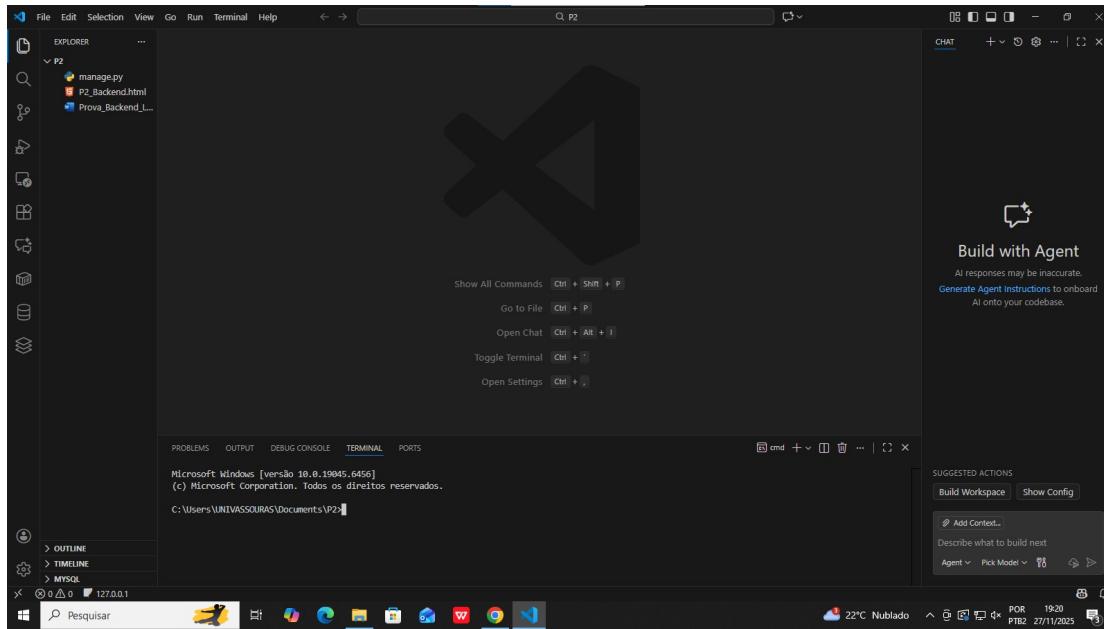


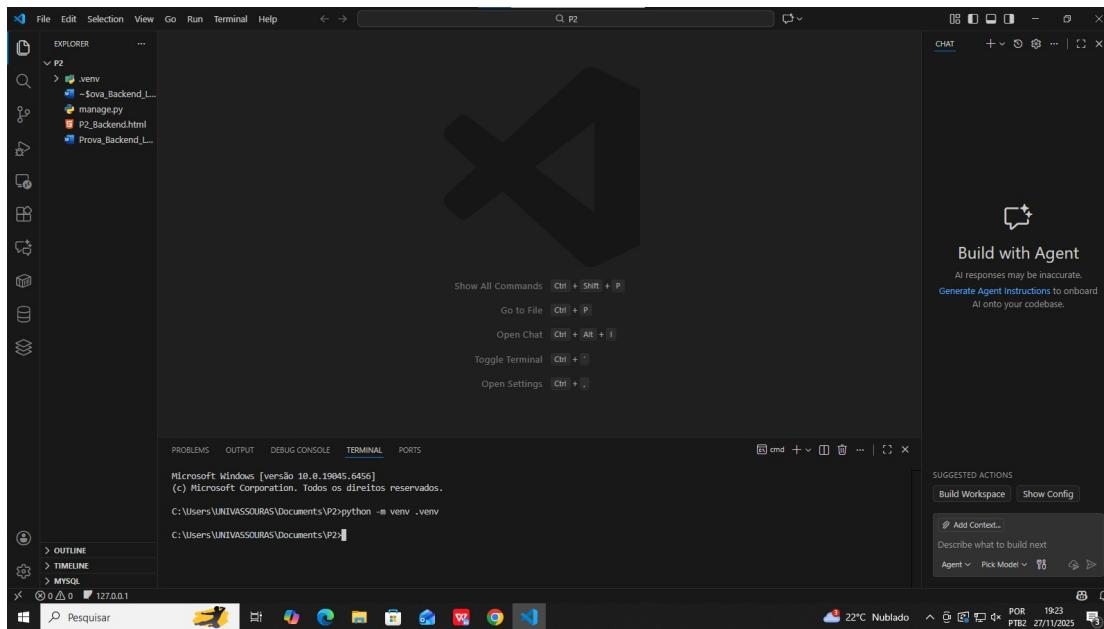
UNIVASSOURAS

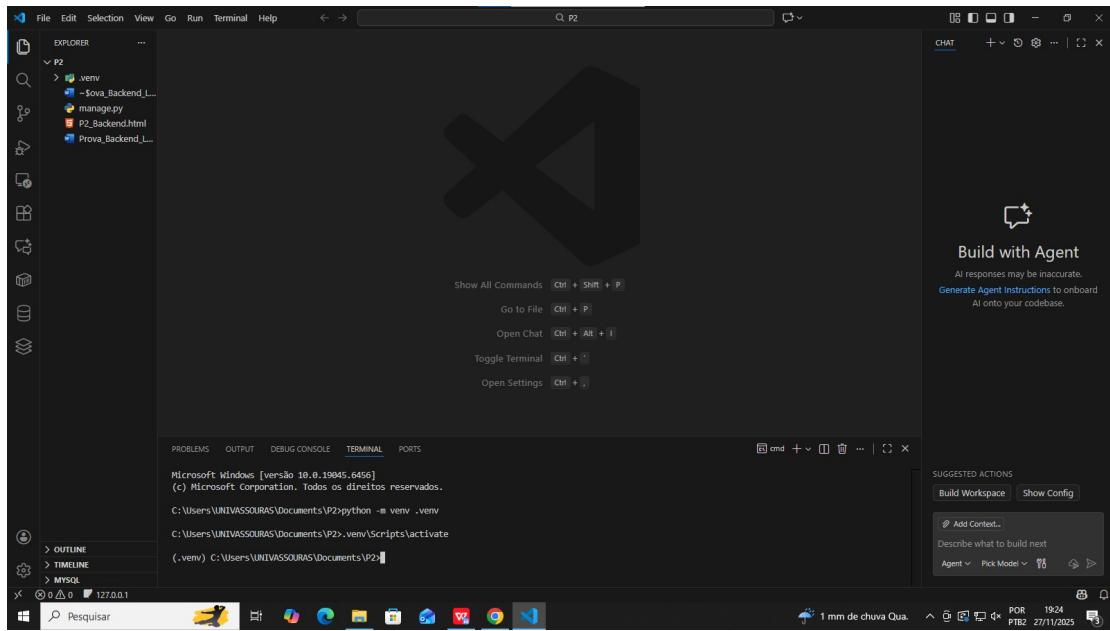
LEANDRO LIMA CARDOSO - 202323366

PASSO 01: Iniciando o ambiente.

