

UNIVERSIDADE FEDERAL DO PIAUÍ
CIÊNCIA DA COMPUTAÇÃO

autores: Oscar William Nunes De Carvalho
Rayanne Ellen Lopes Figueiredo
Guilherme Eduardo Almeida Martinelis
Pedro Henrique Sousa De Oliveira

TUTORIAIS
PYTHON E DJANGO

TERESINA
2023

Tutorial T1

INSTALAÇÃO E CONFIGURAÇÃO DAS FERRAMENTAS DE DESENVOLVIMENTO.

- Linguagem e framework escolhido

Linguagem: Python

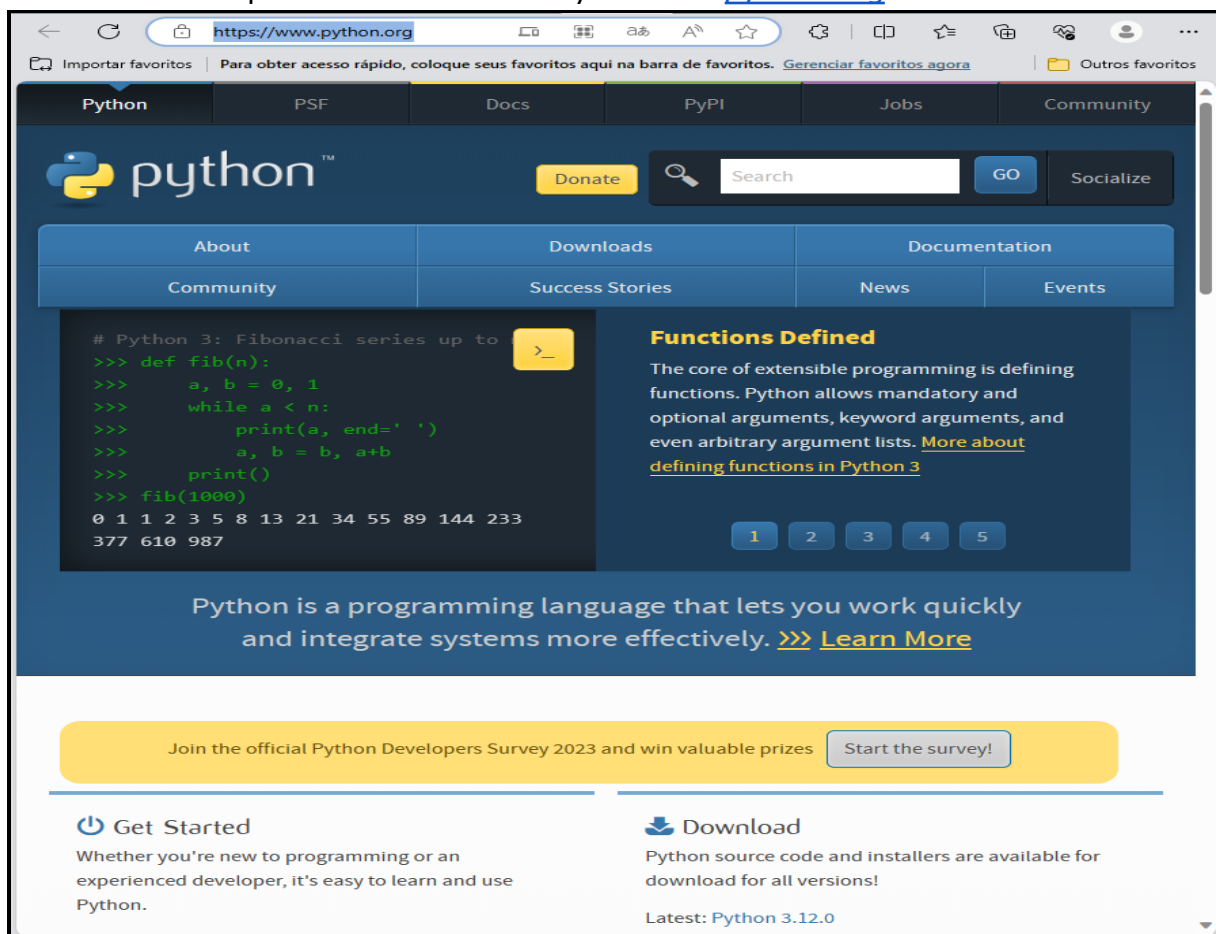
Framework: Django

IDE: Vscode

- Como instalar python no windows 11

1. Faça o download do Python:

- a. Vá para o site oficial do Python em [python.org](https://www.python.org).



b. Clique em "Downloads" no menu principal.



