



INSTITUTO FEDERAL

Sul de Minas Gerais

Campus Muzambinho

Introdução ao framework

Django

+2024

Framework

page - 2

Um framework é uma estrutura pré-definida com ferramentas e regras que agiliza o desenvolvimento de software simplificando tarefas.



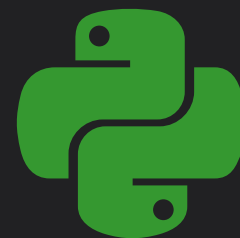
Linguagem Python

page - 3

Python é uma linguagem de programação extremamente versátil, de sintaxe simples e fácil aprendizado.



Atente-se com a indentação!



MVT

page - 4

Model (Modelo)

Define a estrutura do banco de dados (tabelas, campos e relacionamentos).

View (Visão)

Controla a lógica da aplicação e funciona como uma ponte entre o modelo e o template.

Template (Apresentação)

Lida com a apresentação e a exibição dos dados.



O que é Django?

page - 6

Django é um framework de desenvolvimento web em Python que segue o padrão MVT. Ele oferece uma estrutura robusta para facilitar a criação de aplicações web. É utilizado por empresas como Instagram, Pinterest e Mozilla.



→ **request**
(pedido)

→ **response**
(resposta)



Começar seu primeiro projeto

page - 7



Crie uma conta no GitHub

Você precisará de uma conta no GitHub para organizar seus repositórios e posteriormente para trabalharmos versionamento de código com Git.



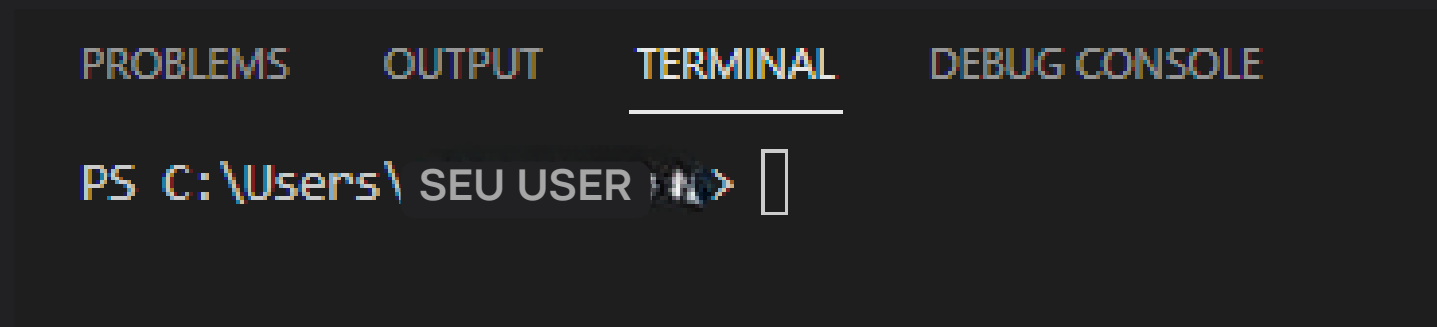
Abra o Visual Studio Code

Abra o editor Visual Studio Code, trabalharemos por meio dele para criar nosso primeiro projeto com Django.