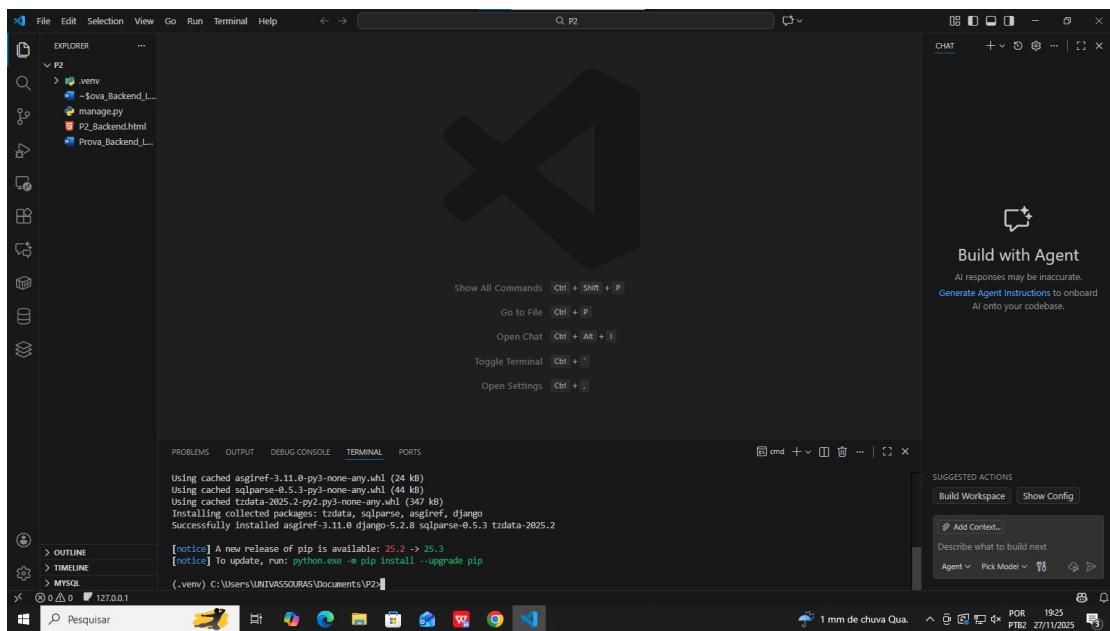


PASSO 02: Iniciando o ambiente virtual (venv).

