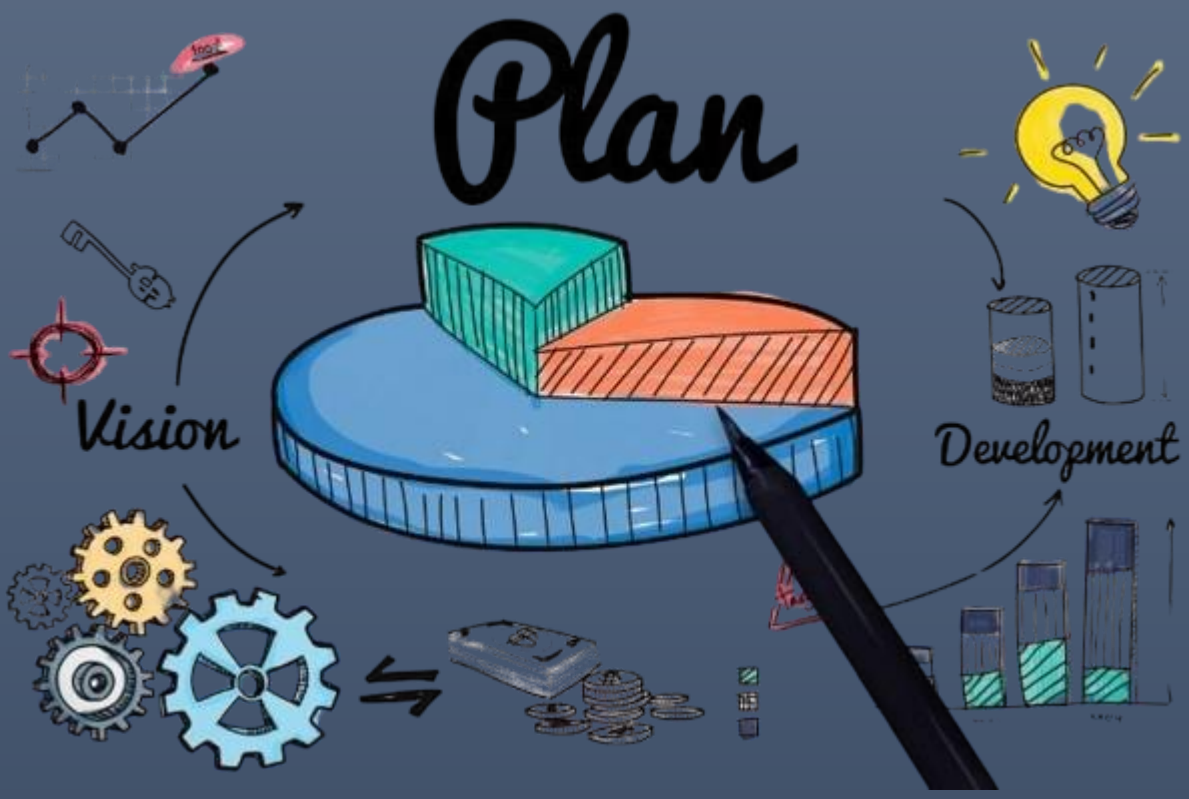


Proyecto VIII

UNIDAD 3

Actividad 3.1 Desarrollo del Backend

Profesor (a): Úrsula del Pilar González Robles

Clave de materia: IH740_2024B

Estudiante: Erick Paz López

Código de estudiante: 220966416

Proceso de Desarrollo del Backend para la Gestión de Inventarios

El desarrollo del backend para la aplicación de gestión de inventarios de una tienda de abarrotes se realizó de acuerdo con los lineamientos establecidos en la actividad integradora anterior, utilizando el framework Django. A continuación, describo el proceso detallado de desarrollo, las dificultades encontradas y cómo fueron resueltas, además de los cambios que se realizaron en la planeación del proyecto.

