Instruções para Atividade Django - Turma de Programação Python

📋 Objetivo da Atividade

Em duplas, vocês vão criar uma plataforma web simples usando Django. Vocês escolherão um tema (como lista de tarefas, catálogo de livros, cadastro de alunos, etc.) e desenvolverão as páginas básicas para gerenciar esse conteúdo.

🎯 O Que Precisamos Entregar

Funcionalidades Obrigatórias:

- Página inicial - Apresentação do sistema
- Página de cadastro - Formulário para adicionar novos itens
- Página de listagem - Mostra todos os itens cadastrados
- Página de detalhes - Mostra informações detalhadas de um item específico
- Navegação - Links para mover entre as páginas
- Armazenamento - Dados salvos em uma lista no arquivo `views.py`

🗂️ Estrutura do Projeto que Vocês Vão Criar

```
text
meu_projeto_django/          # Pasta principal do projeto
├── venv/                     # Ambiente virtual (será criado)
├── core/                     # Configurações do projeto Django
│   ├── __init__.py
│   ├── settings.py           # ← Aqui registramos nosso app
│   ├── urls.py               # ← Aqui incluímos as URLs do app
│   └── wsgi.py
├── minhaapp/                 # App principal (nome que vocês escolherem)
```

```
│   ├── __init__.py
│   ├── admin.py
│   ├── apps.py
│   ├── models.py
│   ├── tests.py
│   ├── views.py            # ← Aqui programamos a lógica principal
│   └── urls.py             # ← Aqui definimos as rotas do app
├── templates/              # Pasta para os arquivos HTML
│   ├── home.html           # ← Página inicial
│   ├── cadastro.html       # ← Página de cadastro
│   ├── lista.html          # ← Página de listagem
│   └── detalhes.html       # ← Página de detalhes
├── manage.py
└── db.sqlite3              (será criado automaticamente)
```

🚀 Passo a Passo do Desenvolvimento

📁 FASE 1: CONFIGURAÇÃO DO AMBIENTE

1.1 Criar a Pasta do Projeto

```
bash
# No terminal, navegue até onde quer salvar o projeto e digite:
mkdir meu_projeto_django
cd meu_projeto_django
```

1.2 Configurar Ambiente Virtual

```
bash
# Criar ambiente virtual
python -m venv venv

# Ativar ambiente virtual
# Windows:
venv\Scripts\activate
# Linux/Mac:
source venv/bin/activate

# O prompt deve mostrar (venv) no início
```

1.3 Instalar Django

```
bash
# Com o ambiente virtual ativado:
pip install django
```

1.4 Criar Projeto Django

```
bash
# Isso cria a estrutura básica do Django
django-admin startproject core
```

1.5 Criar Nosso App

```
bash
# Um app é como um "módulo" do nosso projeto
python manage.py startapp minhaapp
```

📝 Explicação: O Django trabalha com "projetos" (sites completos) e "apps" (funcionalidades específicas). Nosso projeto é o `core` e nosso app é `minhaapp`.

⚙️ FASE 2: CONFIGURAR O DJANGO

2.1 Registrar o App (core/settings.py)

Abra o arquivo `core/settings.py` e procure por `INSTALLED_APPS`. Adicione o nome do nosso app:

```
python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'minhaapp',  # ← ADICIONAR ESTA LINHA
```

```
]
```

2.2 Configurar Templates (core/settings.py)

No mesmo arquivo `settings.py`, procure por `TEMPLATES` e modifique a linha `'DIRS'`:

```
python
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates'],  # ← MODIFICAR ESTA LINHA
        # ... resto do código permanece igual
    },
]
```

2.3 Configurar URLs do Projeto (core/urls.py)

Substitua o conteúdo do arquivo `core/urls.py` por:

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('minhaapp.urls')),  # ← Inclui as URLs do nosso app
]
```

🔗 FASE 3: CRIAR AS ROTAS (URLs)

3.1 Criar Arquivo de URLs do App

Crie um arquivo chamado `urls.py` dentro da pasta `minhaapp/`:

minhaapp/urls.py

```
python
from django.urls import path
```

```
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('cadastro/', views.cadastro, name='cadastro'),
    path('lista/', views.lista, name='lista'),
    path('detalhes/<int:item_id>/', views.detalhes, name='detalhes'),
]
```

📝 Explicação: Aqui mapeamos URLs para funções:

- `''` → Página inicial → função `home()`
- `'cadastro/'` → Página de cadastro → função `cadastro()`
- `'detalhes/1/'` → Mostra detalhes do item com ID 1

🧠 FASE 4: PROGRAMAR A LÓGICA (VIEWS)

4.1 Implementar as Views

Abra `minhaapp/views.py` e substitua por:

```
python
from django.shortcuts import render, redirect

# LISTA onde armazenaremos nossos dados
# ⚠️ ESTES DADOS SÃO PERDIDOS QUANDO O SERVIDOR REINICIA
itens = []

def home(request):
    """Página inicial do site"""
    return render(request, 'home.html')

def cadastro(request):
    """Página de cadastro de novos itens"""
    if request.method == 'POST':
        # Se o formulário foi enviado (método POST)
        nome = request.POST.get('nome')
        descricao = request.POST.get('descricao')

        # Criar um novo item
        novo_item = {
```

```
            'id': len(itens) + 1,      # Gera um ID sequencial
            'nome': nome,
            'descricao': descricao,
            # Adicionem mais campos conforme necessário
        }

        # Adiciona à lista
        itens.append(novo_item)

        # Redireciona para a página de lista
        return redirect('lista')

    # Se for método GET, mostra o formulário vazio
    return render(request, 'cadastro.html')

def lista(request):
    """Página que mostra todos os itens cadastrados"""
    context = {
        'itens': itens  # Passa a lista para o template
    }
    return render(request, 'lista.html', context)

def detalhes(request, item_id):
    """Página que mostra detalhes de um item específico"""
    # Procura o item pelo ID
    item = None
    for i in itens:
        if i['id'] == item_id:
            item = i
            break

    # Se não encontrou, volta para a lista
    if item is None:
        return redirect('lista')

    context = {
        'item': item
    }
    return render(request, 'detalhes.html', context)
```

🎨 FASE 5: CRIAR OS TEMPLATES (HTML)

5.1 Criar Pasta de Templates

Na pasta principal do projeto (onde está o `manage.py`), criem uma pasta chamada `templates`.

5.2 Template: Página Inicial (home.html)

templates/home.html

```
html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistema de Cadastro</title>
    <style>
        /* TODO: Adicionem estilos CSS aqui */
        body { font-family: Arial; margin: 0; padding: 20px; }
        nav { background: #333; padding: 10px; margin-bottom: 20px; }
        nav a { color: white; text-decoration: none; margin-right: 15px; }
        .card { background: #f9f9f9; padding: 20px; border-radius: 5px; }
    </style>
</head>
<body>
    <nav>
        <a href="/">Home</a>
        <a href="/cadastro/">Cadastro</a>
        <a href="/lista/">Lista</a>
    </nav>

    <div class="card">
        <h1>Bem-vindo ao Sistema!</h1>
        <p>Este é o sistema de cadastro desenvolvido por [NOME DA DUPLA].</p>
        <p>Use o menu acima para navegar.</p>
    </div>
</body>
</html>
```

5.3 Template: Página de Cadastro (cadastro.html)

templates/cadastro.html

```
html
<!DOCTYPE html>
<html lang="pt-br">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cadastro - Sistema</title>
    <style>
        /* TODO: Adicionem estilos CSS aqui */
        body { font-family: Arial; margin: 0; padding: 20px; }
        nav { background: #333; padding: 10px; margin-bottom: 20px; }
        nav a { color: white; text-decoration: none; margin-right: 15px; }
        form { max-width: 500px; }
        input, textarea { width: 100%; padding: 8px; margin: 5px 0; }
        button { background: blue; color: white; padding: 10px; border: none; }
    </style>
</head>
<body>
    <nav>
        <a href="/">Home</a>
        <a href="/cadastro/">Cadastro</a>
        <a href="/lista/">Lista</a>
    </nav>

    <h1>Cadastrar Novo Item</h1>

    <form method="POST">
        {% csrf_token %}
        <p>
            <label>Nome:</label>
            <input type="text" name="nome" required>
        </p>
        <p>
            <label>Descrição:</label>
            <textarea name="descricao" rows="4"></textarea>
        </p>
        <button type="submit">Salvar</button>
    </form>
</body>
</html>
```

📝 Importante: A tag `{% csrf_token %}` é obrigatória em formulários Django por segurança!

5.4 Template: Página de Lista (lista.html)

templates/lista.html