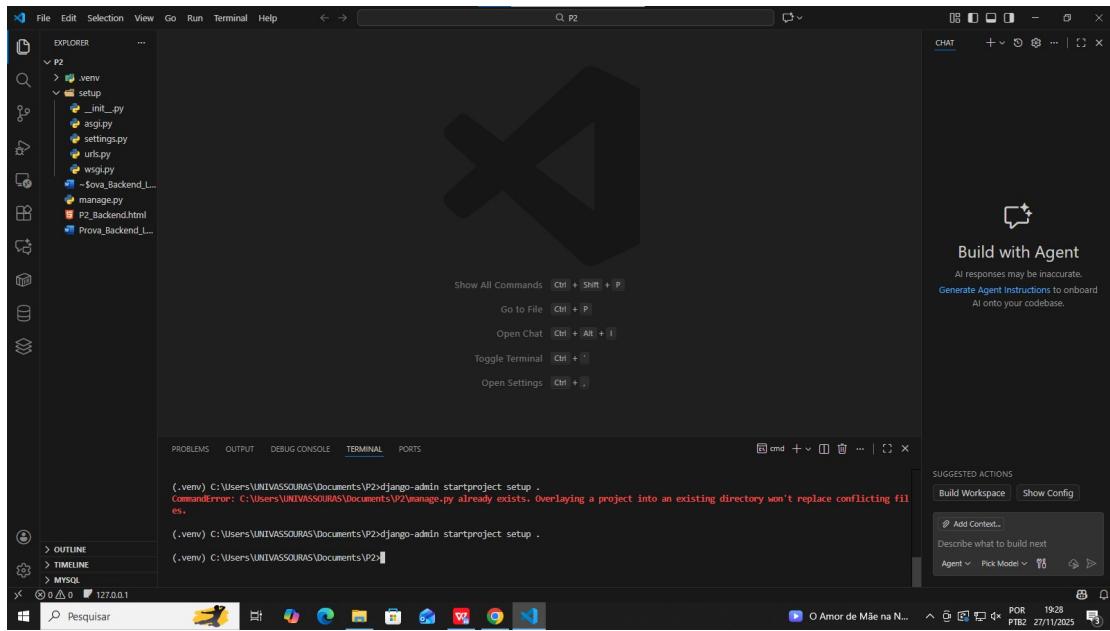




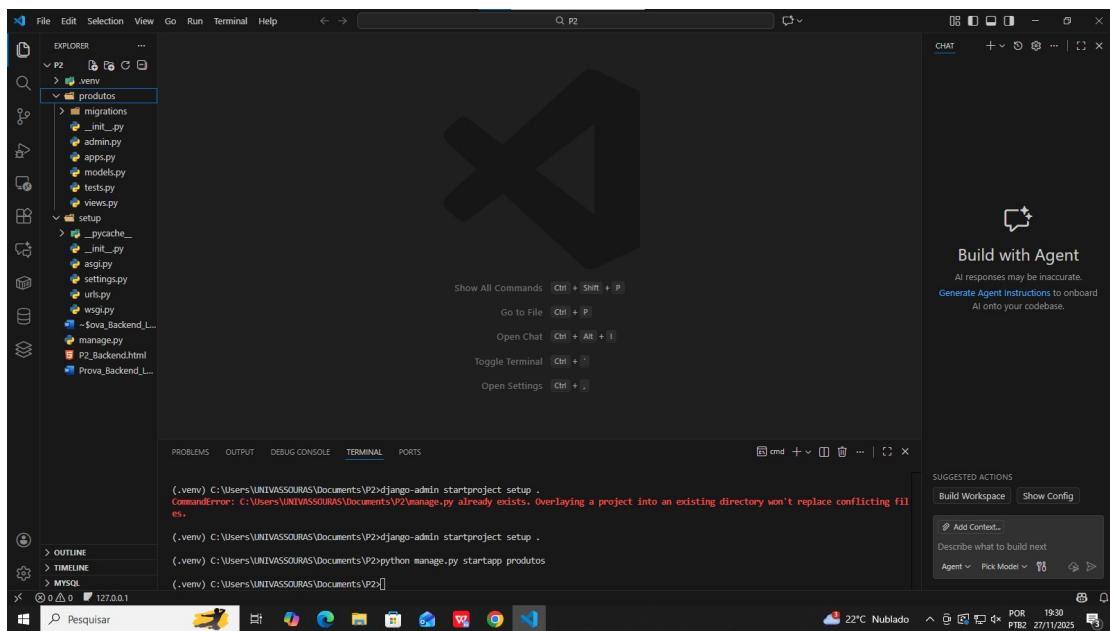
PASSO 03: Instalando Django.



PASSO 03: Iniciando o projeto Django.



PASSO 04: Criando aplicação Django.



PASSO 05: Configurando o settings do setup.

The screenshot shows a Windows desktop environment with a Visual Studio Code (VS Code) window open. The title bar of the VS Code window reads "File Edi Selection View Go Run Terminal Help". The left sidebar (Explorer) shows a file tree for a Django project named "P2". The "settings.py" file is open in the main editor area, displaying configuration code for installed apps, middleware, and templates. The bottom status bar shows command-line output related to project setup. A floating "Build with Agent" card is visible on the right, and the taskbar at the bottom includes icons for File Explorer, Task View, Start, and other system tools.

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

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    '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',
]

ROOT_URLCONF = 'setup.urls'

TEMPLATES = [
```

PASSO 06: Criando o modelo de Produto.

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with a Django project open. The left sidebar displays the file structure of the 'produtos' app, including files like `migrations`, `admin.py`, `apps.py`, `models.py`, `tests.py`, `views.py`, and `__init__.py`. The `models.py` file is currently selected and shown in the main editor area:

```
from django.db import models

class Produto(models.Model):
