Projeto Feriados

Orlando Saraiva Júnior

Agenda

- Django
  - Criação do projeto “Feriados”, com testes
  - O fluxo MVT
  - Modelos (classes Model)
  - ORM
- Git / Github
  - Uso do Git Desktop

Ativando o Ambiente Virtual

Vamos começar ...
No Windows

No prompt de comando (cmd), digite:

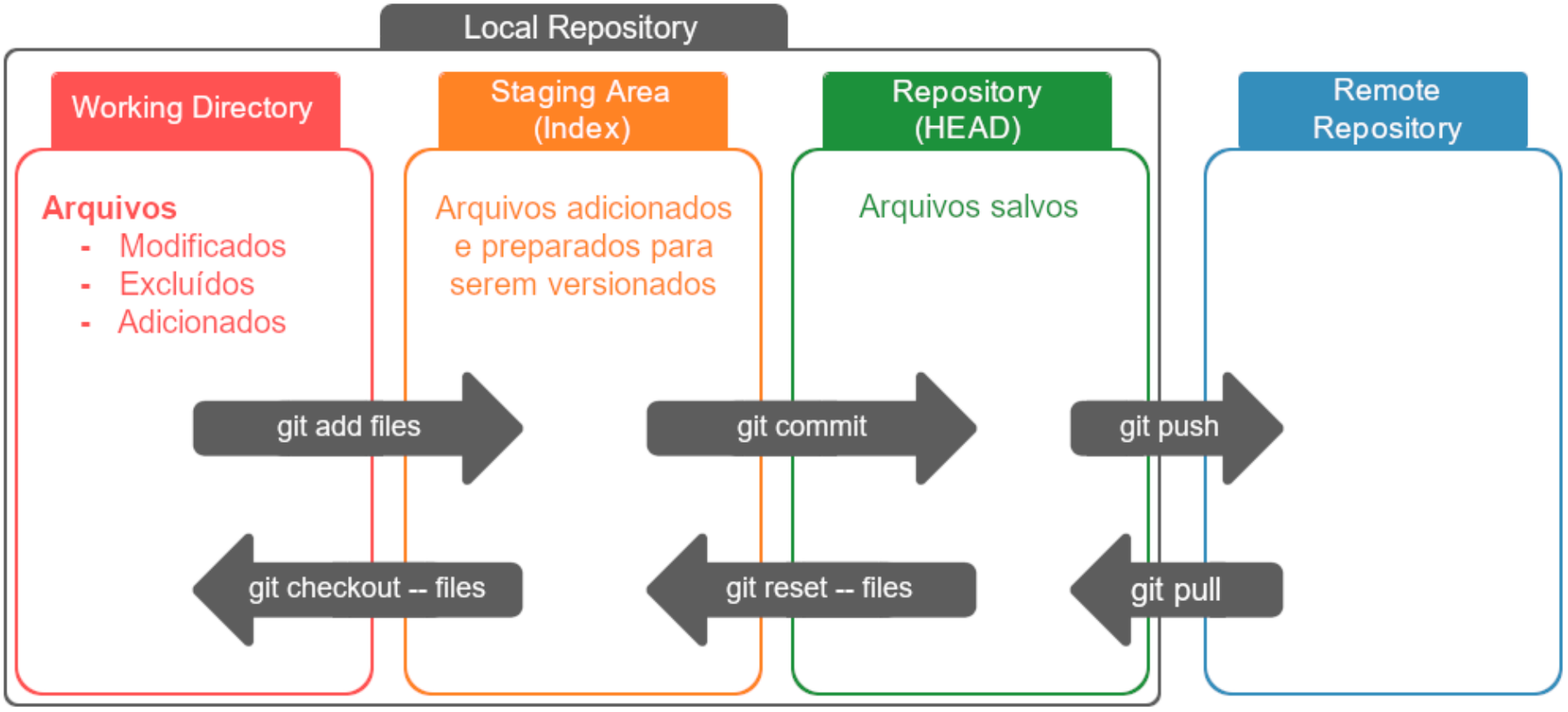
- pip install virtualenv
- virutalenv venv
- cd venv
- cd Scripts
- Activate.bat
- (venv)

Vamos começar ...
No Linux

No terminal, digite:

- virtualenv -p python3 venv
- source venv/bin/activate
- (venv)

O fluxo git


Local Repository
Working Directory
Staging Area (Index)
Repository (HEAD)
Remote Repository
Arquivos
- Modificados
- Excluídos
- Adicionados
Arquivos adicionados e preparados para serem versionados
Arquivos salvos
git add files
git commit
git push
git checkout -- files
git reset -- files
git pull

Framework Django

Uma história sobre desenvolvimento WEB..