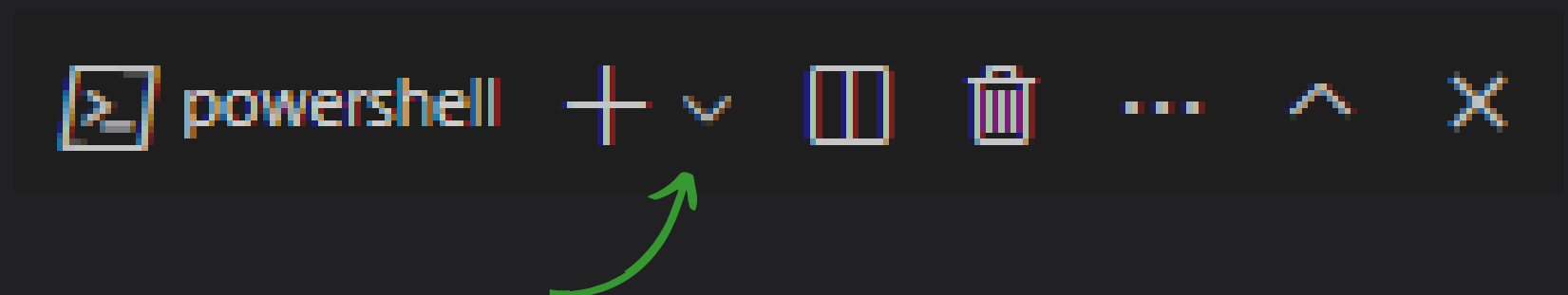


Acesse o terminal do VSCode

1 - Aperte as teclas
"Ctrl+Shift+Aspas" ou
siga o caminho:



2 - No canto inferior direito
você encontrará:



Seta para baixo, selecione:

Command Prompt

Para iniciar seu projeto, siga a sequência ATENTAMENTE:

```
C:\Users\17 SEU USER 14\projetos\projetoloja>py -m venv .venv
```

```
C:\Users\17 SEU USER 44\projetos\projetoloja>cd .venv/scripts
```

```
C:\Users\1 SEU USER 44\projetos\projetoloja\.venv\Scripts>activate
```

```
(.venv) C:\Users\1 SEU USER 14\projetos\projetoloja\.venv\Scripts>cd ../../
```

```
(.venv) C:\Users\1 SEU USER 44\projetos\projetoloja>pip install django
```

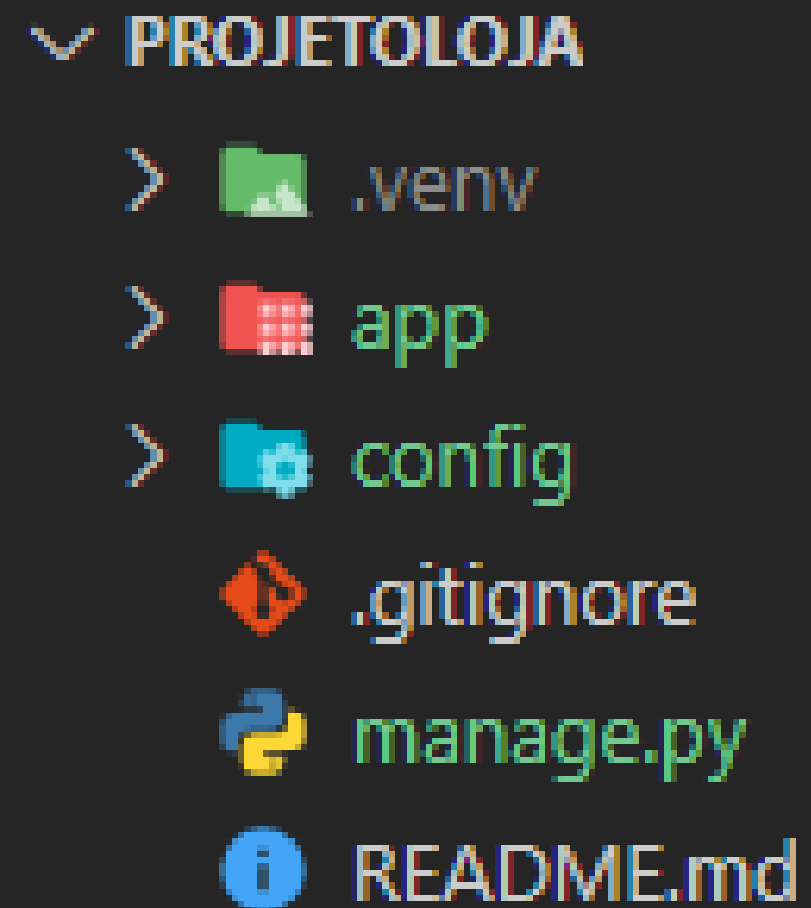
```
(.venv) C:\Users\ SEU USER 14\loja>pip install django psycopg2
```

```
(.venv) C:\Users\1 SEU USER 9\aplicacoes>pip install Pillow
```

```
(.venv) C:\Users\1 SEU USER 1\projetos\projetoloja>py -m django startproject config .
```

```
(.venv) C:\Users\1 SEU USER 1\projetos\projetoloja>py -m django startapp app
```