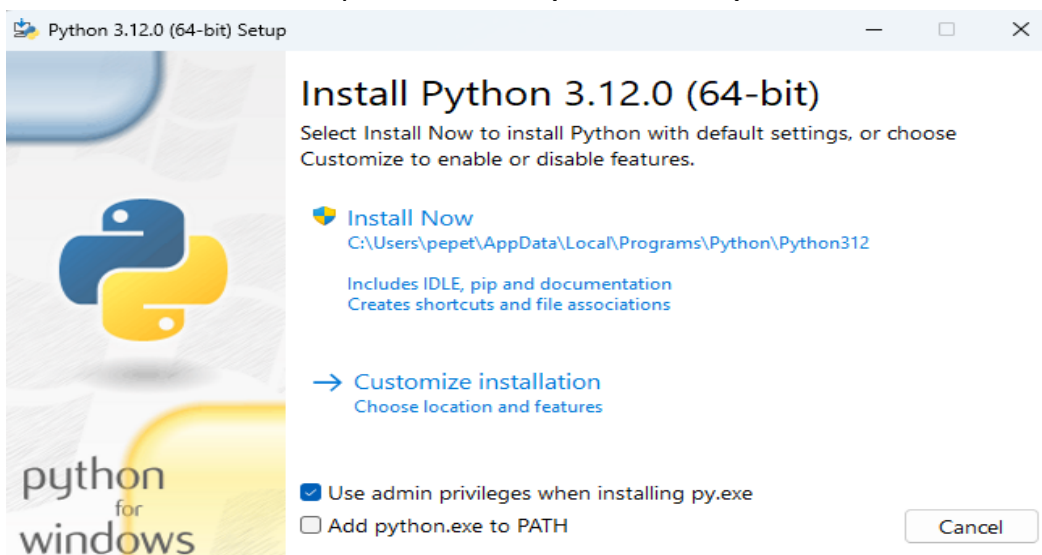
2. Baixe o Instalador

- a. Clique no botão de download para baixar o instalador do Python. O arquivo terá um nome como python-3.x.x.exe, onde "x.x" é a versão específica do Python.

Windows installer (64-bit)	Windows	Recommended	32ab6a1058dfbde76951b7aa7c2335a6	26507904	SIG	.sigstore
--	---------	-------------	----------------------------------	----------	---------------------	---------------------------

3. Execute o Instalador

- a. Abra o arquivo baixado (o instalador) e execute-o.

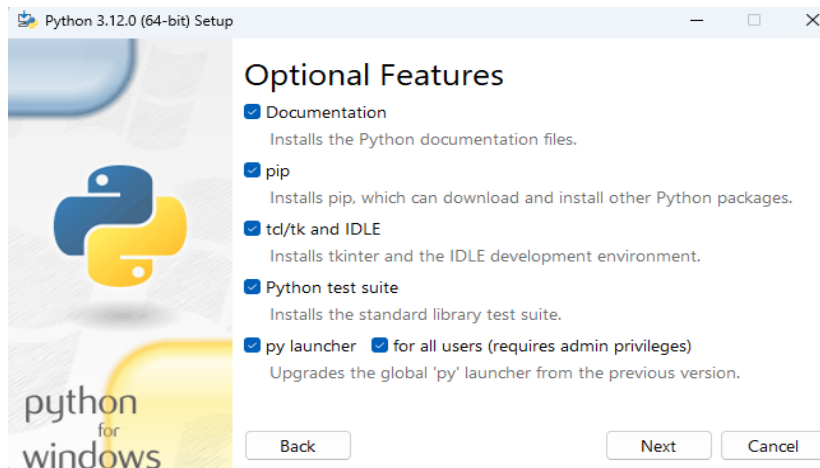


4. Configure a Instalação:

- a. Na primeira tela do instalador, certifique-se de marcar a caixa que diz "Add Python to PATH". Isso facilita o uso do Python a partir da linha de comando.

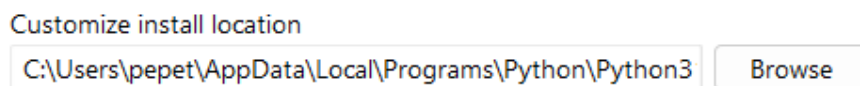
5. Selecione Componentes:

- a. Em seguida, você verá uma tela com diferentes opções. A menos que você tenha um motivo específico para alterar essas configurações, pode deixá-las como estão e clicar em "Next".



6. Caminho de Instalação

- a. Escolha o local onde o Python será instalado. O caminho padrão é geralmente adequado para a maioria dos usuários.

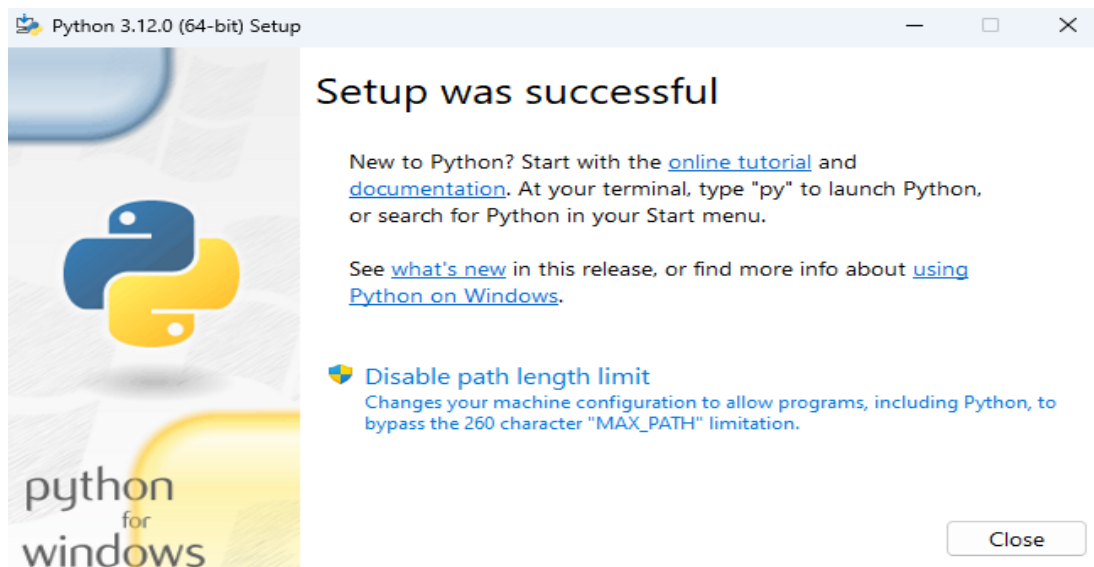


7. Instalação

- a. Clique no botão "Install" para iniciar o processo de instalação. Isso pode levar alguns minutos.

8. Conclua a Instalação

- a. Após a instalação, você verá uma tela indicando que o Python foi instalado com sucesso. Certifique-se de marcar a caixa que diz "Disable path length limit" se ela estiver disponível.