    name = models.CharField(max_length = 100)
    category = models.CharField(max_length = 100)
    price = models.DecimalField(max_digits=8, decimal_places=2)

    def __str__(self):
        return f'{self.name} ({self.category}) - R$ {self.price}'
```

A red error message is visible in the terminal tab, indicating that `migrations.py` already exists and cannot be replaced.

The right side of the interface features a "Build with Agent" button and a "SUGGESTED ACTIONS" bar with options like "Build Workspace" and "Show Config". The bottom status bar shows the current file is `P02`, has 1 mm de chuva Qua, and includes navigation icons for back, forward, and search.

PASSO 07: Fazendo a migração do modelo de Produto.

```

models.py
produtos > models.py > ...
1  from django.db import models
2
3  class Produto(models.Model):
4      name = models.CharField(max_length = 100)
5      category = models.CharField(max_length = 100)
6      price = models.DecimalField(max_digits=8, decimal_places=2)
7
8      def __str__(self):
9          return f'{self.name} ({self.category}) - R$ {self.price}'
10

```

The screenshot shows the VS Code interface with the 'models.py' file open in the editor. The code defines a 'Produto' model with three fields: 'name' (CharField with max length 100), 'category' (CharField with max length 100), and 'price' (DecimalField with max digits 8 and 2 decimal places). A __str__ method is also defined to return the product name and category followed by its price.

PASSO 08: Criando a view para listar produtos.