1. Preparación del Entorno de Desarrollo

El primer paso fue crear y configurar un entorno virtual para aislar las dependencias del proyecto. Esto fue crucial para evitar conflictos con otros proyectos y mantener el entorno controlado. Luego, se instaló el framework Django, lo cual no presentó mayores complicaciones. Sin embargo, en un principio, olvidé activar correctamente el entorno virtual, lo que generó errores al intentar instalar dependencias. Esta situación fue resuelta verificando las rutas y activando el entorno con el comando adecuado.

2. Creación del Proyecto Django y la Aplicación "Inventarios"

Después de configurar el entorno, se creó el proyecto Django utilizando el comando django-admin startproject mi_proyecto. Posteriormente, se creó una aplicación específica para la gestión de inventarios llamada inventarios. Este proceso se desarrolló sin dificultades, aunque fue necesario registrar la aplicación manualmente en el archivo de configuración de Django (settings.py), lo que fue un recordatorio importante sobre la correcta integración de aplicaciones en Django.

3. Diseño y Creación del Modelo Producto

El siguiente paso fue diseñar el modelo Producto, el cual define las propiedades de cada producto en el inventario, tales como nombre, descripción, precio, cantidad, y fecha de creación. En este punto, no hubo complicaciones técnicas, ya que el diseño del modelo era simple. Se definieron los tipos de datos más adecuados para cada campo, y se crearon las migraciones necesarias para reflejar este diseño en la base de datos. Una vez hechas las migraciones, se verificó que las tablas en la base de datos se crearan correctamente.

4. Creación de Formularios y Vistas

Para interactuar con los productos, fue necesario desarrollar formularios que permitieran la creación y edición de productos desde la interfaz web. Aquí surgió un pequeño inconveniente al definir los formularios, ya que al principio olvidé importar el modelo Producto dentro del archivo de formularios. Esto generaba errores de importación y validación. La solución fue revisar los archivos involucrados, asegurándome de que todos los componentes estuvieran correctamente conectados. Una vez solucionado, se implementaron las vistas para listar y agregar productos.

5. Desarrollo de las Plantillas HTML

La creación de las plantillas HTML fue un proceso sencillo, ya que sólo fue necesario crear dos archivos: uno para listar los productos (listar_productos.html) y otro para permitir agregar nuevos productos (agregar_producto.html). Sin embargo, durante las pruebas iniciales, se produjo un error de ruta en la navegación entre las vistas, ya que las URLs no estaban correctamente vinculadas en las plantillas. Este problema se resolvió configurando las rutas de manera explícita y verificando los nombres de las URL en los archivos correspondientes.

6. Pruebas del Sistema y Ajustes

El servidor de desarrollo de Django se inició sin problemas, y las funcionalidades básicas del sistema fueron probadas exitosamente. Se pudieron agregar, listar y gestionar productos correctamente. Durante las pruebas se realizaron pequeños ajustes a los mensajes de validación y al formato de presentación de los productos en la interfaz, con el fin de mejorar la experiencia del usuario. No fue necesario hacer cambios mayores en la planificación general del proyecto.

7. Dificultades y Soluciones

Una de las principales dificultades surgió al intentar crear el administrador de Django, ya que inicialmente no se ejecutó correctamente el comando para crear un superusuario debido a errores de configuración en la terminal. Esto se solucionó revisando los pasos anteriores y corrigiendo el entorno de trabajo para que el comando pudiera ejecutarse sin inconvenientes.

Además, uno de los errores más recurrentes estuvo relacionado con las rutas y la configuración de URLs, especialmente al integrar varias vistas y plantillas. Al principio, algunas URL estaban mal escritas o no enlazaban correctamente. La solución fue estructurar claramente las rutas en urls.py y usar siempre los nombres de rutas para garantizar que las vistas y plantillas estuvieran correctamente conectadas.

