Escola Estadual Professor João Anastácio
CURSO TÉCNICO EM INFORMÁTICA
AVALIAÇÃO DO 4º BIMESTRE DA DISCIPLINA
Programação Web
Laboratório Web

ALUNO (a):_____
PROFESSOR(a):
DATA: / /
VALOR: 07 pontos **TURMA: _____** **Nota: _____**

"A paciência é a fortaleza do débil e a impaciência a debilidade do forte."

(Immanuel Kant)

API de Gerenciamento de Tarefas Simples (To-Do List API)

Passo 1: Configuração e Criação do Projeto

 # Crie e ative o ambiente virtual

python3 -m venv env

source env/bin/activate

 # Instale Django e Django REST Framework

pip install django djangorestframework

 # Crie o projeto (test_project) e o app (tasks)

mkdir test_project

cd test_project

django-admin startproject core **.**

Passo 2: Configuração Básica do Django

#Abra o arquivo:

core/settings.py

INSTALLED_APPS = [

...

'rest_framework',

'tasks'

]

Passo 3: Criação do Modelo (`Task`)

```python
from django.db import models

class Task(models.Model):
    title = models.CharField(max_length=200)
    description = models.TextField(blank=True)
    completed = models.BooleanField(default=False)
    created_at = models.DateTimeField(auto_now_add=True)

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

Passo 4: Migrações

python manage.py makemigrations tasks

python manage.py migrate

Passo 5: Criação do Serializer

Crie o arquivo **`tasks/serializers.py`** para converter os dados do modelo em JSON e vice-versa.

```python
from rest_framework import serializers
from .models import Task

class TaskSerializer(serializers.ModelSerializer):
    class Meta:
        model = Task
        # Inclua todos os campos do modelo na API
        fields = '__all__'
        # Opcional: marque 'created_at' como read-only
        read_only_fields = ('created_at',)
```

Passo 6: Criação do ViewSet (CRUD)

Edite o arquivo `tasks/views.py` para usar o `ModelViewSet`, que automaticamente implementa os métodos List, Create, Retrieve, Update e Destroy (CRUD completo).

```python
from rest_framework import viewsets
from .models import Task
from .serializers import TaskSerializer

class TaskViewSet(viewsets.ModelViewSet):
    # Define o conjunto de dados a ser usado (todas as tarefas)
    queryset = Task.objects.all().order_by('created_at')
    # Define o Serializer a ser usado para conversão JSON
    serializer_class = TaskSerializer
# Create your views here.
```

Passo 7: Definição de Rotas

Edite o `urls.py` do projeto e crie um arquivo `urls.py` para o app. Usaremos o `DefaultRouter` do DRF para gerar todas as rotas CRUD automaticamente a partir do `ViewSet`.

A. Crie `tasks/urls.py`

test_project > tasks > urls.py > ...

```python
from rest_framework.routers import DefaultRouter
from .views import TaskViewSet

# Cria um roteador e registra o ViewSet
router = DefaultRouter()
router.register(r'tasks', TaskViewSet)

# O router gera URLs para todas as operações CRUD
urlpatterns = router.urls
```

B. Edite `core/urls.py`

```python
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    # Inclua as rotas da nossa API no prefixo 'api/v1/'
    path('api/v1/', include('tasks.urls')),
]
```

Configuração do CORS no Django REST Framework

Instale o django-cors-headers usando o pip:

```
pip install django-cors-headers
```

```
INSTALLED_APPS = [
'rest_framework',
'corsheaders', # <--- Adicione aqui
 'tasks',
 ]
```

Adicionar aos MIDDLEWARE

No mesmo arquivo (settings.py), adicione o *middleware* do CORS. Ele deve ser colocado o mais alto possível na lista, **antes** de qualquer *middleware* que possa gerar respostas (como o CommonMiddleware ou CsrfViewMiddleware).

core/settings.py

```
MIDDLEWARE = [
'corsheaders.middleware.CorsMiddleware', # <--- Adicione aqui (Muito importante ser o primeiro ou entre os primeiros!)
'django.middleware.security.SecurityMiddleware',
 # ... outros middlewares
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

4. Definir as Configurações de Origem (CORS Settings)

Ainda no core/settings.py, você deve definir quais *origins* (domínios/portas) terão permissão para acessar sua API.

Para Produção (Permitir Origens Específicas)

#core/settings.py

```
# Permite todas as origens
CORS_ALLOW_ALL_ORIGINS = True
CORS_ALLOW_CREDENTIALS = True
```