```

views.py
produtos > views.py > ...
1  from django.shortcuts import render
2  from django.views.generic import ListView
3  from django.urls import reverse_lazy
4  from .models import Produto
5
6  class ProdutoListView(ListView):
7      model = Produto
8      template_name = "produtos/produto_list.html"
9      context_object_name = "produtos"
10

```

The screenshot shows the VS Code interface with the 'views.py' file open in the editor. The code defines a 'ProdutoListView' class that inherits from Django's 'ListView'. It specifies the model as 'Produto', the template name as 'produtos/produto_list.html', and the context object name as 'produtos'.

PASSO 09: Criando a url de setup.

The screenshot shows the Visual Studio Code interface with the Python extension installed. The Explorer sidebar on the left lists files and folders for a Django project named 'P2'. The 'urls.py' file in the 'produtos' directory is open in the editor. The code defines a URL pattern for listing products:

```
8     1. Add an import: from my_app import views
9
10    2. Add a URL to urlpatterns: path('', views.home, name='home')
11
12 Class-based views
13     1. Add an import: from other_app.views import Home
14
15     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
16
17 Including another URLconf
18     1. Import the include() function: from django.urls import include, path
19
20     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
21
22 """
23
24 urlpatterns = [
25     path('admin/', admin.site.urls),
26     path('', include('produtos.urls'))
27 ]
```

The terminal at the bottom shows the command 'python manage.py makemigrations' being run, with the output indicating successful migrations for auth, sessions, and produtos models.

PASSO 09: Criando a url para listar produtos.

The screenshot shows the Visual Studio Code interface with the Python extension installed. The Explorer sidebar on the left lists files and folders for a Django project named 'P2'. The 'urls.py' file in the 'produtos' directory is open in the editor. The code defines a URL pattern for listing products:

```
1  from django.urls import path
2  from produtos.views import ProdutoListView
3
4 urlpatterns = [
5     path("", ProdutoListView.as_view(), name="produto_list")
6 ]
```

The terminal at the bottom shows the command 'python manage.py makemigrations' being run, with the output indicating successful migrations for auth, sessions, and produtos models.

PASSO 10: Criando o template de listar produtos.

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files and folders for a Django project named 'P2'. The code editor window displays a template file named 'produto_list.html' with the following content:

```

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   <title>Produtos</title>
7 </head>
8 <body>
9   <h1>Produtos</h1>
10  <ul>
11    {% for produto in produtos %}
12      <li>
13        {{ produto.name }} ({{ produto.category }}) - R$ {{ produto.price }}
14        <br>
15      </li>
16    {% empty %}
17    <li>Nenhum produto cadastrado.</li>
18  {% endfor %}
19 </ul>
20 </body>
21 </html>

```

The terminal at the bottom shows the output of migrations:

```

Applying auth.0010_alter_group_name_max_length... OK
Applying auth.0011_update_proxy_permissions... OK
Applying auth.0012_alter_user_first_name_max_length... OK
Applying products.0001_initial... OK
Applying sessions.0001_initial... OK

```

PASSO 11: Testando a tela pagina de listagem de produtos.

The screenshot shows the Visual Studio Code interface and a web browser window. The browser window displays a Django development server at '127.0.0.1:8000'. The page title is 'Produtos' and the content is:

Produtos

- Nenhum produto cadastrado.

The terminal in VS Code shows the server starting and a log entry:

```

November 27, 2025 - 19:58:01
Django version 5.2.8, using settings 'setup.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or
for more information on how to run production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[27/Nov/2025 19:58:00] "GET /favicon.ico HTTP/1.1" 404 2461
[27/Nov/2025 19:58:00] "GET /favicon.ico HTTP/1.1" 404 2461

```

PASSO 12: Criando a view para adicionar produtos.