9. Verifique a Instalação

- Abra o Prompt de Comando (CMD) ou o PowerShell e digite o seguinte comando para verificar se o Python foi instalado corretamente e exibir a versão instalada:
- Digite no Prompt: `python --version`
- Caso a instalação tenha sido concluída com sucesso a versão instalada aparecerá.

```
Microsoft Windows [versão 10.0.22621.2428]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\pepet>python --version
Python 3.11.1

C:\Users\pepet>
```

• Como instalar o Django no windows 11:

1. Instale o Django

- Execute o seguinte comando no CMD para instalar o Django usando o pip: `pip install Django`

```
C:\Users\pepet>pip install django
```

- Isso baixará e instalará o Django e suas dependências

2. Verifique a Instalação

- Após a instalação, você pode verificar se o Django foi instalado corretamente usando o comando: `python -m django --version`

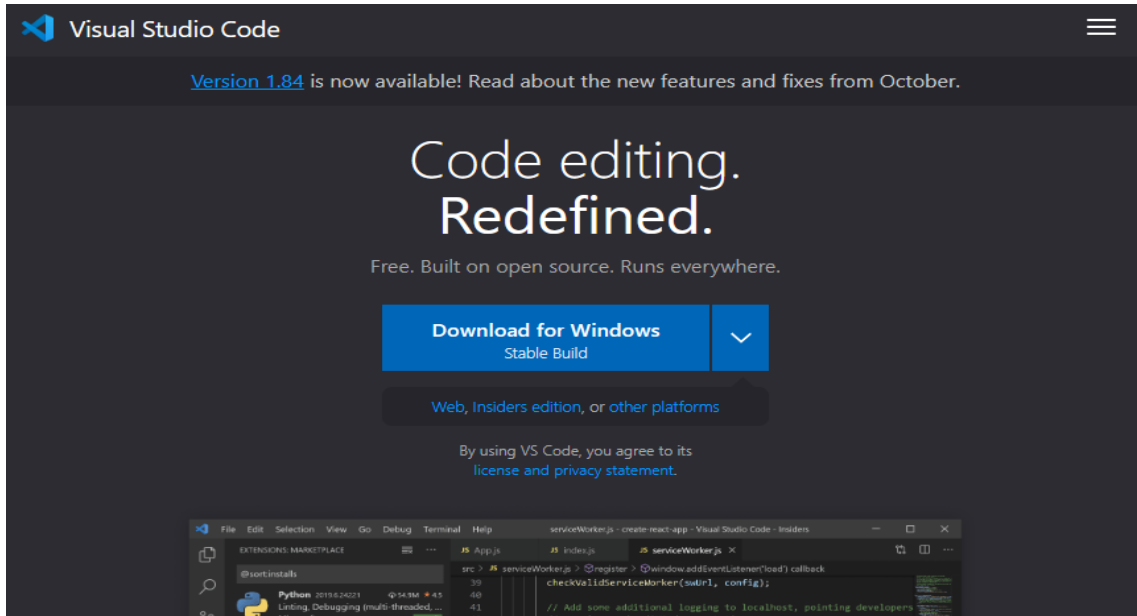
```
PS C:\Users\pepet> python -m django --version
4.2.7
```

- Isso deve exibir a versão do Django que foi instalada

- **Como instalar o Vscode:**

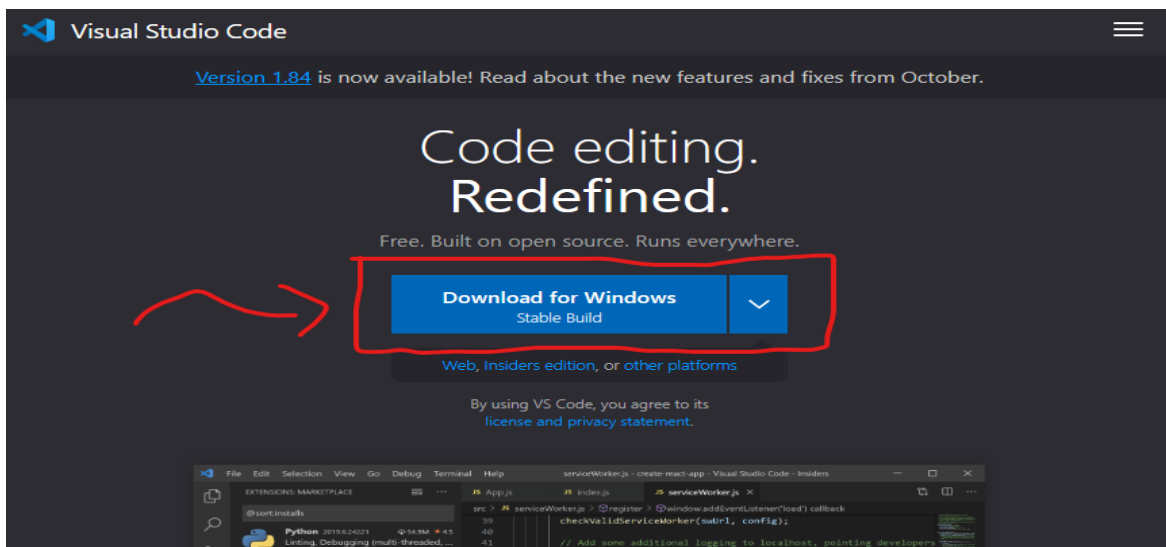
- 1. Acesse o Site do VSCode:**

- a. Abra o navegador da web e vá para o site oficial do Visual Studio Code: <https://code.visualstudio.com/>.