“ Quero um cadastro de clientes, via web.

Precisa ser feito em uma semana”

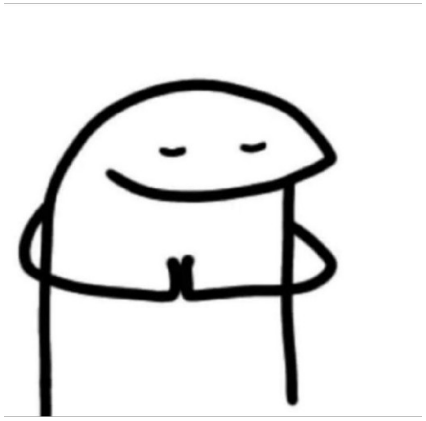
Uma história sobre desenvolvimento WEB..

“ Como fazer isso ??? ”

Uma história sobre desenvolvimento WEB..

Uma Semana depois...

Saiu ! Parabéns !!!



Uma história sobre desenvolvimento WEB..

Um mês depois...

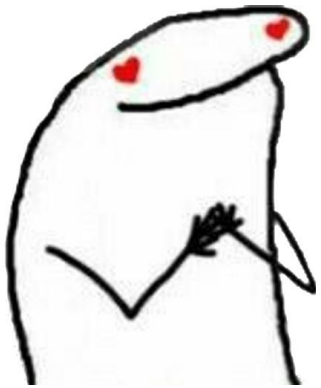
O número de usuários cresceu...

Uma história sobre desenvolvimento WEB..

Dois meses depois...

O número de usuários cresceu mais !

No próximo mês, pode rolar uma promoção !



Uma história sobre desenvolvimento WEB..

Dois meses depois...

Alguém descobriu uma falha de segurança !

SQL Injection

Você ficou preocupado !

Uma história sobre desenvolvimento WEB..

Dois meses depois...

Alguém descobriu uma falha de segurança !

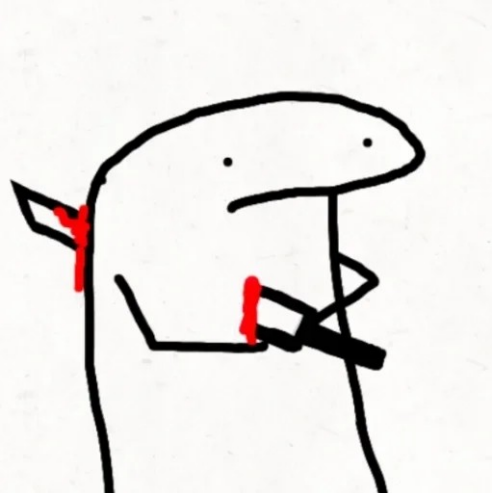
SQL Injection

Você ficou p*** !

Uma história sobre desenvolvimento WEB..

Dois meses depois...

Você foi demitido !



Uma história sobre desenvolvimento WEB..

Foco no negócio

vs

Foco na tecnologia

Uma história sobre desenvolvimento WEB..

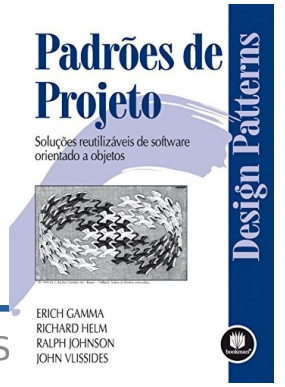
O que é um framework ?

Framework

Um framework é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma específica classe de software. [Deu89, JF88]

Você customiza um framework, para uma aplicação específica, atraves da criação de subclasses específicas para a aplicação, sendo essas subclasses específicas das classes abstratas do framework.

Os frameworks sempre tem um particular domínio de aplicação.