```
File Edit Selection View Go Run Terminal Help ⏎ ↻ P2
EXPLORER ... model.py views.py urls.py produto_list.html
produtos > views.py ...
1 from django.shortcuts import render
2 from django.views.generic import ListView, CreateView
3 from django.urls import reverse_lazy
4 from .models import Produto
5
6 class ProdutoListView(ListView):
7     model = Produto
8     template_name = "produtos/produto_list.html"
9     context_object_name = "produtos"
10
11 class ProdutoCreateView(CreateView):
12     model = Produto
13     fields = ["name", "category", "price"]
14     success_url = reverse_lazy("produto_list")
15

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard
AI onto your codebase.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
November 27, 2025 - 19:58:01
Django version 5.2.8, using settings 'setup.settings'.
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[27/Nov/2025 19:58:00] "GET / HTTP/1.1" 200 308
Not Found: /favicon.ico
[27/Nov/2025 19:58:00] "GET /favicon.ico HTTP/1.1" 404 2461
C:\Users\WILWSS\URB\Documents\P2\produtos\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
Ln 15, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.6 (venv) SUGGESTED ACTIONS
Build Workspace Show Config
Agent Pick Model > > >
Ln & Col 1 Spaces:4 UTF-8 CRLF Python 3.13.6 (venv) >
```

PASSO 13: Criando a url para adicionar produtos.

```
File Edit Selection View Go Run Terminal Help ⏎ ↻ P2
EXPLORER ... model.py views.py urls.py produto_list.html
produtos > urls.py ...
1 from django.urls import path
2 from produtos.views import ProdutoListView, ProdutoCreateView
3
4 urlpatterns = [
5     path("", ProdutoListView.as_view(), name="produto_list"),
6     path("create/", ProdutoCreateView.as_view(), name="produto_create")
7 ]
8

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard
AI onto your codebase.

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
November 27, 2025 - 19:58:01
Django version 5.2.8, using settings 'setup.settings'.
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
[27/Nov/2025 19:58:00] "GET / HTTP/1.1" 200 308
Not Found: /favicon.ico
[27/Nov/2025 19:58:00] "GET /favicon.ico HTTP/1.1" 404 2461
C:\Users\WILWSS\URB\Documents\P2\produtos\views.py changed, reloading.
Watching for file changes with StatReloader
Performing system checks...
Ln 15, Col 1 Spaces:4 UTF-8 CRLF Python 3.13.6 (venv) SUGGESTED ACTIONS
Build Workspace Show Config
Agent Pick Model > > >
Ln & Col 1 Spaces:4 UTF-8 CRLF Python 3.13.6 (venv) >
```

PASSO 13: Criando o template do formulário para adicionar produtos.

```

1 <h1>Produto</h1>
2
3
4 <form method="post">
5   {% csrf_token %}
6   {{ form.as_p }}
7
8   <button type="submit">Salvar</button>
9 </form>
10
11

```

The screenshot shows the Visual Studio Code interface with the product_list.html file open in the editor. The code displays a simple HTML structure with a form for saving products. A message at the bottom of the code indicates it's a development server and to use a production WSGI or ASGI server instead.

PASSO 13: Adicionando o botão no template de listagem de produtos.

```

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     <title>Produtos</title>
7   </head>
8   <body>
9
10  <h1>Produtos</h1>
11
12  <ul>
13    {% for produto in produtos %}
14      <li>
15        {{ produto.name }} ({{ produto.category }}) - R$ {{ produto.price }}
16        <br>
17        {% empty %}
18        <li>Nenhum produto cadastrado.</li>
19        {% endfor %}
20    </ul>
21
22  <a href="{% url 'produto_create' %}">Adicionar Produto</a>
23
24 </body>
25 </html>

```

This screenshot shows the same VS Code environment after adding a new line to the template. The new line, Adicionar Produto, creates a link for adding new products. The terminal output at the bottom shows the server starting again.

PASSO 14: Testando a adição de um novo produto.

The screenshot shows a Windows desktop environment. On the left, a Visual Studio Code window displays the code for a Django application's product list template. The code includes HTML, CSS, and Python logic for displaying a list of products and a link to add a new one. On the right, a web browser window shows the resulting page titled 'Produtos'. It lists a single item: 'Nenhum produto cadastrado.' Below this is a link to 'Adicionar Produto'. The taskbar at the bottom shows various pinned icons and the system clock.

This screenshot is similar to the one above, but the browser window now displays a product creation form. The URL in the address bar is '127.0.0.1:8000/create/'. The form has fields for 'Name', 'Category', and 'Price', with a 'Salvar' (Save) button. A 'Google Translate' dropdown is visible above the form, showing options for 'inglês' and 'português'. The taskbar at the bottom remains the same.

The screenshot shows a Windows desktop with a code editor and a browser. The code editor displays a Python file (models.py) and an HTML template (produtos/list.html). The HTML template contains a list of products and a link to add a new one. The browser window shows a list of products: 'Computador (Informatica) - R\$ 5000,00'. Below the list is a button labeled 'Adicionar Produto'.

This screenshot is identical to the one above, showing the same code editor and browser interface. The code editor has the same Python and HTML files open, and the browser shows the same list of products and 'Adicionar Produto' button.

PASSO 15: Criando a view para a edição de produto.