8. Conclusión

El proceso de desarrollo del backend para la gestión de inventarios de una tienda de abarrotes se llevó a cabo de manera exitosa, aunque surgieron algunos desafíos menores relacionados con la configuración del entorno y el manejo de rutas. Estos problemas fueron resueltos a través de revisiones detalladas de los archivos y las configuraciones involucradas. La estructura del proyecto fue mantenida según lo planeado inicialmente, sin necesidad de hacer cambios significativos en la planificación general. Con el backend en funcionamiento, el siguiente paso será integrar el frontend y realizar pruebas adicionales para garantizar una experiencia de usuario fluida.

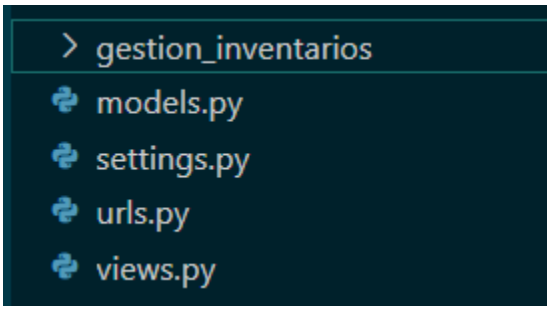
Este proyecto ha proporcionado una experiencia valiosa en el desarrollo de aplicaciones web utilizando Django, permitiendo afianzar conocimientos sobre modelos, vistas, formularios, plantillas y la gestión del ciclo de vida de una aplicación.

Descargar el framework Django

```
PS C:\Users\Erick\Desktop\9no SEMESTRE\MATERIAS\Proyecto VIII\ACTIVIDADES\Actividad 3.1\codigo> pip install django
>>
Collecting django
  Downloading Django-5.1.2-py3-none-any.whl.metadata (4.2 kB)
Collecting asgiref<4,>=3.8.1 (from django)
  Downloading asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from django)
  Downloading sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Collecting tzdata (from django)
  Downloading tzdata-2024.2-py2.py3-none-any.whl.metadata (1.4 kB)
Downloading Django-5.1.2-py3-none-any.whl (8.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.3/8.3 MB 2.8 MB/s eta 0:00:00
Downloading asgiref-3.8.1-py3-none-any.whl (23 kB)
Downloading sqlparse-0.5.1-py3-none-any.whl (44 kB)
Downloading tzdata-2024.2-py2.py3-none-any.whl (346 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.8.1 django-5.1.2 sqlparse-0.5.1 tzdata-2024.2
PS C:\Users\Erick\Desktop\9no SEMESTRE\MATERIAS\Proyecto VIII\ACTIVIDADES\Actividad 3.1\codigo> S
```

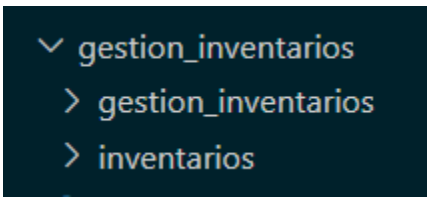
Crear el proyecto Django

```
PS C:\Users\Erick\Desktop\9no SEMESTRE\MATERIAS\Proyecto VIII\ACTIVIDADES\Actividad 3.1\codigo> django-admin startproject gestion_inventarios
>>
PS C:\Users\Erick\Desktop\9no SEMESTRE\MATERIAS\Proyecto VIII\ACTIVIDADES\Actividad 3.1\codigo>
```



Crear la aplicación Django

```
PS C:\Users\Erick\Desktop\9no SEMESTRE\MATERIAS\Proyecto VIII\ACTIVIDADES\Actividad 3.1\codigo\gestion_inventarios> python manage.py startapp inventarios
```