Ao final, sua estrutura de pastas deve estar semelhante:



A screenshot of a file explorer window showing the directory structure of a project named 'PROJETOLOJA'. The project is expanded, revealing several subdirectories and files. The subdirectories are '.venv', 'app', and 'config', each preceded by a right-pointing arrow. The files are '.gitignore', 'manage.py', and 'README.md', each preceded by its respective icon (a flame for .gitignore, a Python logo for manage.py, and an information icon for README.md). To the right of the file names, there are status indicators: a green dot for 'app' and 'config', and a green 'U' for 'manage.py'.

- ▼ PROJETOLOJA
 - > .venv
 - > app
 - > config
 - .gitignore
 - manage.py
 - README.md

Siga o caminho '`config/settings.py`' e configure seu projeto conforme apresentado:

```
import os # Import the os module
```

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
    'app', # Add the app to the list of installed apps  
]
```

//////////

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

Mude para



```
DATABASES = {  
    'default': {  
        'ENGINE': 'django.db.backends.postgresql',  
        'NAME': 'loja',  
        'USER': 'postgres',  
        'PASSWORD': '123456',  
        'HOST': 'localhost',  
        'PORT': '5432',  
    }  
}
```

```
TEMPLATES = [  
    {  
        'BACKEND': 'django.template.backends.django.DjangoTemplates',  
        'DIRS': [os.path.join(BASE_DIR, 'app/templates')],
```

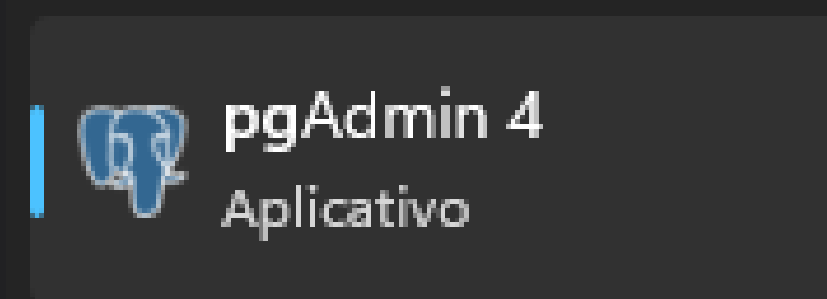
Adicione isso a linha 'DIRS'