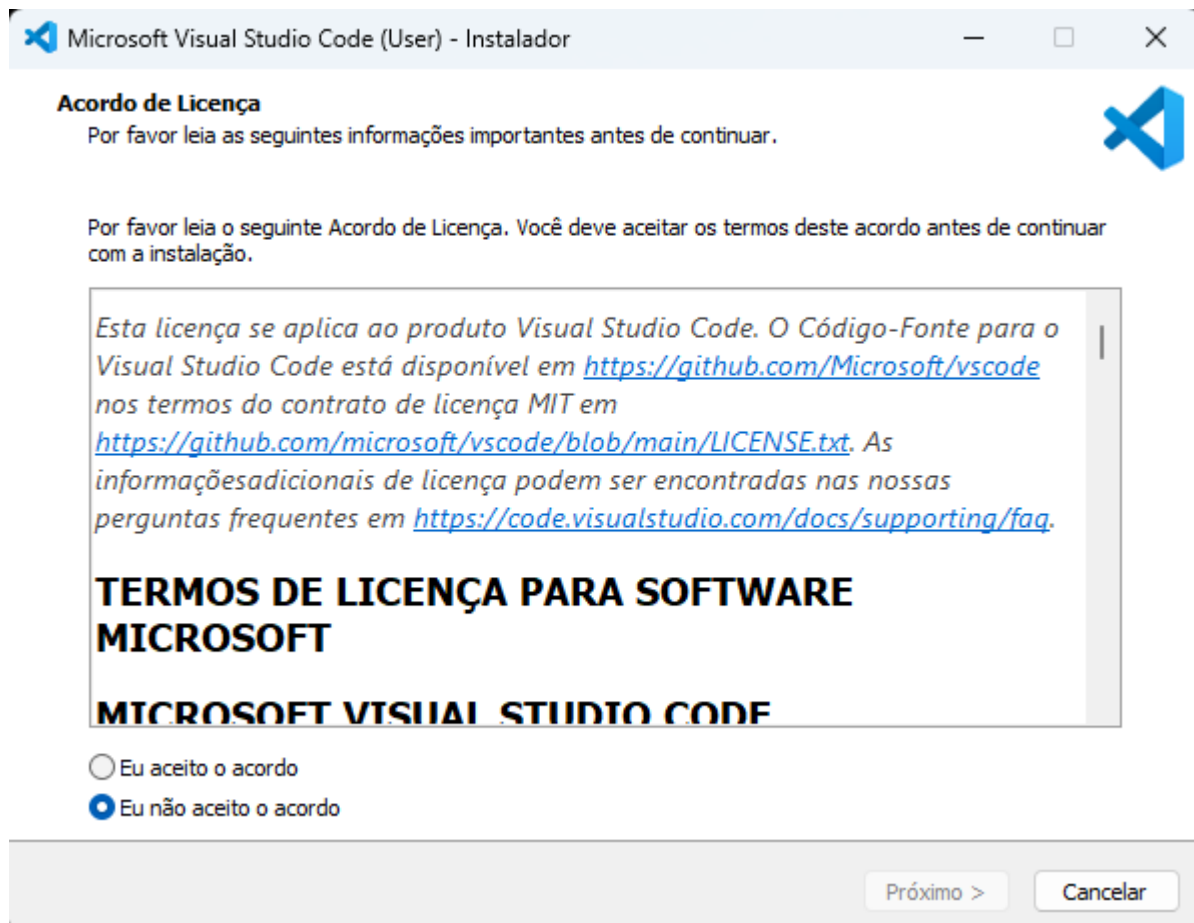
- 2. Download do Instalador:**

- a. Na página inicial, você verá um botão para fazer o download do VSCode. Clique neste botão para baixar o instalador.



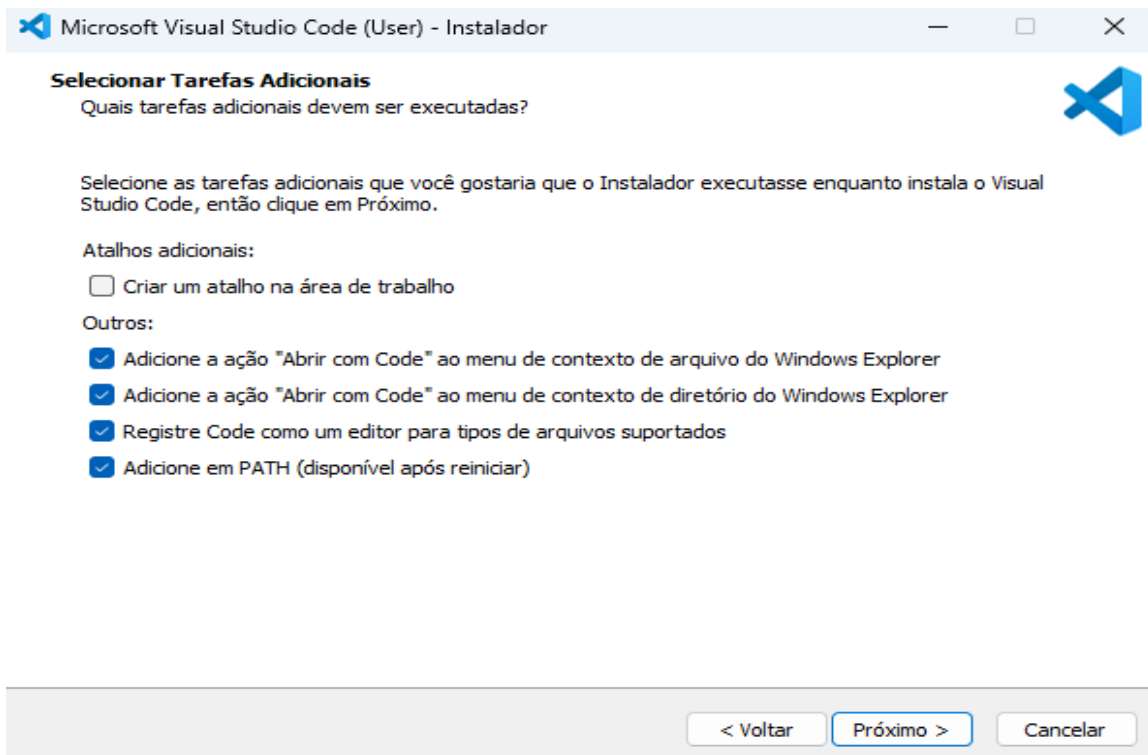
3. Execute o Instalador:

- a. Após o download, execute o arquivo de instalação que foi baixado. O arquivo terá uma extensão .exe.