Agregar la nueva aplicación

Ejecutar el servidor de desarrollo:

```
(.venv) PS C:\Users\Erick\Desktop\9no SEMESTRE\MATERIAS\Proyecto VIII\ACTIVIDADES\Actividad 3.1\codigo\mi_proyecto> python manage.py runserver
>>
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 22, 2024 - 11:40:51
Django version 5.1.2, using settings 'mi_proyecto.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

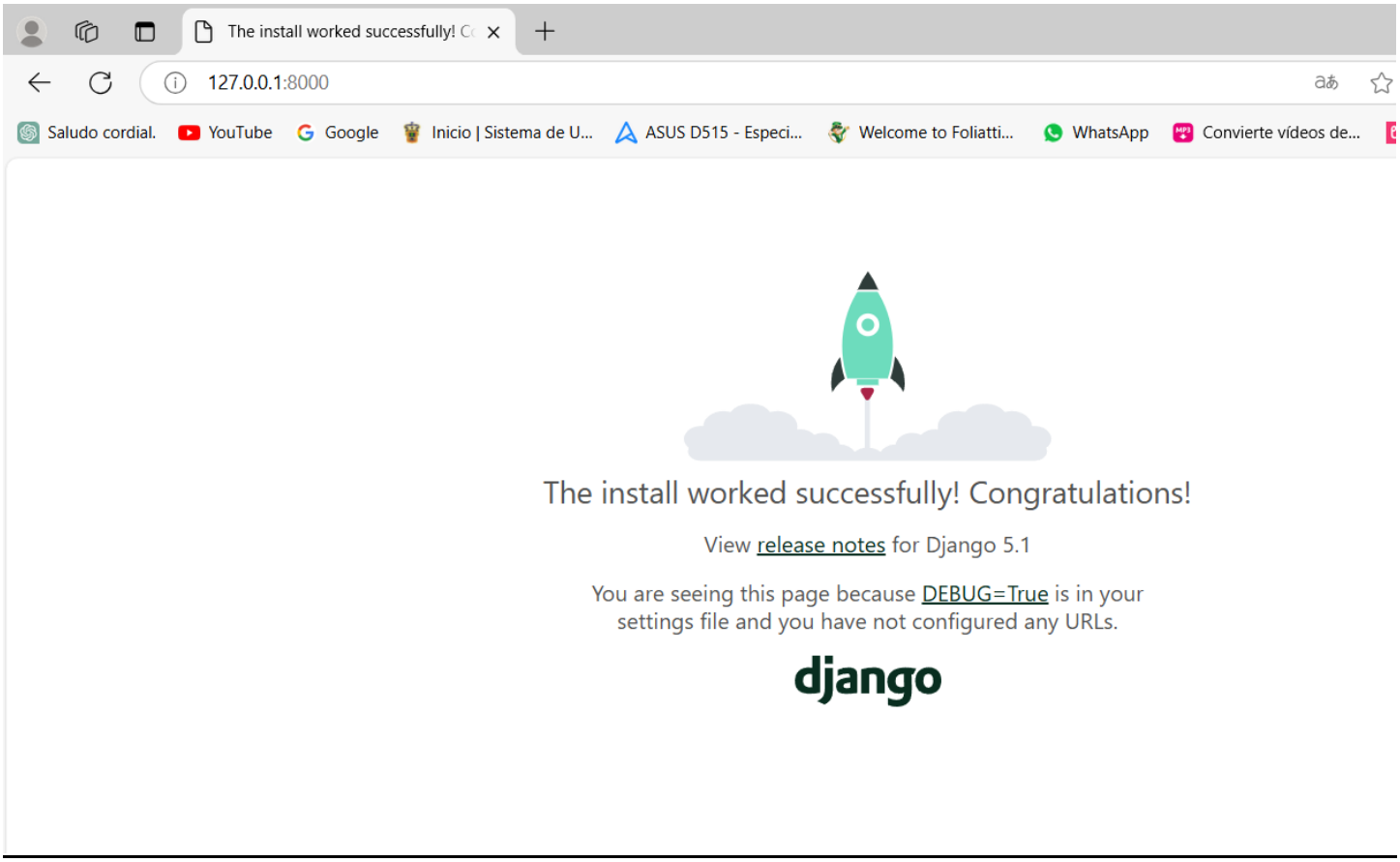
```python
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'inventarios',
```

Definir un modelo

gestion_inventarios > inventarios > models.py > ...

