Fatec
Araras

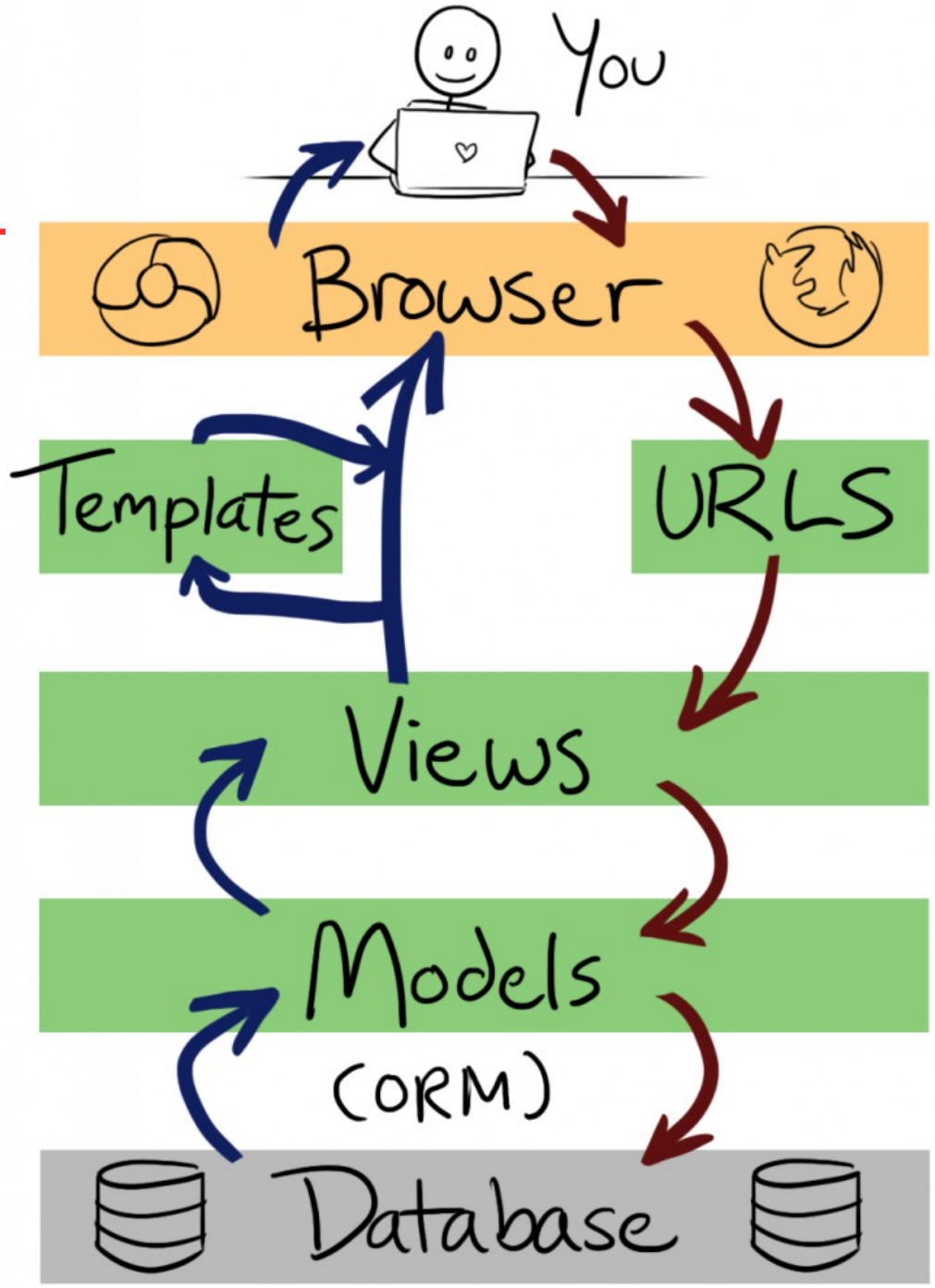
DESENVOLVIMENTO WEB III

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Agenda

- Django
  - Django Rest Framework
- O Projeto

Fluxo MTV



Fatec Araras

Ativando o Ambiente Virtual

Vamos começar ...
No Windows

Fatec Araras

No prompt de comando (cmd), digite:

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

Vamos começar ...
No Linux

Fatec Araras

No terminal, digite:

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

Arquivo requirements.txt

Fatec Araras

pip install -r requirements.txt

Projeto Feriado v. 0.7

Django Rest Framework

Fatec Araras

O Django REST Framework é uma biblioteca que transforma projetos Django em APIs RESTful com segurança, serialização e views especializadas.