```

File Edit Selection View Go Run Terminal Help ⌘ P2
EXPLORER ... model.py view.py urls.py produto_list.html produto_form.html
produtos > views.py ...
1 from django.shortcuts import render
2 from django.views.generic import ListView, CreateView, UpdateView
3 from django.urls import reverse_lazy
4 from .models import Produto
5
6 class ProdutoListView(ListView):
7     model = Produto
8     template_name = "produtos/produto_list.html"
9     context_object_name = "produtos"
10
11 class ProdutoCreateView(CreateView):
12     model = Produto
13     fields = ["name", "category", "price"]
14     success_url = reverse_lazy("produto_list")
15
16 class ProdutoUpdateView(UpdateView):
17     model = Produto
18     fields = ["name", "category", "price"]
19     success_url = reverse_lazy("produto_list")
20

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[27 Nov/2025 20:11:38] "POST /create/ HTTP/1.1" 302 0
[27 Nov/2025 20:11:38] "GET / HTTP/1.1" 200 596
[27 Nov/2025 20:11:38] "GET /create/ HTTP/1.1" 200 662
[27 Nov/2025 20:12:06] "POST /create/ HTTP/1.1" 302 0
[27 Nov/2025 20:12:06] "GET / HTTP/1.1" 200 695
(.venv) C:\Users\UNIVASOURIS\Documents\P2]

SUGGESTED ACTIONS Build Workspace Show Config

Pesquisar

PASSO 16: Criando a url para a edição de produto.

```

File Edit Selection View Go Run Terminal Help ⌘ P2
EXPLORER ... model.py view.py urls.py produto_list.html produto_form.html
produtos > urls.py ...
1 from django.urls import path
2 from produtos.views import ProdutoListView, ProdutoCreateView, ProdutoUpdateView
3
4 urlpatterns = [
5     path("", ProdutoListView.as_view(), name="produto_list"),
6     path("create/", ProdutoCreateView.as_view(), name="produto_create"),
7     path("edit/<int:pk>/", ProdutoUpdateView.as_view(), name="produto_update")
8 ]
9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[27 Nov/2025 20:11:38] "POST /create/ HTTP/1.1" 302 0
[27 Nov/2025 20:11:38] "GET / HTTP/1.1" 200 596
[27 Nov/2025 20:11:44] "GET /create/ HTTP/1.1" 200 662
[27 Nov/2025 20:12:06] "POST /create/ HTTP/1.1" 302 0
[27 Nov/2025 20:12:06] "GET / HTTP/1.1" 200 695
(.venv) C:\Users\UNIVASOURIS\Documents\P2]

SUGGESTED ACTIONS Build Workspace Show Config

Pesquisar

PASSO 17: Adicionando o botão no template de listagem de produtos.

PASSO 18: Testando a edição de um produto.

VS Code Screenshot:

- Code Editor:** Shows the `produtos/list.html` file with Django template code. The code lists products and includes a link to edit each product.
- Terminal:** Shows logs of requests to the development server at port 8000.
- Browser:** Shows a list of products (Computador, Celular, Tablet) with an "Editar" link next to each.

PASSO 18: Testando a edição de um produto.

VS Code Screenshot:

- Code Editor:** Shows the `produtos/list.html` file with Django template code. The code lists products and includes a link to edit each product.
- Terminal:** Shows logs of requests to the development server at port 8000.
- Browser:** Shows a list of products (Computador, Celular, Tablet) with an "Editar" link next to each.

The screenshot shows a Windows desktop environment. On the left, there is a code editor window for a Django project. The file being edited is `produtos/templates/produtos/list.html`. The code displays a list of products in English. On the right, there are two browser windows. The top browser window shows a list of products in English. The bottom browser window shows a product detail page with fields for Name (Celular), Category (Comunicacoes), and Price (3000.00), with a 'Salvar' button.

The screenshot shows a Windows desktop environment. On the left, there is a code editor window for a Django project. The file being edited is `produtos/templates/produtos/list.html`. The code displays a list of products in English. On the right, there are two browser windows. The top browser window shows a list of products in English. The bottom browser window shows a list of products in Portuguese, including a computer (Computador - R\$ 5000,00), a modified mobile phone (Celular MODIFICADO - R\$ 2850,00), and a tablet (Tablet - R\$ 2500,00).

PASSO 19: Criando a view para apagar um produto.