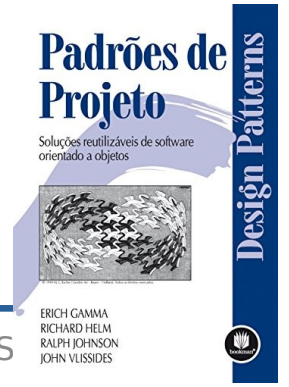


Framework

Um framework é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma específica classe de software. [Deu89, JF88]

Você customiza um framework, para uma aplicação específica, atraves da criação de subclasses específicas para a aplicação, sendo essas subclasses específicas das classes abstratas do framework.

Os frameworks sempre tem um particular domínio de aplicação.

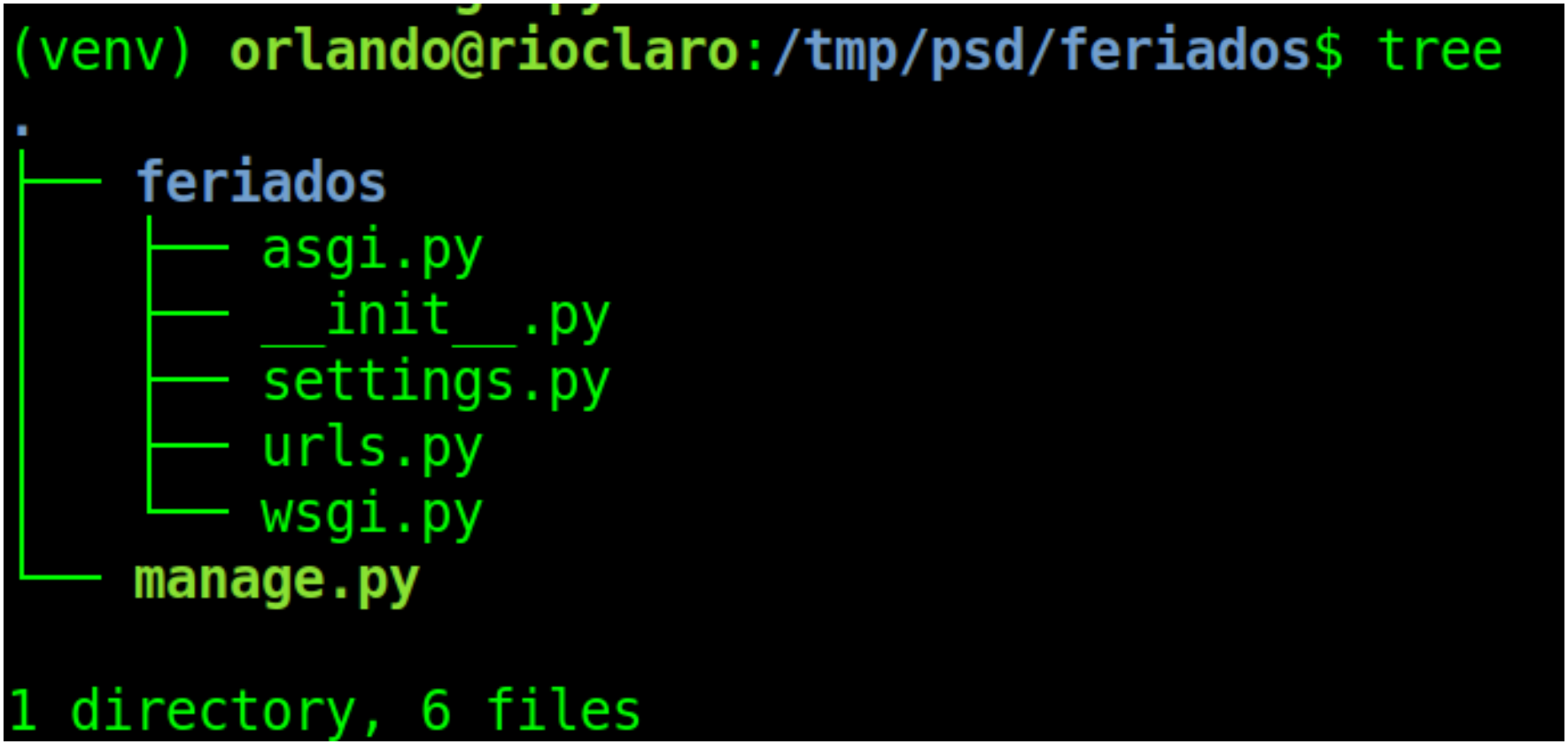


Projeto Feriado v. 1

Iniciando o projeto

- (venv) pip install django
- (venv) django-admin startproject feriados
- (venv) cd feriados