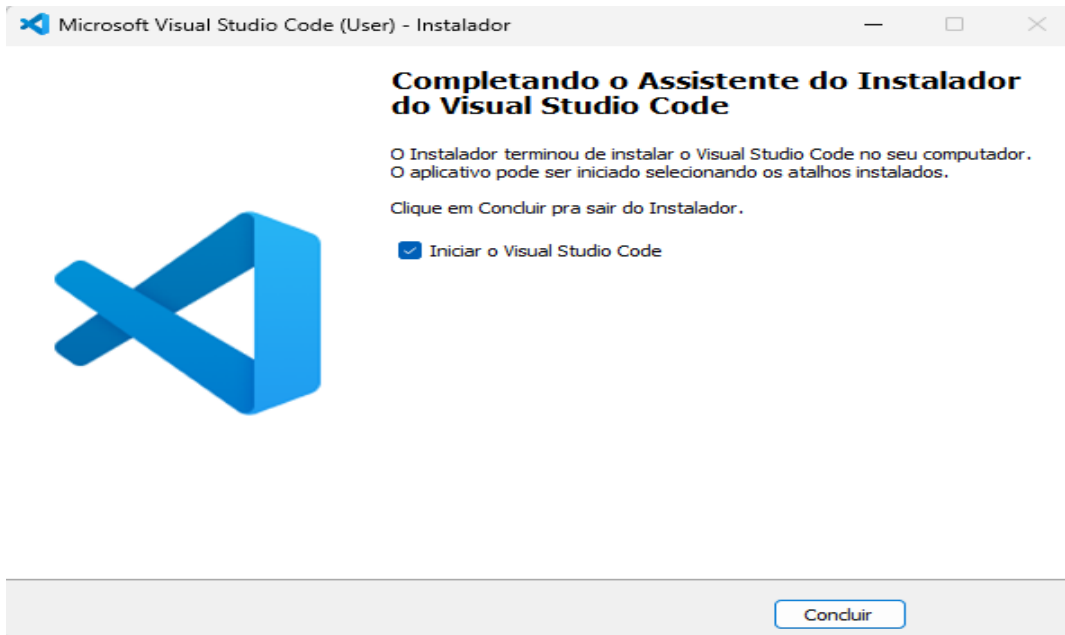
4. Configurações de Instalação:

- a. Durante a instalação, você pode escolher as opções desejadas, como criar atalhos na área de trabalho e adicionar o VSCode ao menu de contexto do Windows. Escolha as opções que melhor atendam às suas preferências.



5. Instalação Concluída:

- Após a conclusão da instalação, você verá uma mensagem indicando que o Visual Studio Code foi instalado com sucesso.



Tutorial T2

CRIANDO UMA APLICAÇÃO WEB BÁSICA MOSTRANDO UM FORMULÁRIO HTML, POR EXEMPLO: EXIBIR FORMULÁRIO DE CADASTRO DE USUÁRIO.

- Framework escolhido

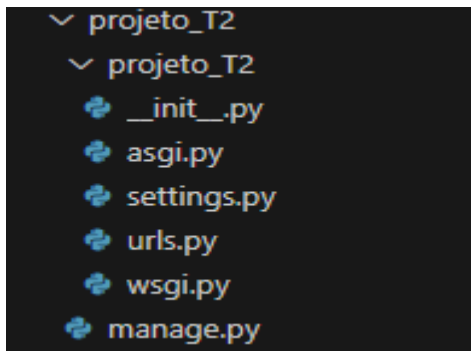
Framework: Django

IDE: Vscode

- Como criar um formulário usando Html, Django:

1. Inicie um projeto no django:

- a. Para isso use: 'django-admin startproject nome_do_projeto' no terminal e diretório aonde quer iniciar o projeto.



- b. Vá para o novo diretório com o comando: 'cd nome_do_projeto' e digite 'python manage.py migrate'
- c. Confirme que tudo está em ordem com o comando: 'python manage.py runserver' e acesse <http://127.0.0.1:8000/>



A instalação funcionou com sucesso! Parabéns;!

Você está vendo esta página porque `DEBUG=True` está no arquivo de configurações e você não configurou nenhuma URL.



Documentação do Django

Tópicos, referências, & how-to's

<https://www.djangoproject.com>



Tutorial: Um aplicativo de sondagem

Introdução ao Django

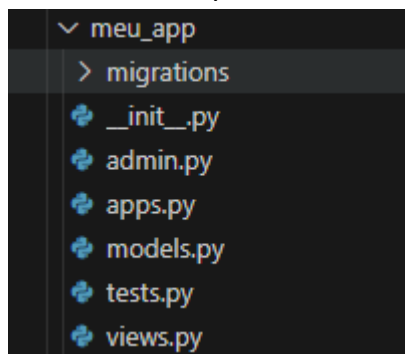


Comunidade Django

Conecte-se, obtenha ajuda ou contribua

2. Crie um aplicativo no django:

- Use o comando: `'python manage.py startapp nome_do_app'` para criar seu aplicativo