Vantagens:

- Serializa modelos Python em JSON/XML
- Cria views com lógica REST (GET, POST, PUT, DELETE)
- Suporte a autenticação e permissões
- Interface web para testes interativos
- Fácil integração com frontend moderno (React, Vue, Angular)

Fonte: https://www.django-rest-framework.org/

Sem fazer uso do Django Rest Framework

Fatec Araras

O Django possui suporte nativo a respostas em JSON, sem precisar instalar o Django REST Framework.

```
from django.http import JsonResponse
from .models import FeriadoModel

def listar_feriados_json(request):
    feriados = FeriadoModel.objects.all().values('id', 'nome', 'dia', 'mes')
    return JsonResponse(list(feriados), safe=False)
```

Sem fazer uso do Django Rest Framework

Fatec Araras

O que você ganha com isso?

- Mais leve: Sem dependência externa
- Didático: Bom para aprender como APIs funcionam
- Flexível: Controle manual sobre a resposta

- Pontos de Atenção
- Sem serialização automática (você faz na mão)
- Sem validação embutida
- Sem autenticação integrada

Django Rest Framework

Fatec Araras

pip install djangorestframework

setting.py (do projeto)

Fatec Araras

```
...

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core.apps.CoreConfig',
    'rest_framework',
]

...
```

serializers.py (do app)

Fatec Araras

```
from rest_framework import serializers
from .models import FeriadoModel

class FeriadoSerializer(serializers.ModelSerializer):
    class Meta:
        model = FeriadoModel
        fields = ['id', 'nome', 'dia', 'mes', 'modificado_em']
```

views.py (do app)

Fatec Araras

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import FeriadoModel
from .serializers import FeriadoSerializer
from django.shortcuts import get_object_or_404


class FeriadoListCreateView(APIView):
    def get(self, request):
        feriados = FeriadoModel.objects.all()
        serializer = FeriadoSerializer(feriados, many=True)
        return Response(serializer.data)


    def post(self, request):
        serializer = FeriadoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

views.py (do app)

Fatec Araras

```
class FeriadoDetailView(APIView):
    def get(self, request, pk):
        feriado = get_object_or_404(FeriadoModel, pk=pk)
        serializer = FeriadoSerializer(feriado)
        return Response(serializer.data)


    def put(self, request, pk):
        feriado = get_object_or_404(FeriadoModel, pk=pk)
        serializer = FeriadoSerializer(feriado, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


    def delete(self, request, pk):
        feriado = get_object_or_404(FeriadoModel, pk=pk)
        feriado.delete()
        return Response(status=status.HTTP_204_NO_CONTENT)
```

urls.py (do app)

Fatec Araras

```
# core/urls.py
from django.urls import path
from .views import FeriadoListCreateView, FeriadoDetailView


urlpatterns = [
    path('api/feriados/', FeriadoListCreateView.as_view(), name='api_feriados_list_create'),
    path('api/feriados/<int:pk>/', FeriadoDetailView.as_view(), name='api_feriados_detail'),
]
```

Testando aplicação

GET

```
curl http://localhost:8000/api/feriados/
```

POST

```
curl -X POST http://localhost:8000/api/feriados/ \
    -H "Content-Type: application/json" \
    -d '{"nome": "Dia da Mentira", "dia": 1, "mes": 4}'
```

Testando aplicação

Fatec Araras

GET

```
curl http://localhost:8000/api/feriados/1/
```

PUT

```
curl -X PUT http://localhost:8000/api/feriados/1/ \
     -H "Content-Type: application/json" \
     -d '{"nome": "Natal", "dia": 25, "mes": 12}'
```

DELETE

```
curl -X DELETE http://localhost:8000/api/feriados/1/
```

Fatec Araras

Para proteger suas rotas de API com autenticação por token no Django REST Framework, você pode usar o **TokenAuthentication**.

setting.py (do projeto)

Fatec Araras

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'rest_framework.authtoken',
]
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ]
}
```

Fatec Araras

```
python manage.py migrate
```

urls.py (do app)

Fatec Araras

Criar rota para obter Token

```
from django.urls import path
from .views import FeriadoListCreateView, FeriadoDetailView
from rest_framework.authtoken.views import obtain_auth_token

urlpatterns = [
    path('api/feriados/', FeriadoListCreateView.as_view(), name='api_feriados_list_create'),
    path('api/feriados/<int:pk>/', FeriadoDetailView.as_view(), name='api_feriados_detail'),
    path('api/token/', obtain_auth_token, name='api_token_auth'),
]
```

python manage.py shell

Fatec Araras

Criar Token