```
html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Lista de Itens</title>
    <style>
        /* TODO: Adicionem estilos CSS aqui */
        body { font-family: Arial; margin: 0; padding: 20px; }
        nav { background: #333; padding: 10px; margin-bottom: 20px; }
        nav a { color: white; text-decoration: none; margin-right: 15px; }
        .item { background: #f0f0f0; padding: 10px; margin: 10px 0; }
        .btn { background: green; color: white; padding: 5px 10px; text-decoration:
none; }
    </style>
</head>
<body>
    <nav>
        <a href="/">Home</a>
        <a href="/cadastro/">Cadastro</a>
        <a href="/lista/">Lista</a>
    </nav>

    <h1>Itens Cadastrados</h1>

    {% if itens %}
        {% for item in itens %}
        <div class="item">
            <h3>{{ item.nome }}</h3>
            <p>{{ item.descricao }}</p>
            <a class="btn" href="/detalhes/{{ item.id }}/">Ver Detalhes</a>
        </div>
        {% endfor %}
    {% else %}
        <p>Nenhum item cadastrado ainda.</p>
        <a href="/cadastro/">Cadastrar primeiro item</a>
    {% endif %}
</body>
</html>
```

📝 Explicação do Jinja:

- `{% if itens %}` - Verifica se a lista tem itens
- `{% for item in itens %}` - Loop para cada item na lista
- `{{ item.nome }}` - Exibe o valor do campo "nome" do item

5.5 Template: Página de Detalhes (detalhes.html)

templates/detalhes.html

```
html
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Detalhes - {{ item.nome }}</title>
    <style>
        /* TODO: Adicionem estilos CSS aqui */
        body { font-family: Arial; margin: 0; padding: 20px; }
        nav { background: #333; padding: 10px; margin-bottom: 20px; }
        nav a { color: white; text-decoration: none; margin-right: 15px; }
        .card { background: #f9f9f9; padding: 20px; }
        .btn { background: blue; color: white; padding: 8px 15px; text-decoration:
none; }
    </style>
</head>
<body>
    <nav>
        <a href="/">Home</a>
        <a href="/cadastro/">Cadastro</a>
        <a href="/lista/">Lista</a>
    </nav>

    <div class="card">
        <h1>{{ item.nome }}</h1>
        <p><strong>ID:</strong> {{ item.id }}</p>
        <p><strong>Descrição:</strong> {{ item.descricao }}</p>

        <!-- ADICIONEM MAIS CAMPOS AQUI CONFORME NECESSÁRIO -->

        <div style="margin-top: 20px;">
            <a class="btn" href="/lista/">Voltar para Lista</a>
            <a class="btn" href="/cadastro/">Cadastrar Novo</a>
        </div>
    </div>
</body>
</html>
```

🏃 FASE 6: EXECUTAR O PROJETO

6.1 Executar Migrações

```
bash
# Na pasta do projeto (onde está manage.py)
python manage.py migrate
```

6.2 Iniciar Servidor

```
bash
python manage.py runserver
```

6.3 Acessar o Site

Abram o navegador e visitem: http://127.0.0.1:8000/

💡 Dicas Importantes

Para Personalizar o Projeto:

1. Escolham um Tema:
    - Lista de Tarefas? (campos: tarefa, prioridade, data)
    - Catálogo de Livros? (campos: título, autor, ano)
    - Cadastro de Alunos? (campos: nome, curso, email)
2. Modifiquem a Estrutura de Dados:
   No `views.py`, alterem o dicionário `novo_item` para ter os campos do tema escolhido.
3. Estilizem o CSS:
   Usem a tag `<style>` em cada HTML para deixar bonito.

Comandos Úteis:

```
bash
# Parar o servidor: Ctrl+C no terminal
```

```
# Reiniciar servidor:
python manage.py runserver

# Desativar ambiente virtual (quando terminar):
deactivate
```

✅ Checklist Final

- Ambiente virtual criado e ativado
- Django instalado
- Projeto e app criados
- App registrado no settings.py
- URLs configuradas
- Views implementadas
- Templates criados (4 páginas)
- Navegação funcionando
- Formulário de cadastro salvando
- Lista mostrando itens
- Página de detalhes funcionando
- CSS básico adicionado

🆘 Em Caso de Dúvidas

1. Erro de Importação: Verifiquem se o nome do app está correto no `settings.py`
2. Página não encontrada (404): Verifiquem as URLs no `urls.py`
3. Template não encontrado: Verifiquem se os arquivos HTML estão na pasta `templates/`
4. Formulário não funciona: Verifiquem se tem `{% csrf_token %}` no formulário

Bom trabalho, pessoal! 🚀