

Desenvolvimento de Webapps em Django

- **Criação de ambiente de desenvolvimento**

conda create -n aula_django python=3.9

conda activate aula_django

- **Instalação do Framework Django**

pip install Django

- **Criação de slugs automáticos a partir de um campo do Model**

pip install django-autoslug

- **Criação de campos de timestamp:**

Biblioteca que auxilia na criação de Models que utilizam data e hora de criação e modificação.

pip install django-model-utils

- **Biblioteca que auxilia no shell mais iterativo**

pip install ipython

- **Biblioteca Pillow**

Biblioteca necessária em projetos que irão trabalhar com imagens

pip install pillow

Outras funcionalidades para aumentar a produtividade:

pip install autopep8

pip install pylint

- **Instalação de requisitos de projeto:**

pip install -r requirements.txt

flag -r (--requirements) : requisitos a serem instalados

ctrl+shift+p

select linter -> pylint

- **Criação de projeto:**

django-admin startproject store (ou django-admin startproject store . #para não criar a segunda pasta)

- **Criação de apps:**

Primeiro app irá tratar da apresentação geral dos produtos

django-admin startapp pages (ou python manage.py startapp pages)

Segundo app irá tratar dos registros dos usuários

django-admin startapp users (ou python manage.py startapp users)

Terceiro app irá tratar dos registros dos produtos

django-admin startapp products (ou python manage.py startapp products)

Vamos criar mais apps posteriormente...

Antes de iniciar a programação do projeto e criar os models, vamos entender um pouco sobre o mapeamento de urls e templates no Django:

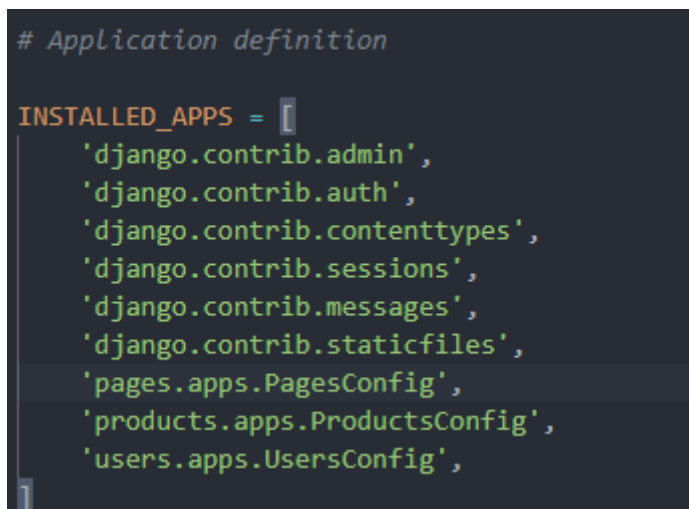
Primeiro passo é incluir nossos apps no projeto

Em settings.py -> INSTALLED_APPS, adicione:

'pages.apps.PagesConfig',

'products.apps.ProductsConfig',

'users.apps.UsersConfig',



```
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'pages.apps.PagesConfig',
    'products.apps.ProductsConfig',
    'users.apps.UsersConfig',
]
```

Agora vamos criar os arquivos urls.py nas pastas de cada app, assim teremos:

Projeto: store -> urls.py

App: pages -> urls.py

App: users -> urls.py

App: products -> urls.py

Na pasta do projeto store, em urls.py:

Faça a importação do include e adicione as rotas para as urls de cada app:

path('pages/', include('pages.urls')),

path('users/', include('users.urls')),

path('products/', include('products.urls')),

```

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

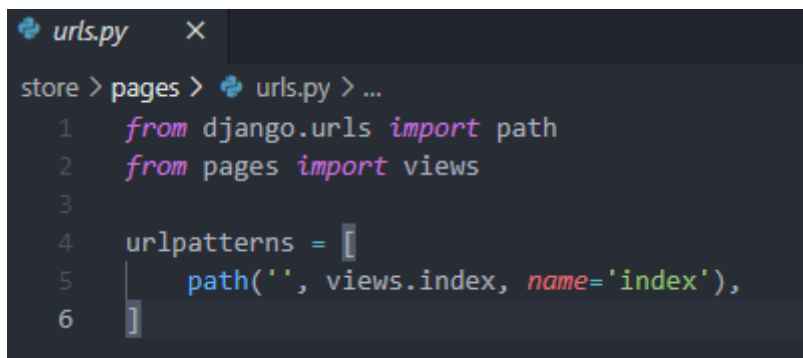
# fiz a importação para apresentar a duplicação de chamada a mesma página
from pages import views

urlpatterns = [
    path('admin/', admin.site.urls),

    # Essa linha está chamando um método em views dentro do app pages,
    # duplicando a chamada que está em pages.urls.py
    path('', views.index, name='index'),
    path('pages/', include('pages.urls')),
    path('users/', include('users.urls')),
    path('products/', include('products.urls')),
]

```

Dentro de urls.py de cada app crie um padrão de chamadas de urls



```

store > pages > urls.py > ...
1  from django.urls import path
2  from pages import views
3
4  urlpatterns = [
5      path('', views.index, name='index'),
6  ]

```

Procure ir testando seu projeto (subindo o server) ao longo das modificações, isso facilita na hora de encontrar problemas de codificação.

No app pages, em views.py, inclua o seguinte código:

```

from django.shortcuts import render

from django.http import HttpResponse

# Create your views here.

def index(request):

    return HttpResponse('Olá Mundo!')

```

```
views.py X
store > pages > views.py > ...
1  from django.shortcuts import render
2  from django.http import HttpResponse
3
4  # Create your views here.
5  def index(request):
6      return HttpResponse('Olá Mundo!')
```

Agora acesse:

<http://127.0.0.1:8000/>

<http://127.0.0.1:8000/pages/>

Você verá o mesmo código, pois tanto o urls.py (projeto store) quanto o urls.py (app pages) estão chamando o mesmo método index de view.py (app pages)

- **Criação e configuração de pastas para templates:**

No projeto store, em settings.py, vamos incluir caminhos para as pastas de templates e arquivos estáticos para funcionar de forma modular.

Será necessário importar a biblioteca os e utilizar métodos da mesma:

```
import os
```

```
TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')
```

```
from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
# Não siga a informação acima, pois em um server baseado em sistema Windows não irá funcionar.

BASE_DIR = Path(__file__).resolve().parent.parent
# print(__file__)
# print(BASE_DIR)

TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')
# print(TEMPLATE_DIR)
```

```

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [TEMPLATE_DIR,],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

```

- Criação e configuração de pastas para static files:

```

from pathlib import Path
import os

# Build paths inside the project like this: BASE_DIR / 'subdir'.
# Não siga a informação acima, pois em um server baseado em docker

BASE_DIR = Path(__file__).resolve().parent.parent
# print(__file__)
# print(BASE_DIR)

TEMPLATE_DIR = os.path.join(BASE_DIR, 'templates')
# print(TEMPLATE_DIR)

STATIC_DIR = os.path.join(BASE_DIR, 'static')
# print(STATIC_DIR)

STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
print(STATIC_ROOT)

```

```

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [
    STATIC_DIR,
]

```

Agora que as configurações para utilização de páginas html e arquivos estáticos estão concluídas, vamos criar nossas primeiras páginas html, rotarizar corretamente e criar métodos em views para renderizar as páginas para os usuários.