```
107     LANGUAGE_CODE = 'pt-br' # Change the language code to pt-br  
108  
109     TIME_ZONE = 'America/Sao_Paulo' # Change the time zone to America/Sao_Paulo
```

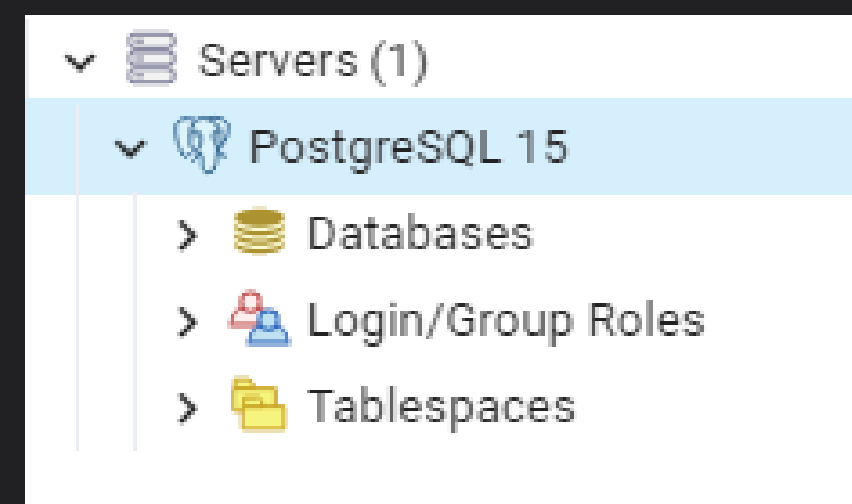
```
# URL to use when referring to static files located in STATIC_ROOT.  
STATIC_URL = 'static/'  
# Root for collecting static files.  
STATIC_ROOT = os.path.join(STATIC_URL, 'static')  
# Directories where static files will be stored  
STATICFILES_DIRS = [  
    os.path.join(BASE_DIR, 'app/static')  
]
```

//////////

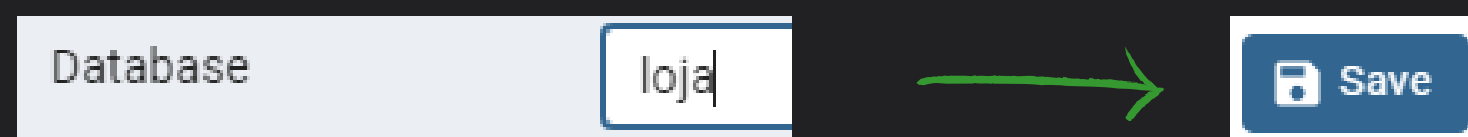
Tecle Windows, pesquise por “pgAdmin” e abra o aplicativo.