Arquivos Gerados



Iniciando o app

- (venv) python manage.py startapp core

Arquivos Gerados

```
(venv) orlando@rioclaro:/tmp/psd/feriados$ tree
.
├── core
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tests.py
│   └── views.py
├── feriados
│   ├── asgi.py
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

3 directories, 13 files
```

urls.py (do projeto)

- feriados/urls.py

```
from django.contrib import admin
from django.urls import path, include
urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('core.urls')),
]
```

Views

- feriados/core/views.py

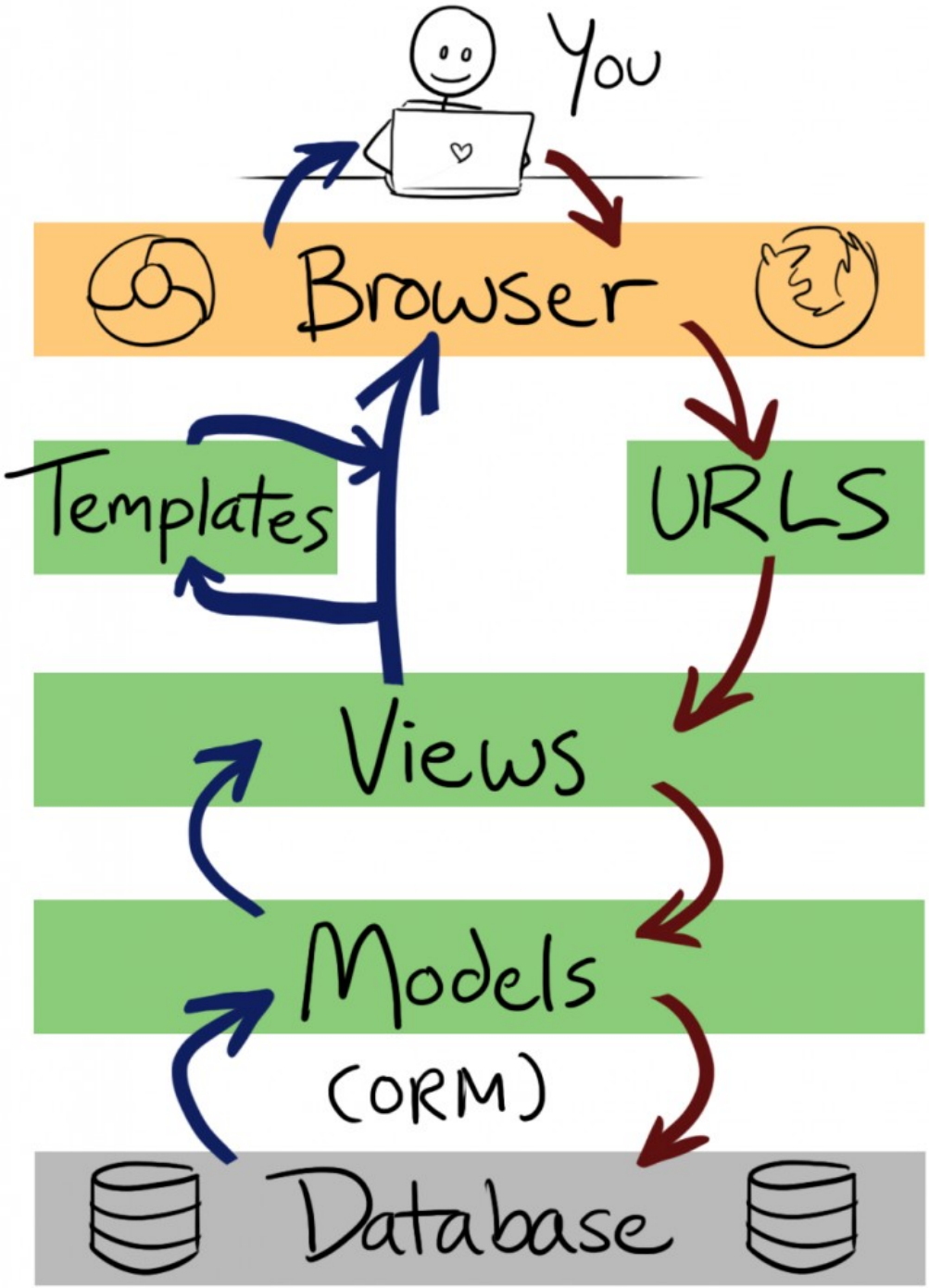
```
from django.http import HttpResponse


def natal(request):
    return HttpResponse("Não é natal.")
```

urls.py (do app)