```
from django.shortcuts import render
from django.views.generic import ListView, CreateView, UpdateView, DeleteView
from django.urls import reverse_lazy
from .models import Produto

class ProdutoListView(ListView):
    model = Produto
    template_name = "produtos/produto_list.html"
    context_object_name = "produtos"

class ProdutoCreateView(CreateView):
    model = Produto
    fields = ["name", "category", "price"]
    success_url = reverse_lazy("produto_list")

class ProdutoUpdateView(UpdateView):
    model = Produto
    fields = ["name", "category", "price"]
    success_url = reverse_lazy("produto_list")

class ProdutoDeleteView(DeleteView):
    model = Produto
    success_url = reverse_lazy("produto_list")
```

PASSO 20: Criando a url para apagar um produto.

PASSO 21: Criando template de confirmação para apagar um produto.

```

1 <h1>Apagar Produto</h1>
2
3 <p>Você tem certeza que deseja apagar o produto <strong>{{ object.name }}</strong>?</p>
4
5 <form method="post">
6   {% csrf_token %}
7
8   <button type="submit">Sim</button>
9
10  <a href="{% url 'produto_list' %}">Cancelar</a>
11
12 </Form>
13

```

PASSO 22: Adicionando o botão no template de listagem de produtos.

```

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     <title>Produtos</title>
7   </head>
8   <body>
9
10  <h2>Produtos</h2>
11
12  <ul>
13    {% for produto in produtos %}
14      <li>
15        {{ produto.name }} ({{ produto.category }}) - R$ {{ produto.price }}
16        <br><a href="{% url 'produto_update' produto.pk %}">Editar</a> |
17        <a href="{% url 'produto_delete' produto.pk %}">Apagar</a>
18        <br><br>
19      </li>
20    {% empty %}
21    <li>Nenhum produto cadastrado.</li>
22  {% endfor %}
23
24
25  <a href="{% url 'produto_create' %}">Adicionar Produto</a>
26
27 </body>
28 </html>

```

PASSO 23: Testando a função de apagar um produto.

The screenshot shows a Windows desktop environment with a code editor and a web browser window.

Code Editor (VS Code):

- File Explorer:** Shows the project structure for 'P2' with files like `model.py`, `view.py`, `urls.py`, `produto_list.html`, `produto_form.html`, `produtos.py`, `produtos.migrations`, `produtos.admin.py`, `produtos.forms.py`, `produtos.list.html`, `produtos.urls.py`, `produtos.views.py`, `__init__.py`, `admin.py`, `apps.py`, `models.py`, `settings.py`, `utils.py`, and `urls.py`.
- Terminal:** Displays command-line logs for a Django application, including requests for URLs like `/produtos/`, `/produtos/delete/3/`, and `/produtos/create/`.

Browser Window:

- Title Bar:** Shows the title 'Produtos' and the URL '127.0.0.1:8000'.
- Content Area:**
 - Section Header:** 'Produtos'
 - List:** A list of products:
 - Computador (Informatica) - R\$ 5000,00
Editar | Apagar
 - Celular MODIFICADO (Comunicacoes) - R\$ 2850,00
Editar | Apagar
 - Text:** 'Adicionar Produto'