```
from django.contrib.auth.models import User
from rest_framework.authtoken.models import Token

user = User.objects.get(username='seuusuario')
token, created = Token.objects.get_or_create(user=user)
print(token.key)
```

views.py (do app)

Fatec Araras

```
from rest_framework.permissions import IsAuthenticated
from rest_framework.views import APIView
from rest_framework.response import Response
...

class FeriadoListCreateView(APIView):
    permission_classes = [IsAuthenticated]
    def get(self, request):
        feriados = FeriadoModel.objects.all()
        serializer = FeriadoSerializer(feriados, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = FeriadoSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)
```

Testando aplicação

Fatec Araras

GET

```
curl -H "Authorization: Token seu_token_aqui" http://localhost:8000/api/feriados/
```

Testando aplicação

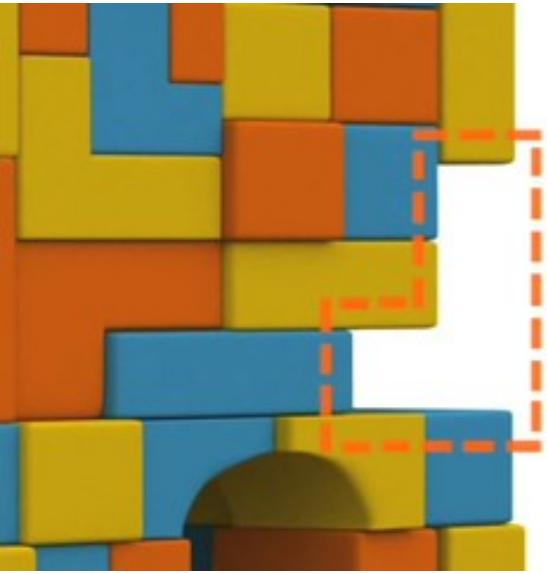
Fatec Araras

Requests

```
import requests


# 1. Obter o token
auth_url = "http://localhost:8000/api/token/"
auth_data = {"username": "admin", "password": "123mudar"}
auth_response = requests.post(auth_url, data=auth_data)
token = auth_response.json().get("token")


# 2. Usar o token
api_url = "http://localhost:8000/api/feriados/"
headers = {"Authorization": f"Token {token}"}
api_response = requests.get(api_url, headers=headers)


print(api_response.status_code)
print(api_response.json())
```

Test it !

Fatec Araras



Testando aplicação

Importe o APIClient:

```
from rest_framework.test import APITestCase, APIClient
from django.urls import reverse
from rest_framework import status
from django.contrib.auth.models import User
from core.models import FeriadoModel
```

Testando aplicação

Fatec Araras

```
class FeriadoAPITests(APITestCase):
    def setUp(self):
        self.client = APIClient()
        # Cria um usuário e token (caso use autenticação)
        self.user = User.objects.create_user(username='admin', password='123')
        self.client.force_authenticate(user=self.user)
        self.feriado = FeriadoModel.objects.create(nome="Natal", dia=25, mes=12


    def test_listar_feriados(self):
        url = reverse('feriado-list')  # Ex: 'api/feriados/'
        response = self.client.get(url)
        self.assertEqual(response.status_code, status.HTTP_200_OK)
        self.assertGreaterEqual(len(response.data), 1)
```

Testando aplicação

Fatec Araras

```
def test_criar_feriado(self):
    url = reverse('feriado-list')
    data = {"nome": "Ano Novo", "dia": 1, "mes": 1}
    response = self.client.post(url, data)
    self.assertEqual(response.status_code, status.HTTP_201_CREATED)

def test_detalhar_feriado(self):
    url = reverse('feriado-detail', args=[self.feriado.id])
    response = self.client.get(url)
    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.assertEqual(response.data["nome"], "Natal")
```

Testando aplicação

Fatec Araras

```
def test_atualizar_feriado(self):
    url = reverse('feriado-detail', args=[self.feriado.id])
    data = {"nome": "Natal Atualizado", "dia": 25, "mes": 12}
    response = self.client.put(url, data)
    self.assertEqual(response.status_code, status.HTTP_200_OK)
    self.feriado.refresh_from_db()
    self.assertEqual(self.feriado.nome, "NATAL ATUALIZADO")

def test_deletar_feriado(self):
    url = reverse('feriado-detail', args=[self.feriado.id])
    response = self.client.delete(url)
    self.assertEqual(response.status_code, status.HTTP_204_NO_CONTENT
    self.assertFalse(FeriadoModel.objects.filter(id=self.feriado.id).exists())
```

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br