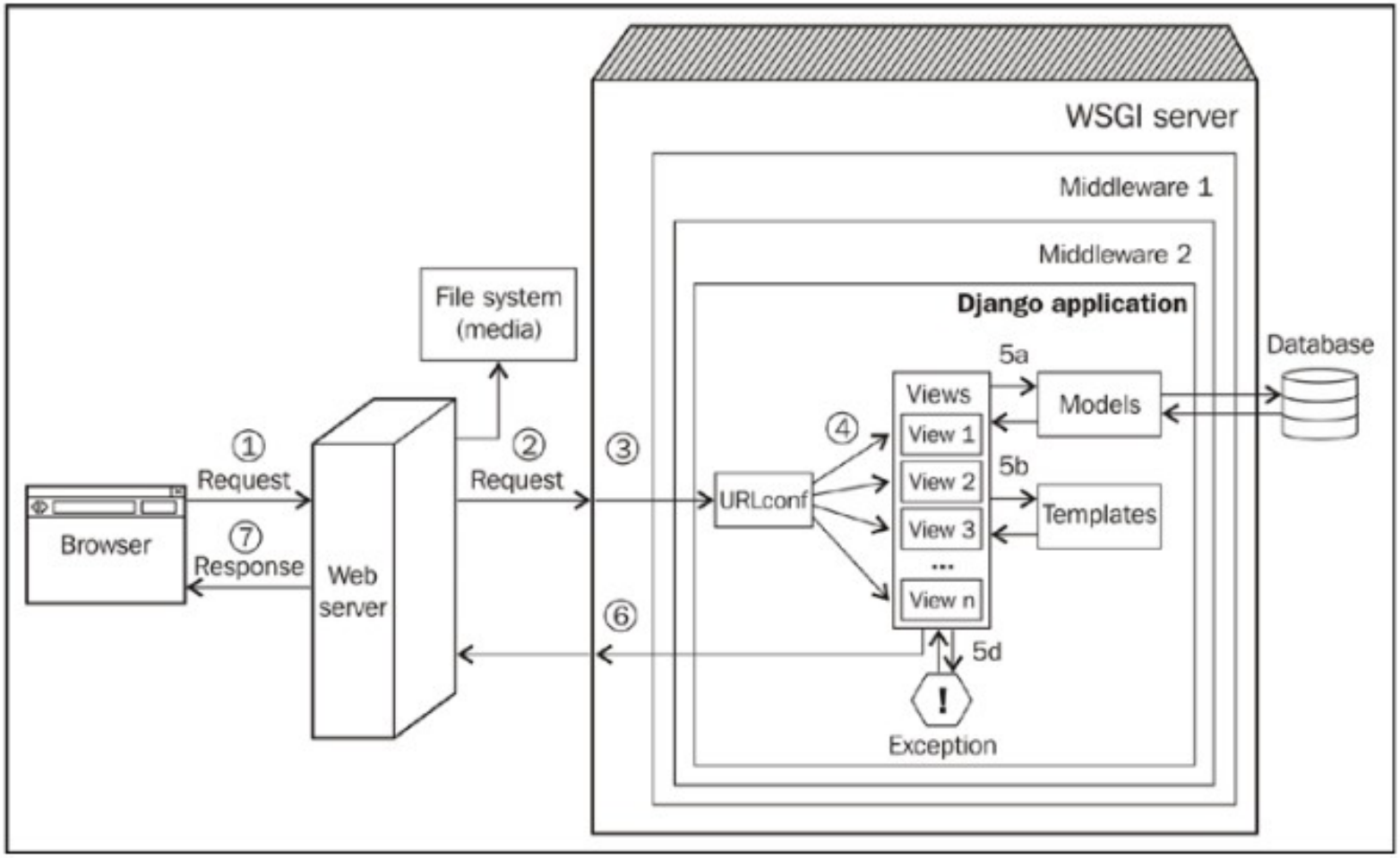
- core/urls.py

```
from django.urls import path
from . import views
urlpatterns = [
    path('', views.natal),
]
```