Digite a senha “123456” e tecle Enter para acessar.
Clique em “Servers” e digite novamente a senha “123456”.

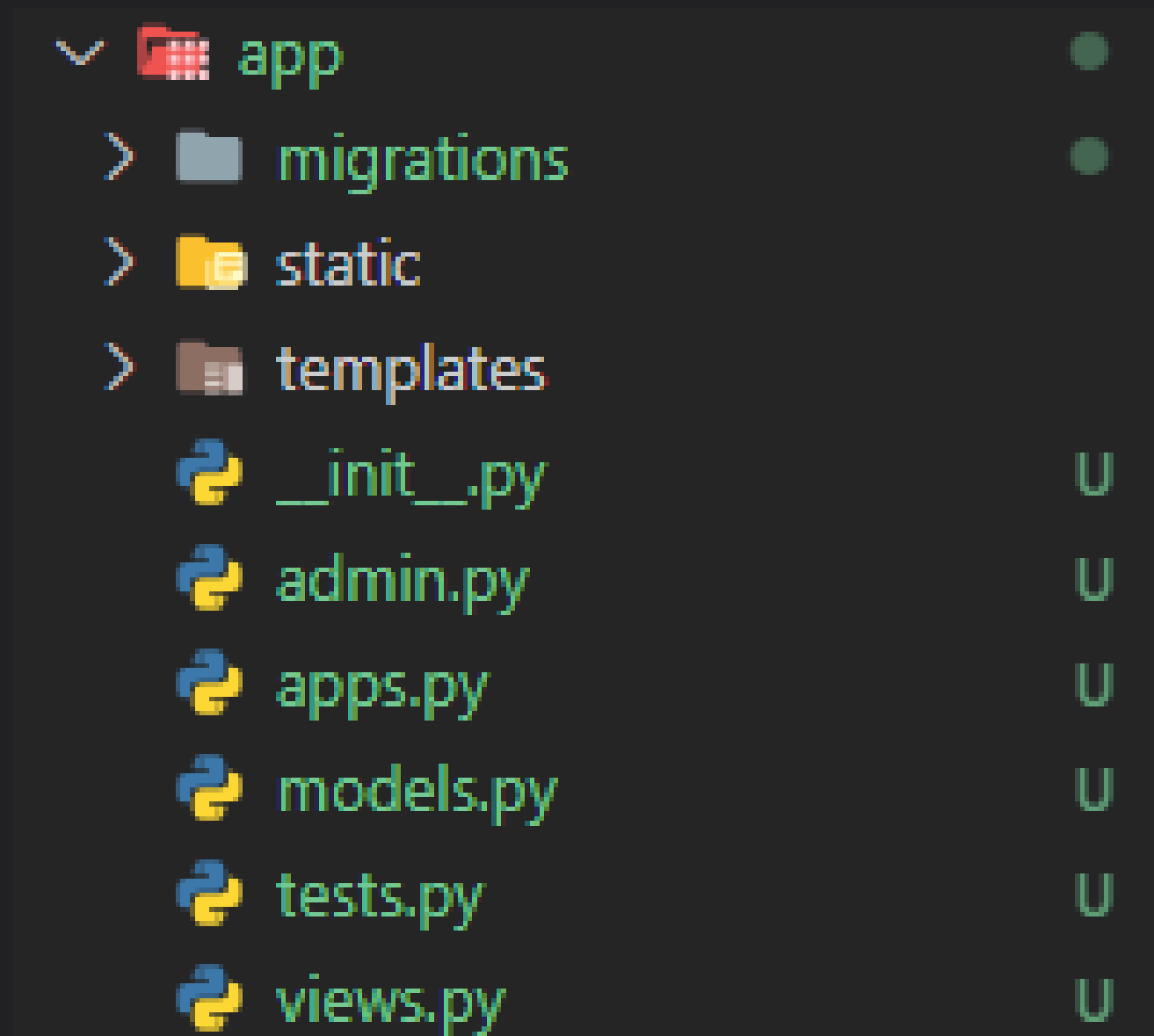


Prossiga clicando com o botão direito do mouse em “Databases”, siga “Create > Database...”. Digite “loja” no primeiro campo e prossiga com o botão “Save”.



//////////

Dentro da pasta '`app`', crie as pastas '`templates`' e '`static`', vamos trabalhar com elas depois.:



//////////

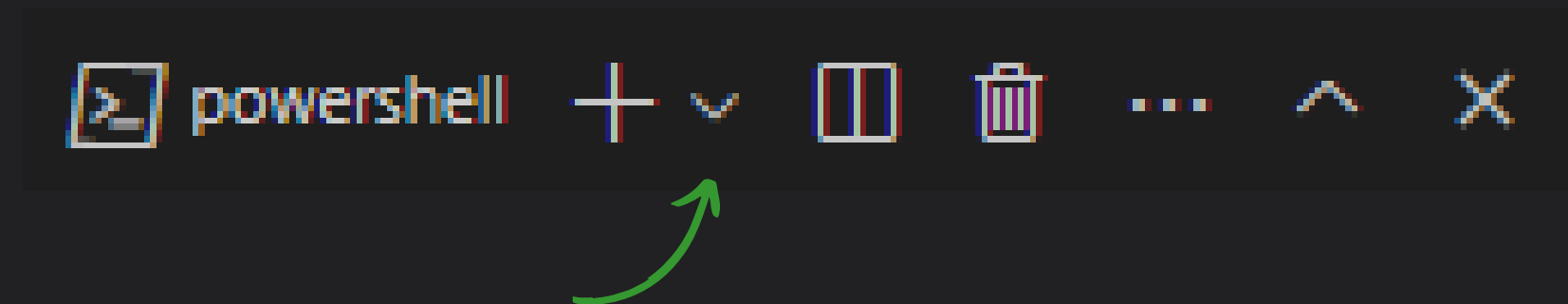
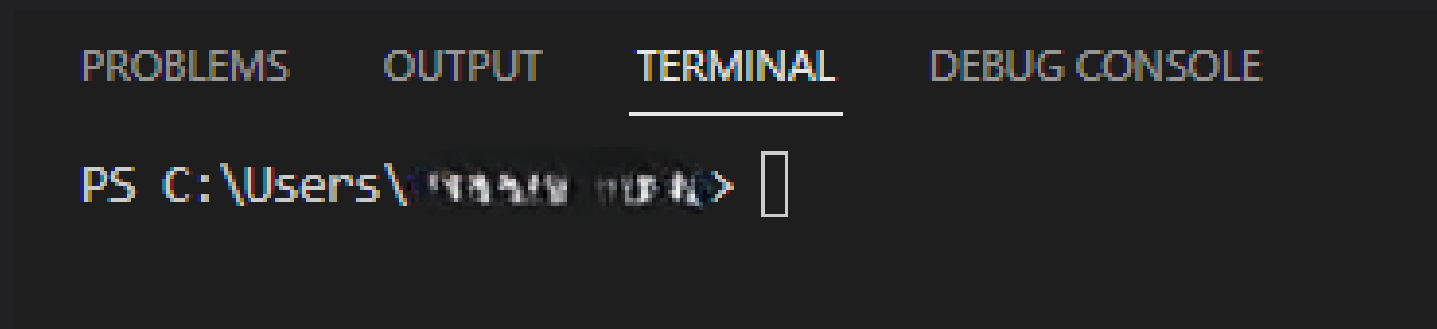
Siga o caminho '`app/models.py`'.

Vamos criar nossos models:

```
3  class Categoria(models.Model):
4      nome = models.CharField(max_length=100)
5
6      class Meta:
7          verbose_name_plural = "Categorias"
8
9      def __str__(self):
10         return self.nome
```

//////////

De volta ao terminal, vamos criar nossas tabela



selecione:

Command Prompt

Com o prompt de comando aberto, digite e execute:

py manage.py makemigrations

E depois:

py manage.py migrate

//////////

De volta ao terminal, vamos criar um administrador.

Com o prompt de comando aberto, digite e execute:

```
py manage.py createsuperuser
```

Insira as credenciais:

Usuário: admin

Endereço de email: (não é obrigatório, apenas tecle Enter)

Password: 123 (a senha não aparece, é normal)

Password (again): 123 (novamente)

Confirme com y e Enter.

```
Superuser created successfully.
```

Se isso apareceu, é porque foi criado com sucesso!



O CAMPO SENHA NÃO APARECE,
SÃO CARACTERES INVISÍVEIS
ENTÃO CUIDADO AO INSERIR A SENHA

//////////

Suba o servidor local de desenvolvimento:

```
py manage.py runserver
```

```
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
September 04, 2023 - 08:12:41
```

```
Django version 4.2.4, using settings 'config.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CTRL-BREAK.
```

Dê Ctrl + Clique em cima do link marcado de vermelho.



Siga o caminho '`app/admin.py`'. Vamos registrar nossos models no Django Admin:

```
3  # Importing all models from app/models.py
4  from .models import *
```

```
admin.site.register(Categoria)
```

//////////

Após abrir em seu navegador padrão, entre com suas credenciais criadas com o super user:





Administração do Django

Usuário:

Senha:

Acessar

Por fim aparecerá nossas tabelas criadas:

APP		
Categorias	+ Adicionar	 Modificar
Produtos	+ Adicionar	 Modificar
AUTENTICAÇÃO E AUTORIZAÇÃO		
Grupos	+ Adicionar	 Modificar
Usuários	+ Adicionar	 Modificar

Clicando em Categoria, abrirá esse forms, vamos adicionar 3 categorias

Adicionar categoria

Nome:

SALVAR

Salvar e adicionar outro(a)

Salvar e continuar editando

Siga o caminho `'app/models.py'`. Vamos criar nossa model de produto:

```
class Produto(models.Model):
    nome = models.CharField(max_length=100)
    fabricante = models.CharField(max_length=100)
    imagem = models.ImageField(upload_to='static/', blank=True)
    preco = models.DecimalField(max_digits=10, decimal_places=2)
    categoria = models.ForeignKey(Categoria, on_delete=models.CASCADE)

    class Meta:
        verbose_name_plural = "Produtos"

    def __str__(self):
        return f'{self.categoria} {self.nome}'
```

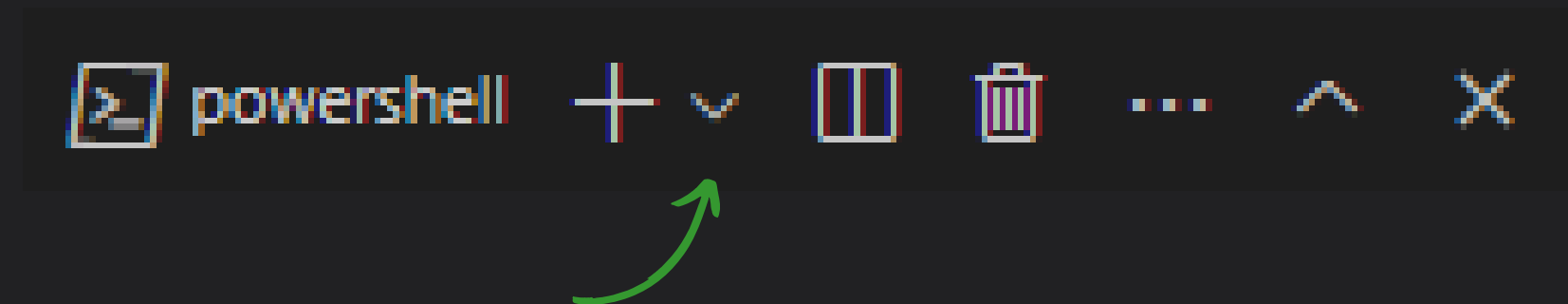
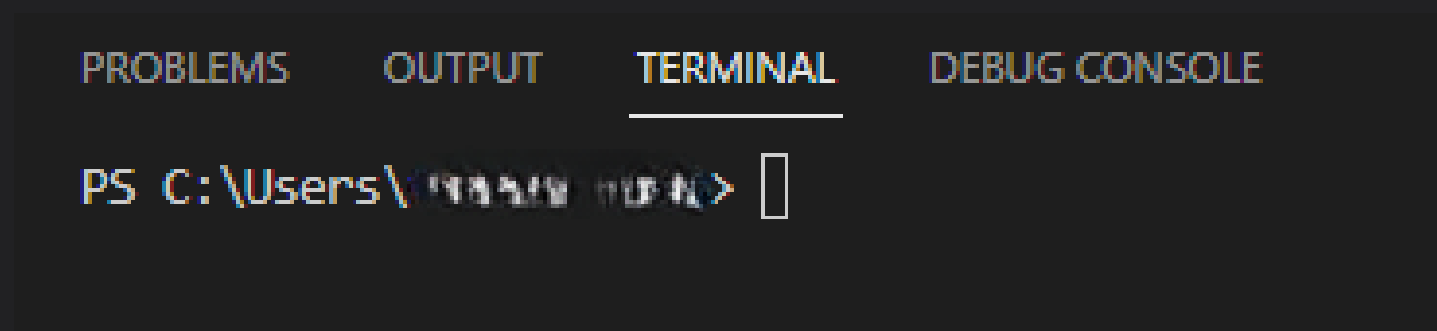
//////////

Siga o caminho '`app/admin.py`'. Vamos registrar a model no Django Admin:

```
3     # Importing all models from app/models.py
4     from .models import *
```