3. Defina um models.py:

- Dentro do diretório meu_app altere o arquivo models.py:

```
from django.db import models

class Usuario(models.Model):
    usuario = models.CharField(max_length=100)
    nome_completo = models.CharField(max_length=255)
    email = models.EmailField()
    senha = models.CharField(max_length=255)
```

4. Crie uma view em views.py:

- a. Em seguida, crie uma view que renderiza o formulário. Abra o arquivo views.py no diretório meu_app e adicione o seguinte:

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import Usuario

def cadastrar_usuario(request):
    if request.method == 'POST':
        usuario = request.POST['usuario']
        nome_completo = request.POST['nome_completo']
        email = request.POST['email']
        senha = request.POST['senha']

        novo_usuario = Usuario(usuario=usuario,
                                nome_completo=nome_completo, email=email, senha=senha)

        # Imprimir os dados no terminal
        print(f'Usuario: {novo_usuario.usuario}')
        print(f'Nome completo: {novo_usuario.nome_completo}')
        print(f'Email: {novo_usuario.email}')
        print(f'Senha: {novo_usuario.senha}')

    return render(request, 'cadastrar_usuario.html')
```

5. Crie um template HTML em templates do aplicativo:

- No diretório do seu aplicativo (meu_app), crie uma pasta chamada templates e dentro dela, crie um arquivo chamado cadastrar_usuario.html com o seguinte conteúdo:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cadastrar Usuário</title>
</head>
<body>
  <h2>Cadastro de Usuário</h2>
  <form method="post" action="{% url 'cadastrar_usuario' %}">
    {% csrf_token %}
    <label for="usuario">Usuário:</label>
    <input type="text" name="usuario" required><br>

    <label for="nome_completo">Nome Completo:</label>
    <input type="text" name="nome_completo" required><br>

    <label for="email">E-mail:</label>
    <input type="email" name="email" required><br>

    <label for="senha">Senha:</label>
    <input type="password" name="senha" required><br>

    <input type="submit" value="Cadastrar">
  </form>
</body>
</html>
```

6. Configure a URL para a view:

- Finalmente, configure a URL para a view que você criou. Abra o arquivo urls.py no diretório do seu aplicativo (meu_app) e adicione as seguintes linhas:

```
from django.urls import path
from .views import cadastrar_usuario

urlpatterns = [
    path('', cadastrar_usuario, name='cadastrar_usuario'),
]
```

- b. Agora, inclua essas URLs no arquivo `urls.py` do seu projeto principal (`meu_projeto/urls.py`):

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('meu_app.urls')),
]
```

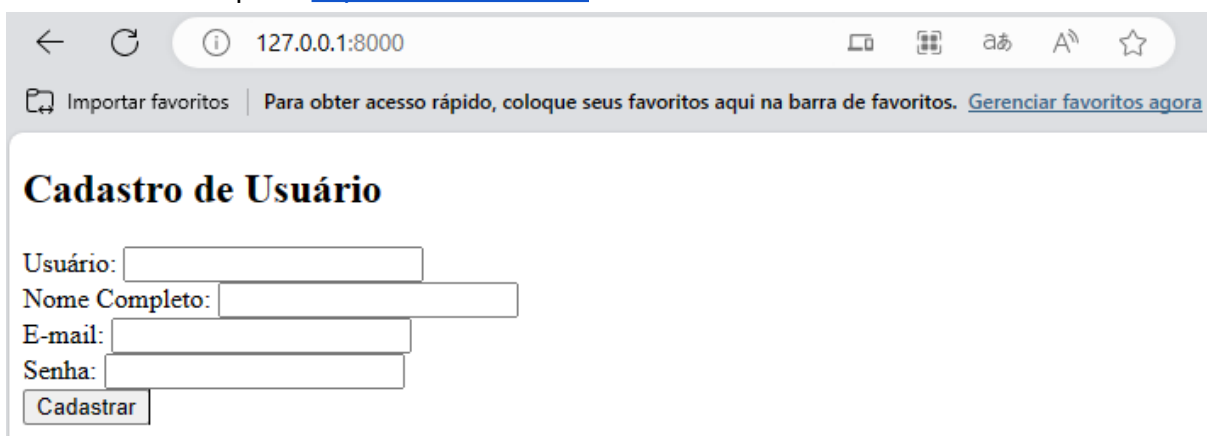
7. Modifique `settings.py`:

- a. adicione `'meu_app'` (nome que você deu ao aplicativo) em `INSTALLED_APPS`:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'meu_app',
]
```

8. Execute o servidor de desenvolvimento:

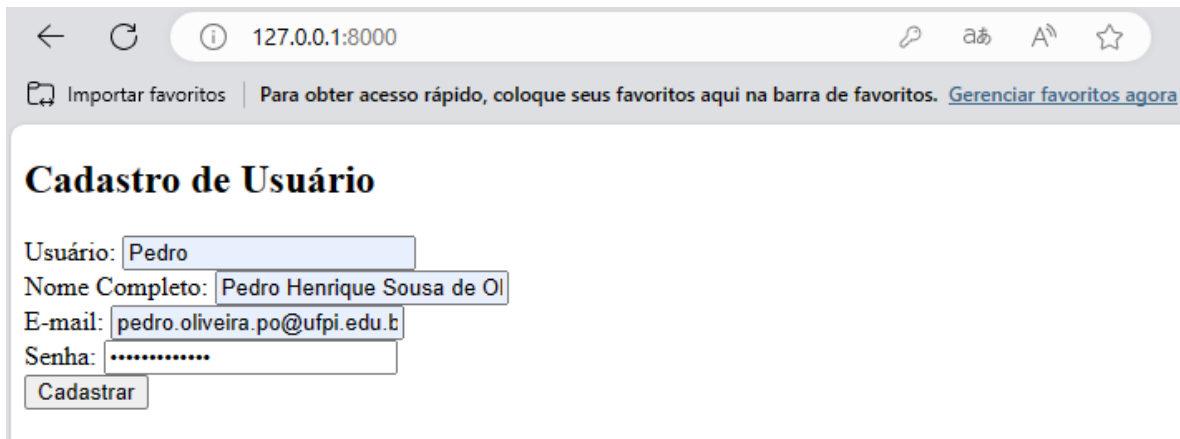
- a. Inicie o servidor de desenvolvimento do Django com o comando: `'python manage.py runserver'` por padrão a porta utilizada será a 8000
- b. Va para: <http://127.0.0.1:8000/>