Fluxo MTV

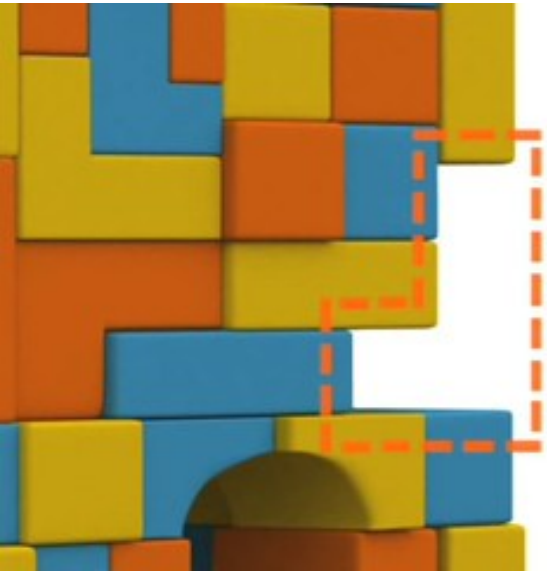


Fluxo MTV



How web requests are processed in a typical Django application

Test it !



Teste unitário /core/tests.py

```
from django.test import TestCase

class NatalTest(TestCase):
    def setUp(self):
        self.resp = self.client.get('/')

    def test_200_response(self):
        self.assertEqual(200, self.resp.status_code)

    def test_texto(self):
        self.assertContains(self.resp, 'natal')
```

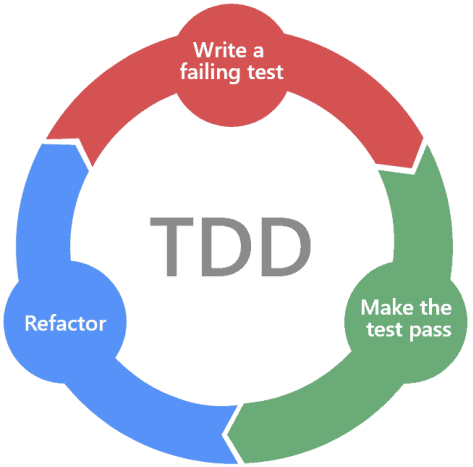
Atividade

- Criar uma rota **/tiradentes**
- Criar uma rota **/nossa_senhora**
- Criar uma rota **/independencia**

Resolução de Atividade

Atividade

- Criar uma rota **/tiradentes**
- Criar uma rota **/nossa_senhora**
- Criar uma rota **/independencia**



Resolução de Atividade

Projeto Feriado v. 2

Pasta Template

- Crie um diretório **templates** no diretório **core**

Primeiro template

- feriados/core/templates/natal.html

```
<!DOCTYPE html>
<html lang="pt_br">
    <head>
        <meta charset="UTF-8">
        <title>Feriado de Natal</title>
    </head>
    <body>
        <h1>Não é natal</h1>
    </body>
</html>
```

Views

- feriados/core/views.py

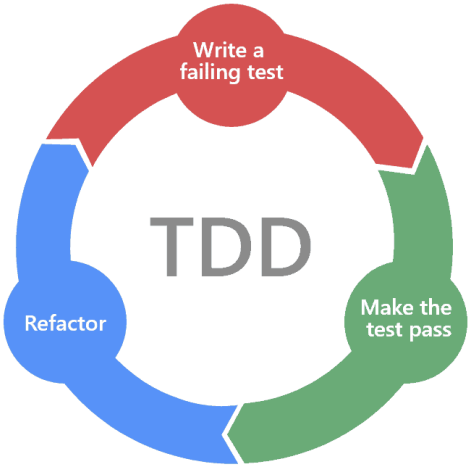
```
from django.shortcuts import render


def natal(request):
    return render(request, 'natal.html')
```

settings.py (No Projeto)

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core', # <== Incluir essa linha aqui
]
```

Atividade

- Criar templates e ajustar as views para cada rota de feriado desenvolvido anteriormente.



Resolução de Atividade

Projeto Feriado v. 3

Views

```
from django.shortcuts import render


def natal(request):
    contexto = {'natal': True,
                'tiradentes': False}
    return render(request, 'natal.html', contexto)
```

Template v.3

