Autor Teste')
        pais = Pais.objects.create(codigo='BR', nome='Brasil', datainicial=datetime.now())
        editora = Editora.objects.create(nome='Editora Teste', pais=pais)

        self.obra = Obra.objects.create(
            titulo='Obra Teste',
            anopublicacao=2022,
            descricao='Descrição da obra teste',
            isbn='1234567890',
            tipo=Obra.TipoObra.LIVRO,
            categoria=categoria,
            autor=autor,
            editora=editora
        )

    def test_consulta_obra(self):
        obra_consultada = Obra.objects.get(titulo='Obra Teste')

        self.assertEqual(obra_consultada.titulo, 'Obra Teste')
        self.assertEqual(obra_consultada.anopublicacao, 2022)
        self.assertEqual(obra_consultada.descricao, 'Descrição da obra teste')
        self.assertEqual(obra_consultada.isbn, '1234567890')
        self.assertEqual(obra_consultada.tipo, Obra.TipoObra.LIVRO)
        self.assertEqual(obra_consultada.categoria, self.obra.categoria)
        self.assertEqual(obra_consultada.autor, self.obra.autor)
        self.assertEqual(obra_consultada.editora, self.obra.editora)
```

Outros Casos de Testes

- RESERVAR LIVROS
- EMPRÉSTIMO DE LIVROS
- DEVOLUÇÃO DE LIVROS

```
(python3) alikson@re352716 bibpub-imd % python manage.py test
Found 14 test(s).
Creating test database for alias 'default'...
/Users/alikson/Envs/python3/lib/python3.11/site-packages/django/db/models/fields/__init__.py:1595: RuntimeWarning: DateTimeField Obra.dataatualizacao received a naive datetime (2023-11-25 18:40:55.346000) while time zone support is active.
  warnings.warn(
System check identified no issues (0 silenced).
............/Users/alikson/Envs/python3/lib/python3.11/site-packages/django/db/models/fields/__init__.py:1595: RuntimeWarning: DateTimeField Reserva.datareserva received a naive datetime (2023-12-09 20:54:33.693755) while time zone support is active.
  warnings.warn(
.
----------------------------------------------------------------------
Ran 14 tests in 0.242s

OK
Destroying test database for alias 'default'...
(python3) alikson@re352716 bibpub-imd %
```

Conclusão

- Foram perfeitamente mapeáveis os conceitos apresentados em Java/JUnit para o Python/PyTest
- Desafio de desenvolver testes em linguagem diferente de Java trouxe muitos desafios e muito aprendizado
- Possibilidade de utilização da programação usada para desenvolver testes para outras utilidades, como automatizar carga de dados através de aplicações web, por exemplo


OBRIGADO!