Roteiro de atividade prática

Nome: Turma: .

**Título da atividade: Desenvolvimento de APIs com *Django***

**Objetivos**

Desenvolver uma API utilizando *Django* e *Django Rest Framework* (DRF) que permita operações CRUD (create, read, update, delete) para gerenciar produtos em um sistema de inventário. A atividade envolverá a configuração do projeto, a criação de modelos, *views* e *serializers*, além da implementação de autenticação básica.

**Lista de materiais**

* Computador com acesso à internet.
* Caderno para anotações.
* 1 caneta.

**Procedimento experimental**

Você foi contratado para desenvolver uma API que servirá como *back-end* para um aplicativo de gerenciamento de inventário. O sistema deve permitir que os usuários criem, atualizem, visualizem e excluam registros de produtos, além de verificar o estoque em tempo real. Para garantir uma rápida implementação e uma estrutura robusta, você decidiu usar *Django Rest Framework* (DRF) para construir essa API.

Situação fictícia elaborada especialmente para o curso.

**Passo a passo da atividade:**

1. **Configuração do ambiente:**
   * Crie um novo projeto *Django* e uma nova aplicação dentro deste projeto:

django-admin startproject inventory\_system

cd inventory\_system

python manage.py startapp products

**Configuração do *Django Rest Framework*:**

* Instale o *Django Rest Framework*:

pip install djangorestframework

Adicione rest\_framework à lista de INSTALLED\_APPS no arquivo settings.py do projeto:

INSTALLED\_APPS = [

...

'rest\_framework',

'products',

]

**Criação do modelo de produto:**

* No arquivo models.py da aplicação products, defina um modelo para representar um produto no inventário:

from django.db import models

class Product(models.Model):

name = models.CharField(max\_length=255)

description = models.TextField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

stock = models.IntegerField()

def \_\_str\_\_(self):

return self.name

**Criação do *serializer* para o modelo de produto:**

* No arquivo serializers.py, crie um *serializer* para o modelo product que será usado para converter o modelo em JSON e vice-versa:

from rest\_framework import serializers

from .models import Product

class ProductSerializer(serializers.ModelSerializer):

class Meta:

model = Product

fields = '\_\_all\_\_'

**Criação das *views* para a API:**

* No arquivo views.py, crie *views* com base em classe para lidar com operações CRUD usando o *Django Rest Framework*:

from rest\_framework import generics

from .models import Product

from .serializers import ProductSerializer

class ProductListCreate(generics.ListCreateAPIView):

queryset = Product.objects.all()

serializer\_class = ProductSerializer

class ProductRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):

queryset = Product.objects.all()

serializer\_class = ProductSerializer

**Configuração das rotas da API:**

* No arquivo urls.py da aplicação products, defina as rotas para as *views* criadas:

from django.urls import path

from .views import ProductListCreate, ProductRetrieveUpdateDestroy

urlpatterns = [

path('products/', ProductListCreate.as\_view(), name='product-list-create'),

path('products/<int:pk>/', ProductRetrieveUpdateDestroy.as\_view(), name='product-detail'),

]

**Testando a API:**

* Use o *Django* admin para adicionar alguns produtos no banco de dados para teste;
* Acesse os *endpoints* por meio de um cliente HTTP (como Postman ou cURL) para testar as operações CRUD:
  + **Listar todos os produtos:** GET /products/
  + **Criar um novo produto:** POST /products/
  + **Recuperar detalhes de um produto específico:** GET /products/<id>/
  + **Atualizar um produto:** PUT /products/<id>/
  + **Excluir um produto:** DELETE /products/<id>/

**Modelo de produto (models.py):**

from django.db import models

class Product(models.Model):

name = models.CharField(max\_length=255)

description = models.TextField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

stock = models.IntegerField()

def \_\_str\_\_(self):

return self.name

***Serializer* para produto (serializers.py):**

from rest\_framework import serializers

from .models import Product

class ProductSerializer(serializers.ModelSerializer):

class Meta:

model = Product

fields = '\_\_all\_\_'

***Views* para operações CRUD (views.py):**

from rest\_framework import generics

from .models import Product

from .serializers import ProductSerializer

class ProductListCreate(generics.ListCreateAPIView):

queryset = Product.objects.all()

serializer\_class = ProductSerializer

class ProductRetrieveUpdateDestroy(generics.RetrieveUpdateDestroyAPIView):

queryset = Product.objects.all()

serializer\_class = ProductSerializer

**Configuração das rotas (urls.py):**

from django.urls import path

from .views import ProductListCreate, ProductRetrieveUpdateDestroy

urlpatterns = [

path('products/', ProductListCreate.as\_view(), name='product-list-create'),

path('products/<int:pk>/', ProductRetrieveUpdateDestroy.as\_view(), name='product-detail'),

]

|  |
| --- |
|  |
|  |
|  |
|  |