```python

from django.db import models

class Producto(models.Model):
    nombre = models.CharField(max_length=100)
    cantidad = models.PositiveIntegerField()
    precio = models.DecimalField(max_digits=10, decimal_places=2)

    def __str__(self):
        return self.nombre

```

```
System check identified no issues (0 silenced).
October 22, 2024 - 11:45:49
Django version 5.1.2, using settings 'mi_proyecto.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

[22/Oct/2024 11:45:51] "GET / HTTP/1.1" 200 50
```



Modelo para productos

settings.py models.py

mi_proyecto > inventarios > models.py > ...

```python

# Create your models here.
from django.db import models

class Producto(models.Model):
    nombre = models.CharField(max_length=100)
    descripcion = models.TextField()
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    cantidad = models.IntegerField()
    fecha_creacion = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.nombre

```

Crear formulario para agregar productos.

settings.py | models.py | forms.py

mi_proyecto > inventarios > forms.py > ...

```python
from django import forms
from .models import Producto

class ProductoForm(forms.ModelForm):
    class Meta:
        model = Producto
        fields = ['nombre', 'descripcion', 'precio', 'cantidad']

```

Vistas para listar y agregar productos

mi_proyecto > inventarios > views.py > ...

```python
from django.shortcuts import render

# Create your views here.
from django.shortcuts import render, redirect
from .models import Producto
from .forms import ProductoForm

def listar_productos(request):
    productos = Producto.objects.all()
    return render(request, 'listar_productos.html', {'productos': productos})

def agregar_producto(request):
    if request.method == 'POST':
        form = ProductoForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('listar_productos')
    else:
```

URL para las vistas

settings.py | models.py | forms.py | views.py | urls.py

mi_proyecto > inventarios > urls.py > ...

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

urlpatterns = [
    path('', views.listar_productos, name='listar_productos'),
    path('agregar/', views.agregar_producto, name='agregar_producto'),
]
```

Plantilla HTML

settings.py | models.py | forms.py | views.py | urls.py ...\inventarios | urls.py ...\mi_proyecto | lista

mi_proyecto > inventarios > templates > listar_productos.html > ...

```html
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Productos</title>
</head>
<body>
    <h1>Lista de Productos</h1>
    <ul>
        {% for producto in productos %}
            <li>{{ producto.nombre }} - ${{ producto.precio }} (Cantidad: {{ producto.cantidad }})</li>
        {% endfor %}
    </ul>
    <a href="{% url 'agregar_producto' %}">Agregar Producto</a>
</body>
</html>
```

Correr el servidor de la aplicación: http://127.0.0.1:8000/inventarios/

Agregar Producto

Nombre: aceite

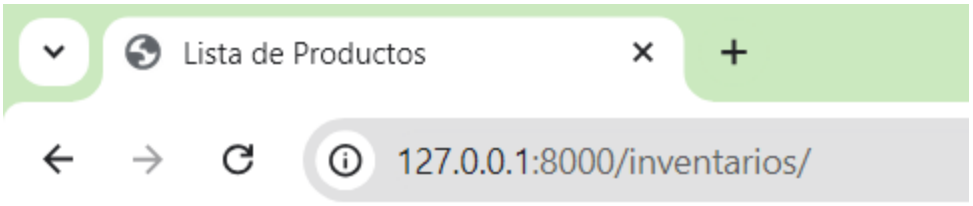
Descripcion: producto hecho a base de maiz

Precio:

Cantidad:

Guardar

Volver a la lista



Lista de Productos

- aceite - $25.00 (Cantidad: 2)

Agregar Producto