

## Lección 6: Actividad de portafolio nro. 6

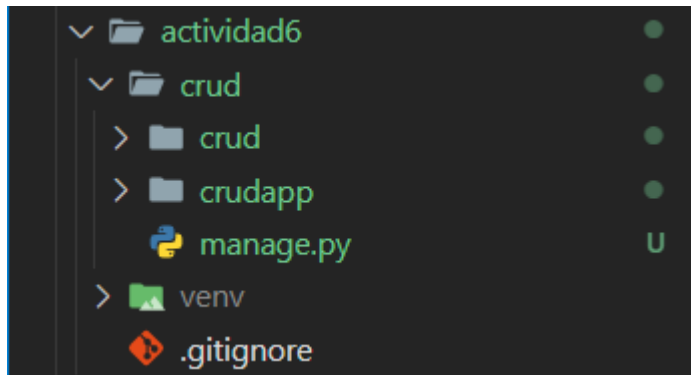
### “Implementación de CRUD en Django”

Nicolás Jofré Andrade - Desarrollo Full-stack Python  
05-2024

**Objetivo:** Crear una aplicación Django que realice operaciones CRUD (crear, leer, actualizar y borrar) sobre una entidad específica, abordando aspectos como enrutamiento, seguridad CSRF y uso de Django ORM.

#### 1. Creación de la aplicación:

- Crea una nueva aplicación llamada 'crudapp'.



#### 2. Definición del modelo:

- Define un modelo 'Producto' en 'models.py' con campos como 'nombre', 'precio' y 'cantidad'.

```
models.py U X
1  from django.db import models
2
3  class Producto(models.Model):
4      nombre = models.CharField(max_length=100)
5      precio = models.DecimalField(max_digits=10, decimal_places=2)
6      cantidad = models.PositiveIntegerField()
7
8      def __str__(self):
9          return self.nombre
10
```

### 3. Migraciones y aplicación del modelo:

- Ejecuta las migraciones para aplicar el modelo a la base de datos.

TERMINAL

```
(venv) PS C:\Users\DivalCL\Desktop\dev\portafolio-py-mod7\actividad6\crud> python manage.py makemigrations  
No changes detected
```

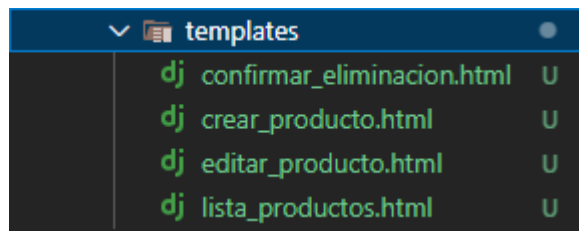
### 4. Vistas y templates para CRUD:

- Crea vistas en 'views.py' para realizar operaciones CRUD sobre los productos.

```
views.py U X  
1  from django.shortcuts import render, redirect, get_object_or_404  
2  from .models import Producto  
3  from .forms import ProductoForm  
4  
5  def lista_productos(request):  
6      productos = Producto.objects.all()  
7      return render(request, 'lista_productos.html', {'productos': productos})  
8  
9  def crear_producto(request):  
10     if request.method == 'POST':  
11         form = ProductoForm(request.POST)  
12         if form.is_valid():  
13             form.save()  
14             return redirect('lista-productos')  
15     else:  
16         form = ProductoForm()  
17     return render(request, 'crear_producto.html', {'form': form})  
18  
19  def editar_producto(request, pk):  
20     producto = get_object_or_404(Producto, pk=pk)  
21     if request.method == 'POST':  
22         form = ProductoForm(request.POST, instance=producto)  
23         if form.is_valid():  
24             form.save()  
25             return redirect('lista-productos')  
26     else:  
27         form = ProductoForm(instance=producto)  
28     return render(request, 'editar_producto.html', {'form': form})  
29  
30  def confirmar_eliminacion(request, pk):  
31     producto = get_object_or_404(Producto, pk=pk)  
32     if request.method == 'POST':  
33         producto.delete()  
34         return redirect('lista-productos')  
35     return render(request, 'confirmar_eliminacion.html', {'producto': producto})  
36
```

- Crea las plantillas HTML correspondientes en el directorio

'templates/crudapp/'

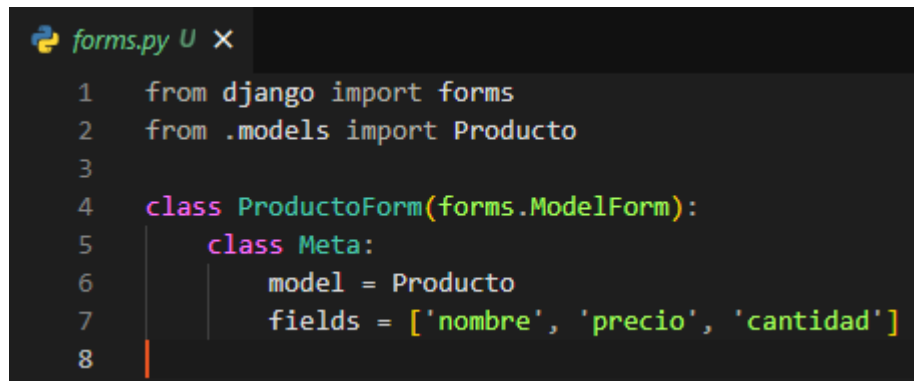


A screenshot of a file explorer window showing the 'templates' directory. It contains four files, each with a Django icon and a status indicator 'U'.