Primeiramente vamos consertar a rota no projeto store -> urls.py

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

# fiz a importação para apresentar a duplicação de chamada a mesma página
# from pages import views

urlpatterns = [
    path('admin/', admin.site.urls),

    # Essa linha está chamando um método em views dentro do app pages,
    # duplicando a chamada que está em pages.urls.py
    # path('', views.index, name='index'),
    path('', include('pages.urls')),
    path('users/', include('users.urls')),
    path('products/', include('products.urls')),
]
```

Agora vamos criar rotas nos arquivos urls.py dos apps:

```
store > pages > urls.py > ...
1  from django.urls import path
2  from pages import views
3
4  app_name = 'pages'
5
6  urlpatterns = [
7      path('', views.index, name='index'),
8      path('about/', views.about, name='about'),
9  ]
```

```
store > products > urls.py > ...
1  from django.urls import path
2  from products import views
3
4  urlpatterns = [
5      path('', views.index, name='index'),
6  ]
```

```
store > users > urls.py > ...
1  from django.urls import path
2  from users import views
3
4  urlpatterns = [
5      path('', views.index, name='index'),
6  ]
```

Nesse momento crie os arquivos html:

```
store > templates > base.html > html > body
14  <title>{% block title %} Projeto E-commerce Django {% endblock title %}</title>
15  </head>
16
17  <body>
18      <nav class="navbar navbar-expand-sm py-3 border-bottom navbar-light">
19          <div class="container">
20              <a href="{% url 'pages:index' %}" class="navbar-brand">
21                  <span class="font-weight-bold h3">Loja Django UCAM</span>
22              </a>
23
24              <button class="navbar-toggler" data-toggle="collapse" data-target="#navbarCollapse">
25                  <span class="navbar-toggler-icon"> </span>
26              </button>
27              <div class="collapse navbar-collapse" id="navbarCollapse">
28                  <ul class="navbar-nav ml-auto">
29                      <li class="nav-item">
30                          <a href="{% url 'pages:about' %}" class="nav-link lead mr-4 font-weight-bold">Sobre Nós</a>
31                      </li>
32                  </ul>
33              </div>
34          </div>
35      </nav>
36      {% block content %}
37      {% endblock content %}
38
39      <!-- Optional JavaScript -->
40      <!-- jQuery first, then Popper.js, then Bootstrap JS -->
41      <script src="{% static 'js/jquery-3.2.1.slim.min.js' %}"></script>
42      <script src="{% static 'js/popper.min.js' %}"></script>
43      <script src="{% static 'js/bootstrap.min.js' %}"></script>
44  </body>
45
46  </html>
```

```
store > templates > index.html > div.container.my-3
1  {% extends 'base.html' %}
2
3  {% block content %}
4      <div class="container my-3">
5          <h1> {{intro}} </h1>
6          <p> {{info}} </p>
7          <p> {{intro2}} </p>
8      </div>
9
10  {% endblock content %}
```



```

store > templates > <> about.html > ...
1  {% extends 'base.html' %}
2
3  {% load static %}
4
5  {% block title %}Sobre Nós{% endblock title %}
6
7  {% block content %}
8
9  <div class="container my-3">
10     <h1 class="font-weight-bold">{{intro}}</h1>
11     <p>{{info}}</p>
12     
13 </div>
14
15 {% endblock content %}

```


Finalizamos com o views.py, do app pages:

```

store > pages > views.py > ...
1  from django.shortcuts import render
2  # from django.http import HttpResponse
3
4  # Create your views here.
5  def index(request):
6      context = {
7          'intro': 'Bem vindos a aula de Desenvolvimento Web I',
8          'info': 'Turma 2021-1',
9          'intro2': 'Mastering Django Web Dev'
10     }
11     return render(request, 'index.html', context=context)
12
13  def about(request):
14      context = {
15          'intro': 'Sobre nós',
16          'info': 'Em construção',
17     }
18     return render(request, 'about.html', context=context)

```

Como estamos utilizando arquivos estáticos, é necessário salvá-los nas pastas referente a cada arquivo:

- static
 - css
 - # bootstrap.min.css
 - # my_stylesheet.css
 - images
 -  under_construction.jpg
 - js
 - JS bootstrap.min.js
 - JS jquery-3.2.1.slim.min.js
 - JS popper.min.js