```
....
<body>
    {% if natal %}
    <h1>É natal</h1>
    {% else %}
        <h1>Não é natal</h1>
    {% endif %}
</body>
....
```

Teste unitário /feriado/tests.py

```
from django.test import TestCase

class NatalTest(TestCase):
    def setUp(self):
        self.resp = self.client.get('/')

    def test_200_response(self):
        self.assertEqual(200, self.resp.status_code)

    def test_texto(self):
        self.assertContains(self.resp, 'natal')

    def test_template_natal(self):
        self.assertTemplateUsed(self.resp, 'natal.html')
```

Projeto Feriado v. 4

Criando o Modelo

Um modelo é a fonte única e definitiva de informações sobre seus dados.

- Cada modelo é uma classe Python, uma subclasse de **django.db.models.Model**.
- Cada atributo representa um campo do banco de dados.

Fonte: https://docs.djangoproject.com/en/4.1/ref/models/

models.py (do app)

```
from django.db import models


class FeriadoModel(models.Model):
    nome = models.CharField('Feriado',max_length=50)
    dia = models.IntegerField('Data')
    mes = models.IntegerField('Mês')

    def __str__(self):
        return self.nome
```

Executar a migração

Execute:

- python manage.py makemigrations
- python manage.py migrate

O que aconteceu ?

Editar models.py (do app)

```
from django.db import models

class FeriadoModel(models.Model):
    nome = models.CharField('Feriado',max_length=50)
    dia = models.IntegerField('Data')
    mes = models.IntegerField('Mês')
    modificado_em = models.DateTimeField(
        verbose_name='modificado em',
        auto_now_add=False, auto_now=True)

    def __str__(self):
        return self.nome
```

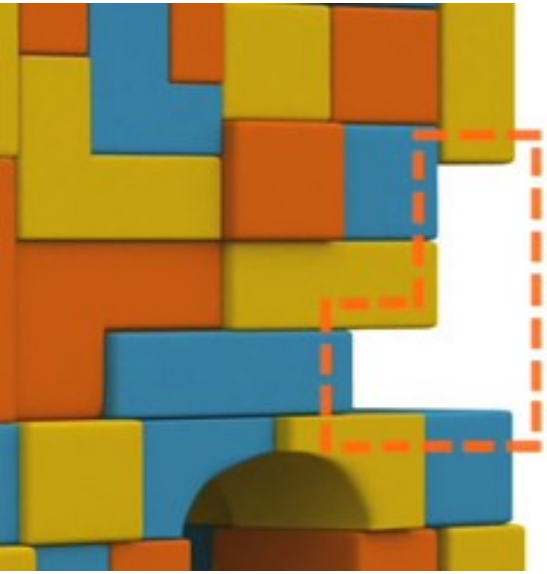
Executar a migração

Execute:

- python manage.py makemigrations
- python manage.py migrate

O que aconteceu ?

Test it !



Teste unitário

/feriado/tests.py (continuação)

```
from feriado.models import FeriadoModel
from datetime import datetime

class FeriadoModelTest(TestCase):
    def setUp(self):
        self.feriado = 'Natal'
        self.mes = 12
        self.dia = 25
        self.cadastro = FeriadoModel(
            nome=self.feriado,
            dia=self.dia,
            mes=self.mes,
        )
        self.cadastro.save()
```

Teste unitário

/feriado/tests.py (continuação)

```
def test_created(self):
    self.assertTrue(FeriadoModel.objects.exists())

def test_modificado_em(self):
    self.assertIsInstance(self.cadastro.modificado_em, datetime)

def test_nome_feriado(self):
    nome = self.cadastro.__dict__.get('nome', '')
    self.assertEqual(nome, self.feriado)

def test_dia_feriado(self):
    dia = self.cadastro.__dict__.get('dia', '')
    self.assertEqual(dia, self.dia)
```

ORM

Object-Relational Mapping

Object Relational Mapping, ou relacionamento objeto-relacional é a técnica que mapeia o uso do banco de dados, convertendo-os em objetos.

Brincando com o ORM

O método save

Execute:

- python manage.py shell
  - from core.models import FeriadoModel
  - f = FeriadoModel(nome="Natal",dia=25,mes=12)
  - f.save()
  - f.pk

O que aconteceu ?

Brincando com o ORM

```
(venv) orlando@rioclaro:/tmp/psd/feriados$ python manage.py shell
Python 3.8.10 (default, Jun 22 2022, 20:18:18)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.5.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from core.models import FeriadoModel