templates	
confirmar_eliminacion.html	U
crear_producto.html	U
editar_producto.html	U
lista_productos.html	U

5. Formulario para crear y editar productos:

- Crea un formulario 'ProductoForm' en 'forms.py' para facilitar la creación y edición de productos.

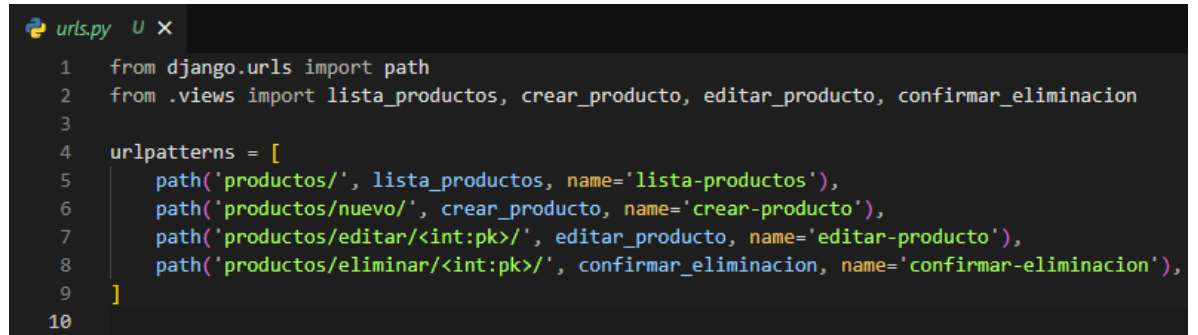


A screenshot of a code editor showing the 'forms.py' file. The code defines a 'ProductoForm' class that inherits from 'forms.ModelForm'. It includes a 'Meta' class with 'model = Producto' and 'fields = ['nombre', 'precio', 'cantidad']'.

```
1 from django import forms
2 from .models import Producto
3
4 class ProductoForm(forms.ModelForm):
5     class Meta:
6         model = Producto
7         fields = ['nombre', 'precio', 'cantidad']
8
```

6. Configuración de Urls:

- Configura las Urls en 'urls.py' de la aplicación.



A screenshot of a code editor showing the 'urls.py' file. The code imports 'path' from 'django.urls' and views from the application. It defines a list of 'urlpatterns' with four entries for listing, creating, editing, and deleting products.

```
1 from django.urls import path
2 from .views import lista_productos, crear_producto, editar_producto, confirmar_eliminacion
3
4 urlpatterns = [
5     path('productos/', lista_productos, name='lista-productos'),
6     path('productos/nuevo/', crear_producto, name='crear-producto'),
7     path('productos/editar/<int:pk>/', editar_producto, name='editar-producto'),
8     path('productos/eliminar/<int:pk>/', confirmar_eliminacion, name='confirmar-eliminacion'),
9 ]
10
```

- Incluye estas Urls en el archivo principal 'urls.py' del proyecto.

```
urls.py U X
1  """
2  URL configuration for crud project.
3
4  The `urlpatterns` list routes URLs to views. For more information please see:
5  |   https://docs.djangoproject.com/en/5.0/topics/http/urls/
6  |   Examples:
7  |   Function views
8  |       1. Add an import:  from my_app import views
9  |       2. Add a URL to urlpatterns:  path('', views.home, name='home')
10 |   Class-based views
11 |       1. Add an import:  from other_app.views import Home
12 |       2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
13 |   Including another URLconf
14 |       1. Import the include() function: from django.urls import include, path
15 |       2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
16 |   """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('', include('crudapp.urls')),
23 ]
24
```

## 7. Configuración de seguridad CSRF:

- Asegúrate de que el middleware de CSRF esté habilitado en 'settings.py'.

```
43 MIDDLEWARE = [
44     'django.middleware.security.SecurityMiddleware',
45     'django.contrib.sessions.middleware.SessionMiddleware',
46     'django.middleware.common.CommonMiddleware',
47     'django.middleware.csrf.CsrfViewMiddleware',
48     'django.contrib.auth.middleware.AuthenticationMiddleware',
49     'django.contrib.messages.middleware.MessageMiddleware',
50     'django.middleware.clickjacking.XFrameOptionsMiddleware',
51 ]
```