The screenshot shows a web browser window with the address bar displaying `127.0.0.1:8000`. The page title is "Cadastro de Usuário". The form contains four input fields: "Usuário:", "Nome Completo:", "E-mail:", and "Senha:". Below the "Senha:" field is a "Cadastrar" button. The browser's address bar also shows a link to "Gerenciar favoritos agora".

9. Testando o formulário:

a. Preencha o formulário e clique em cadastrar:



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000'. The page title is 'Cadastro de Usuário'. The form contains the following fields and values:

- Usuário: Pedro
- Nome Completo: Pedro Henrique Sousa de Oliveira
- E-mail: pedro.oliveira.po@ufpi.edu.br
- Senha: (masked)

A 'Cadastrar' button is located at the bottom of the form.

b. No terminal será possível ver as informações cadastradas

```
Usuario: Pedro
Nome completo: Pedro Henrique Sousa de Oliveira
Email: pedro.oliveira.po@ufpi.edu.br
Senha: senhatesteste123
```

Tutorial T3

INSTALAÇÃO E CONFIGURAÇÃO DO BANCO DE DADOS

Linguagem: Python

Framework: Django

Banco de dados: SQLite3

Será mostrado como está configurado o banco de dados com o SQLite3.

Ao executar o comando migrate presente no início de T2 e o comando makemigrations será gerado um arquivo db.sqlite3 na pasta base do projeto django. Esse é o banco de dados padrão que será gerada pela framework.

No settings.py da pasta do nosso projeto, vemos o diretório para onde o nosso banco de dados está atualmente configurado:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Uma vez que tiver feito todos esses ajustes, execute o par de comandos para atualizar o banco de dados do projeto com o novo aplicativo de imagens: “python manage.py makemigrations”, seguido por “python manage.py migrate”. Com isso, a parte de upload de imagens está pronta.

Agora vamos tratar dos templates. Além de alguns ajustes já mostrados anteriormente que também abrangem os templates, também é necessário editar o arquivo settings.py com a pasta do projeto onde eles se localizam.

```
class Imagem(models.Model):
    imagem = models.ImageField(upload_to="imgs/", null=True, blank=True)
    descricao = models.TextField(blank=True)
```

Uma vez criada a tabela, precisamos definir como ela será preenchida. Geralmente será realizado por meio de diferentes formulários existentes nos aplicativos do nosso projeto.

Por exemplo, nossa tabela de imagens é preenchida por um formulário presente no nosso aplicativo de imagens:

```
class ImagemForm(forms.ModelForm):
    class Meta:
        model = Imagem
        fields = [
            'imagem',
            'descricao'
        ]
```

Assim que essas estruturas estejam definidas e prontas, temos que realizar os dois comandos a seguir:

“python manage.py makemigrations”, que irá verificar a integridade das mudanças realizadas no projeto, nos avisando o que irá ser adicionado no banco de dados (ou se houve algum problema/não existem mudanças para serem realizadas).

“python manage.py migrate” que confirmará as mudanças realizadas.

Tutorial T4

Tutorial de como persistir e recuperar dados de um usuário no Banco de Dados escolhido integrado a aplicação web (servidor).

Linguagem: Python

Framework: Django

Banco de dados: SQLite3

Passo a passo:

1. Em um arquivo .py defina um modelo para representar os dados do usuário:

```
from django.db import models

class UserProfile(models.Model):
    username = models.CharField(max_length=50)
    email = models.EmailField()
    bio = models.TextField(blank=True, null=True)

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


2. Execute o seguinte comando para criar o banco de dados SQLite e aplicar as migrações:

```
python manage.py makemigrations  
python manage.py migrate
```

3. Crie um formulário no arquivo forms.py para coletar dados do usuário:

```
from django import forms  
from .models import UserProfile  
  
class UserProfileForm(forms.ModelForm):  
    class Meta:  
        model = UserProfile  
        fields = ['username', 'email', 'bio']
```

4. No arquivo views.py, crie uma exibição para processar o formulário e exibir os dados do usuário:

```
from django.shortcuts import render, redirect  
from .forms import UserProfileForm  
from .models import UserProfile  
  
def user_profile(request):  
    if request.method == 'POST':  
        form = UserProfileForm(request.POST)  
        if form.is_valid():  
            form.save()  
            return redirect('user_profile')  
    else:  
        form = UserProfileForm()  
  
    profiles = UserProfile.objects.all()  
  
    return render(request, 'user_profile.html', {'form': form, 'profiles': profiles})
```

5. Configure as URLs no arquivo urls.py do seu aplicativo:

```
from django.urls import path  
from .views import user_profile  
  
urlpatterns = [  
    path('user_profile/', user_profile, name='user_profile'),  
]
```

6. Crie um template HTML para exibir o formulário e os dados do usuário. Por exemplo, em `templates/user_profile.html`:

```
<!DOCTYPE html>
<html>
<head>
  <title>User Profile</title>
</head>
<body>

<h2>User Profile</h2>

<form method="post" action="{% url 'user_profile' %}">
  {% csrf_token %}
  {{ form.as_p }}
  <button type="submit">Save</button>
</form>

<h3>Existing Profiles:</h3>
<ul>
  {% for profile in profiles %}
    <li>{{ profile.username }} - {{ profile.email }} - {{ profile.bio }}</li>
  {% endfor %}
</ul>

</body>
</html>
```

7. Adicione as URLs do seu aplicativo à configuração de URLs principal (`urls.py` do projeto):

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

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('yourapp.urls')),
]
```

8. Agora, ao acessar `http://localhost:8000/user_profile/`, você deve ver o formulário para adicionar usuários e a lista de perfis existentes.

