# Playground Avanzado (Parte I)

**CRUD**

**¿Qué es CRUD?**

En informática, CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete). También es conocido como ABM en otros entornos (Altas, Bajas y Modificaciones).

Es el acto de agregar, modificar, eliminar o acceder a datos de una base de datos.

Sin saberlo, algo de eso ya hemos realizado. Ya sabemos insertar datos y buscar datos.

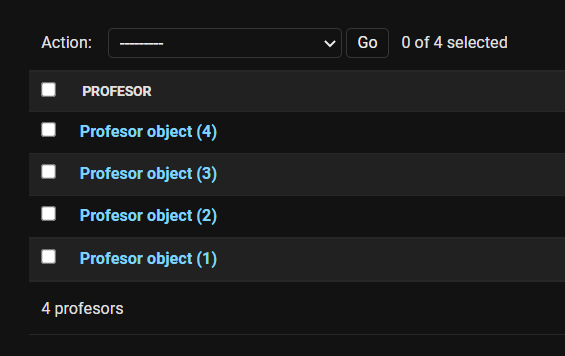
Pero ahora llegó la hora de formalizar un poco esos conceptos y agregar lo que nos falta. Borrar y modificar datos de la BD.

Desde el panel de administración es muy simple. La idea es hacerlo desde nuestras vistas y darle nuestra impronta.

**Arrancando con CRUD**

Primero vamos a agregar datos a alguna de nuestras tablas por medio del panel, así se ven mejor los resultados que obtendremos.

Por ejemplo, voy a crear 4 profesores, y trabajaré todo el ejemplo con los profesores.

Primero detalles que no tienen que ver con CRUD, ya agregué 4 profesores por medio del panel de Admin. 

Esto lo habíamos pasado por alto, pero los objetos generados en la BD se ven poco agradables y no se puede entender cuál información tenemos.

Arreglarlo es muy simple

👇

Solo hay que agregar el método \_\_str\_\_(self): en la clase que quieres que se vea distinta ( o en todas).

Veamos cómo hacerlo 👇

def \_\_str\_\_(self):

return f"Nombre: {self.nombre} - Apellido {self.apellido} - E-Mail {self.email} - Profesión {self.profesion}"



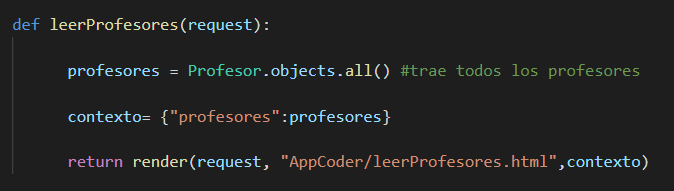
**CRUD - READ**

**Arrancando con CRUD**

Arrancamos con **READ, lectura.**

Vamos a leer todos los profesores que tenemos en nuestra base de datos.

Primero creamos la vista.



Script:

def leerProfesores(request):

profesores = Profesor.objects.all() #trae todos los profesores

contexto= {"profesores":profesores}

return render(request, "AppCoder/leerProfesores.html",contexto)

👉 Luego creamos el template y lo asociamos con **urls.py** a la vista.



path('leerProfesores', views.leerProfesores, name = "LeerProfesores")