In [2]: f = FeriadoModel(nome="Natal", dia=25, mes=12)

In [3]: f
Out[3]: <FeriadoModel: Natal>

In [4]: f.pk

In [5]: f.save()

In [6]: f.pk
Out[6]: 1
```

Brincando com o ORM

O método save

```
from core.models import FeriadoModel

f.nome = “Páscoa”

f.save()

f2 = FeriadoModel.objects.get(pk=1)

f2.id = None

f2.save()
```

O método **save()** realiza update quando identifica id com algum valor, e realiza um create quando id estiver nulo.

Brincando com o ORM

O método create

```
FeriadoModel.objects.all()

f.delete()

FeriadoModel.objects.all()

f3= FeriadoModel.objects.create(

        nome='Tiradentes',dia=21,mes=4

)
```

O método **create()** salva o registro no banco e retorna o objeto.

Brincando com o ORM

O método get

```
FeriadoModel.objects.all()

f3 = FeriadoModel.objects.get(pk=2)


f4 = FeriadoModel.objects.get(nome__contains='a')
```

O método **get()** retorna apenas uma instância. A segunda forma ocorre erro: MultipleObjectsReturned.

Brincando com o ORM

type(f3) => **Instância do modelo.**

type(FeriadoModel) => **Descreve a tabela**

type(FeriadoModel.objects) => **Manager manipula a tabela**

type(FeriadoModel.objects.all()) => **QuerySet**

Brincando com o ORM

QuerySet

```
qs = FeriadoModel.objects.all()


for feriado in qs:
    print(feriado)


qs = FeriadoModel.objects.all()
qs = qs.filter(nome__contains='a')
qs = qs.filter(nome__contains='P')
qs
```

Brincando com o ORM

QuerySet - continuação

```
qs = FeriadoModel.objects.all().update(dia=15)

qs = FeriadoModel.objects.all()

qs.delete()
```

Comandos do manage.py

Comandos do manage.py

python manage.py flush

Apagar todos os dados do banco de dados

python manage.py inspectdb

Gerar classes baseado em uma inspeção no banco de dados.

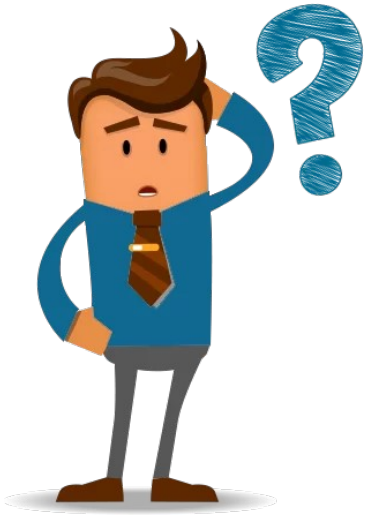
python manage.py makemigrations

Cria um arquivo migrate, arquivos de migração do esquema do banco de dados.

python manage.py migrate

Executa a migração no banco de dados.

Dúvidas

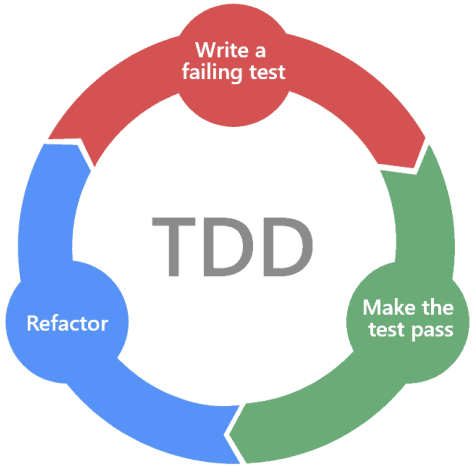


Atividade

Ajustar o projeto, para que exista apenas uma view.

- Esta deve buscar no banco de dados se hoje é feriado ou não, baseado nos feriados cadastrados no banco de dados.

- Se hoje for feriado:
  - Apresente qual feriado é hoje.



Tempo para Atividade

Próximos passos...

- Contribua no github do projeto **Feriado**
  - *Issues* “fáceis”
  - Refatoração

O que podemos melhorar ?

Feedback

- O que você achou que foi bom ?
  - Pontos positivos

- O que você achou que não foi bom ?
  - Pontos de melhorias