Tutorial T5

Tutorial de como fazer upload de arquivos para a aplicação web (servidor). Criação de um formulário de upload de imagens e um formulário para listar os arquivos salvos na aplicação web.

Linguagem: Python

Framework: Django

Banco de dados: SQLite3

Passo a passo:

Passo 1: Dentro do projeto principal:

Deverá ser criado um aplicativo django:

```
PS C:\Users\guial\OneDrive\Área de Trabalho\ufpi\tutorial5> python manage.py startapp fileupload
```

Passo 2: Modelagem do banco de dados:

No arquivo fileupload/models.py, defina o modelo 'UploadedFile' para representar os arquivos enviados.

```
tutorial5 > fileupload > models.py > ...  
1  # Create your models here.  
2  from django.db import models  
3  
4  class UploadedFile(models.Model):  
5      file = models.FileField(upload_to='uploads/')  
6      uploaded_at = models.DateTimeField(auto_now_add=True)  
7  
8  
9
```

Passo 3: Configuração do formulário.

Crie um formulário em 'fileupload/forms.py' para lidar com o envio de arquivos, inclua as URLs do aplicativo em tutorial5/urls.py.

```
tutorial5 > fileupload > forms.py > FileUploadForm > Meta
1  from django import forms
2  from .models import UploadedFile
3
4  class FileUploadForm(forms.ModelForm):
5      class Meta:
6          model = UploadedFile
7          fields = ['file']
```

Passo 4: Configurando o ambiente de mídia.

No arquivo tutorial5/settings.py, configure o ambiente de mídia para lidar com arquivos durante o desenvolvimento.

```
129
130  MEDIA_URL = '/media/'
131  MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
```

Adicione também uma URL de mídia em tutorial5/urls.py.

```
28
29  urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Passo 5: Desenvolvimento de visualizações:

Crie templates em 'fileupload/templates/fileupload/' para renderizar as páginas HTML. Use 'upload_file.html' para a página de upload, 'file_list.html' para a lista de arquivos e 'upload_sucess.html' para a página de sucesso após o upload.

Arquivo.html 'upload_file.html'

```
tutorial5 > fileupload > templates > fileupload > <> upload_file.html > html > body > a
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>File Upload</title>
5  </head>
6  <body>
7
8  <h2>Upload File</h2>
9
10 <form method="post" enctype="multipart/form-data">
11 |   {% csrf_token %}
12 |   {{ form.as_p }}
13 |   <button type="submit">Upload</button>
14 </form>
15
16 <br>
17 <a href="{% url 'file_list' %}">Voltar para "Arquivos" list</a>
18
19 </body>
20 </html>
```

Arquivo.html 'upload_sucess.html'

```
tutorial5 > fileupload > templates > fileupload > <> upload_sucess.html > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Upload Success</title>
5  </head>
6  <body>
7
8  <h2>File Uploaded Successfully</h2>
9
10 <strong>{{ file.file.name }}</strong>
11 <br>
12 
13
14 <br>
15 <a href="{% url 'file_list' %}">Back to file list</a>
16
17 </body>
18 </html>
```

Passo 6: Estilo visual simples com CSS:

Adicione estilos css ao arquivo 'file_list.html' para melhorar a aparência da lista de arquivos (este passo não é estritamente obrigatório, porém o considere colocar-lo pois considero importante a familiaridade com CSS)

```
tutorial5 > fileupload > templates > fileupload > <> file_list.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>File List</title>
7      <style>
8          body {
9              font-family: Arial, sans-serif;
10             margin: 20px;
11             background-color: #f8f8f8;
12         }
13
14         h2 {
15             color: #333;
16         }
17
18         ul {
19             list-style: none;
20             padding: 0;
21             display: flex;
22             flex-wrap: wrap;
23         }
24
25         li {
26             margin: 10px;
27             background-color: #fff;
28             border: 1px solid #ddd;
29             border-radius: 5px;
30             overflow: hidden;
31             box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
32             transition: transform 0.3s ease;
33             cursor: pointer;
34         }
35
36         li:hover {
37             transform: translateY(-5px);
38         }
```

```
tutorial5 > fileupload > templates > fileupload > file_list.html > ...
39
40     img {
41         max-width: 100%;
42         height: auto;
43         display: block;
44         max-height: 200px; /* Defina o tamanho máximo desejado */
45     }
46
47     .upload-link {
48         display: block;
49         padding: 10px;
50         background-color: #4CAF50;
51         color: white;
52         text-decoration: none;
53         margin-top: 20px;
54         border-radius: 5px;
55     }
56 </style>
57 </head>
58 <body>
59
60 <h2>File List</h2>
61
62 {% if files %}
63     <ul>
64         {% for file in files %}
65             <li>
66                 
67                 <div>
68                     <strong>{{ file.file.name }}</strong>
69                 </div>
70             </li>
71         {% endfor %}
72     </ul>
73 {% else %}
74     <p>No files uploaded yet.</p>
75 {% endif %}
76
77 <a href="{% url 'upload_file' %}" class="upload-link">Upload a file</a>
78
79 </body>
80 </html>
81
```

Agora você deve aplicar o comando “python manage.py runserver” para inicializar o servidor da sua aplicação web e testá-lo.