**Template**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    {% for p in profesores %}

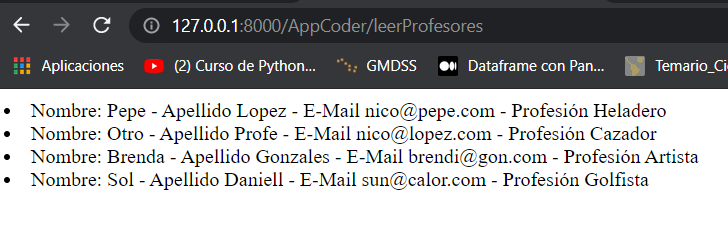
        <li>{{p}}</li>

    {% endfor %}

</body>

</html>

Nuestro resultado será el siguiente 👀

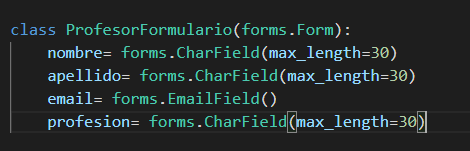
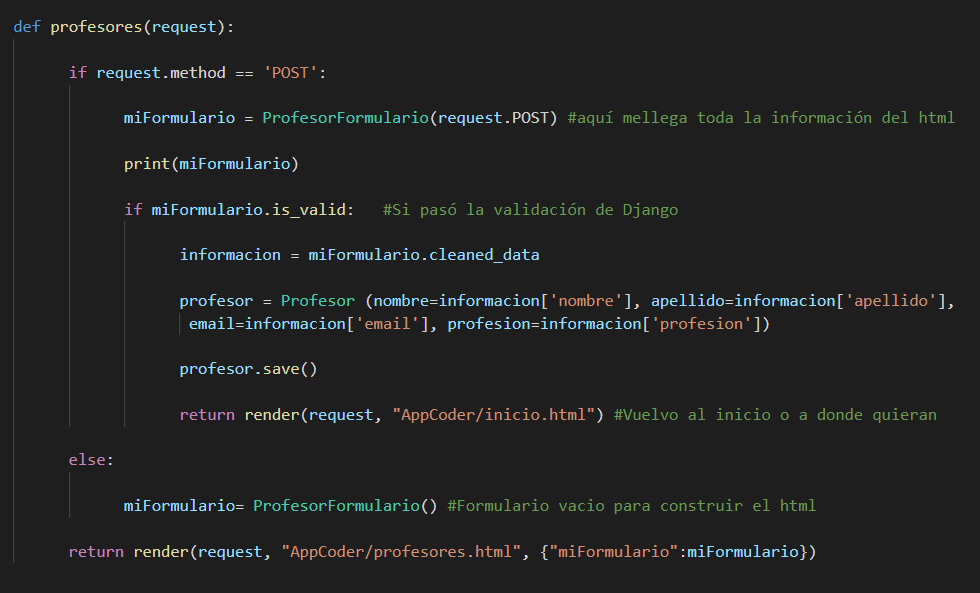


**CRUD - CREATED**

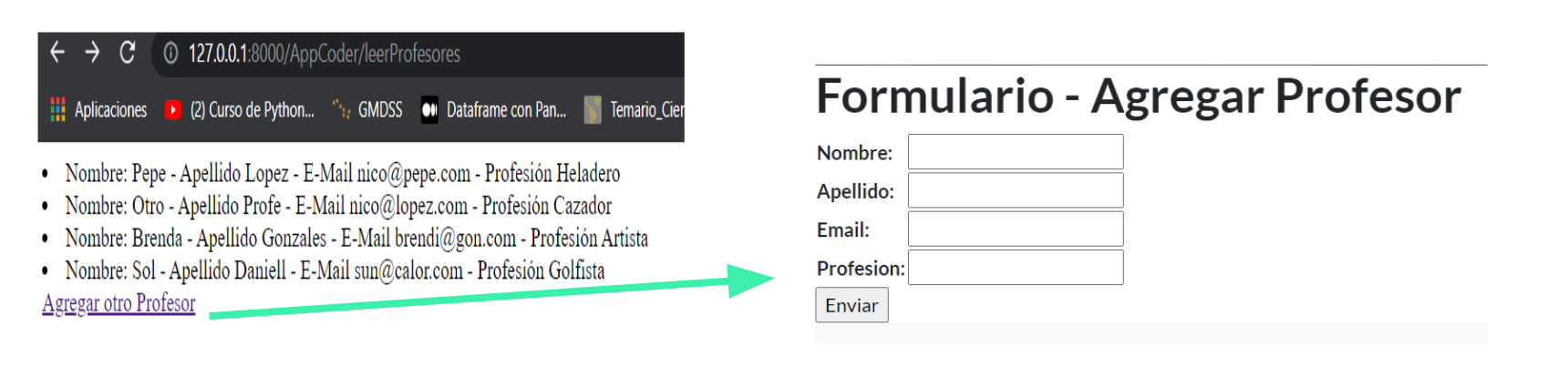
**Arrancando con CRUD**

La creación ya la sabíamos hacer, es más ya la tenemos realizada por medio de un FORMS.

Recordando todo esto ya lo tenemos 😎



Solo agregaremos un vínculo para poder agregar un profesor



**CRUD - Delete**

**Arrancando con CRUD**

Pasamos a borrar, es decir DELETE

Primero creamos la vista para buscar el dato que queremos borrar



Script

def eliminarProfesor(request, profesor\_nombre):

    profesor = Profesor.objects.get(nombre=profesor\_nombre)

    profesor.delete()

    # vuelvo al menú

    profesores = Profesor.objects.all()  # trae todos los profesores

    contexto = {"profesores": profesores}

    return render(request, "AppCoder/leerProfesores.html", contexto)