```
admin.site.register(Produto)
```

De volta ao terminal, vamos criar nossas tabela



selezione:

Command Prompt

Com o prompt de comando aberto, digite e execute:

py manage.py makemigrations

E depois:

py manage.py migrate

//////////



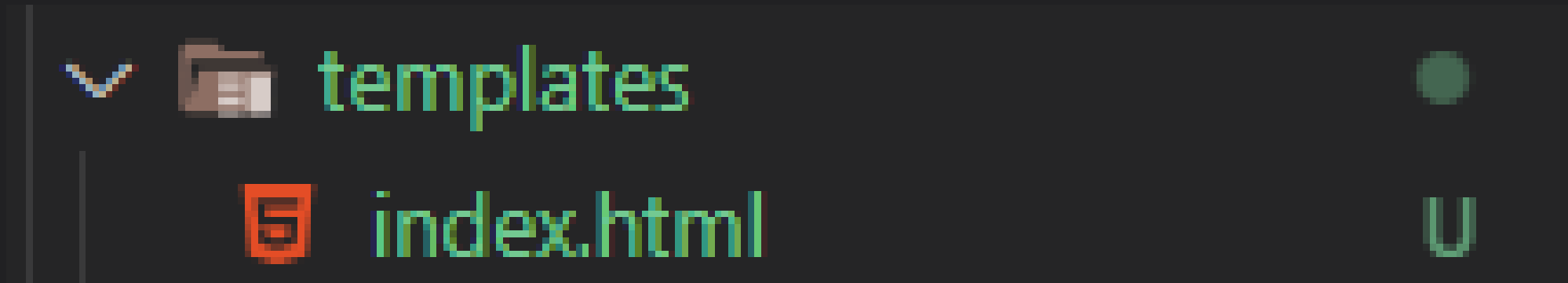
Front-end web

Siga o caminho `'app/views.py'`. Vamos criar a lógica do nosso aplicativo:

```
3  # Importing View class from django.views
4  from django.views import View
5
6  # Importing all models from app/models.py
7  from .models import *
8
9
10
11
12
13  # Index View
14  class IndexView(View):
15      # Defining get method
16      def get(self, request):
17          return render(request, 'index.html',)
18      # Defining post method
19      def post(self, request):
20          pass
```

//////////

Na pasta 'templates' crie o seu
"index.html":



```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
7     <title>Loja</title>
8 </head>
9 <body>
10
11 <div class="container mt-5">
12 <button class="btn btn-warning mt-5" onclick="window.location.href='/admin/'">Admin</button>
13 <h1 class="text-center">Produtos</h1>
14 <table class="table table-info">
15     <thead>
16         <tr>
17             <th>ID</th>
18             <th>Nome</th>
19             <th>Fabricante</th>
20             <th>Preço</th>
21             <th>Categoria</th>
22         </tr>
23     </thead>
```



```
<tbody>
  {% for produto in produtos %}
    <tr>
      <td><strong>{{ produto.id }}</strong></td>
      <td>{{ produto.nome }}</td>
      <td>{{ produto.fabricante }}</td>
      <td>R$ {{ produto.preco|floatformat:2 }}</td>
      <td>{{ produto.categoria }}</td>
    </tr>
  {% endfor %}
</tbody>
</table>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Siga o caminho 'config/urls.py'.
Vamos :

```
1 from django.urls import path
2 from django.contrib import admin
3 from .views import *
4
5 urlpatterns = [
6     path('admin/', admin.site.urls),
7     path("", IndexView.as_view(), name="index"),
8 ]
```

CaioTomazz, 5 months ago • feat: integração

//////////

Agora se não estiver ativado, dê novamente o comando `py manage.py runserver`, clique no link e veja se sua página html foi ativada!

Django Admin				
Produtos				
ID	Nome	Fabricante	Preço	Categoria

//////////

Vamos para a [Página de Administração](#) e vamos adicionar uns produtos:

Add produto

Nome:

Fabricante:

Imagem:

Escolher arquivo

Nenhum arquivo escolhido

Preço:

Categoria:

 ▼

ID

NOME

FABRICANTE

PREÇO

CATEGORIA



2

Fini Ovo Frito

Fini

7.50

Bala Mastigavel



1

Lolo

Nestle

2.50

Chocolate

Por fim aparecerá todos os produtos criados em uma tabela:

page - 34

Django Admin

Produtos

ID	Nome	Fabricante	Preço	Categoria
8	Farofa Temperada	Yoki	R\$ 3,50	Farofa
9	Pop Mania Frutas Tropicais	Pop Mania	R\$ 1,20	Doces
10	Cola Lata	Coca Cola	R\$ 3,50	Refrigerante



Em construção...