Botón

<body>

    {% for p in profesores %}

        <li>{{p}}</li>

        <button>

            <a href="{% url 'EliminarProfesor' p.nombre %}"> Eliminar</a>

        </button>

    {% endfor %}

    <br>

    <a href="profesores">Agregar otro Profesor</a>

</body>

urls.py

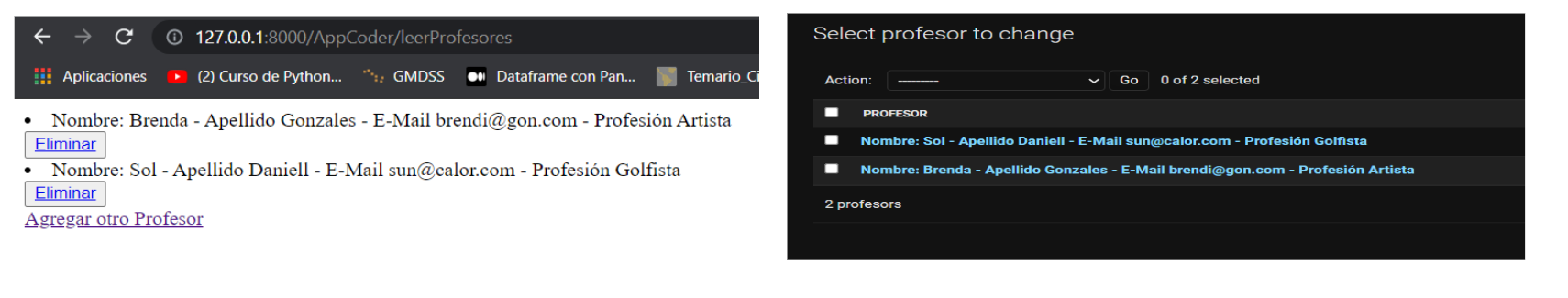
path('eliminarProfesor/<profesor\_nombre>/', views.eliminarProfesor, name="EliminarProfesor")

Texto

Descripción generada automáticamente

Apretando en el botón nuevo podemos ir borrando a los profesores

👇



**CRUD - Update**

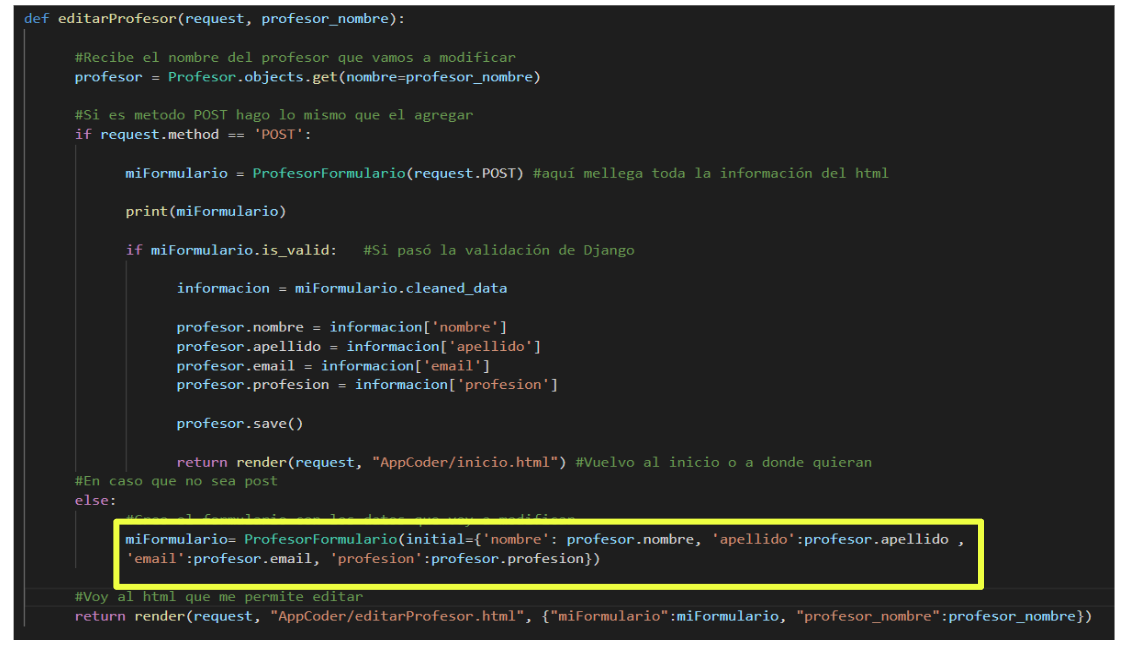
**Arrancando con CRUD**

Solo nos queda poder modificar algún dato ya existente.

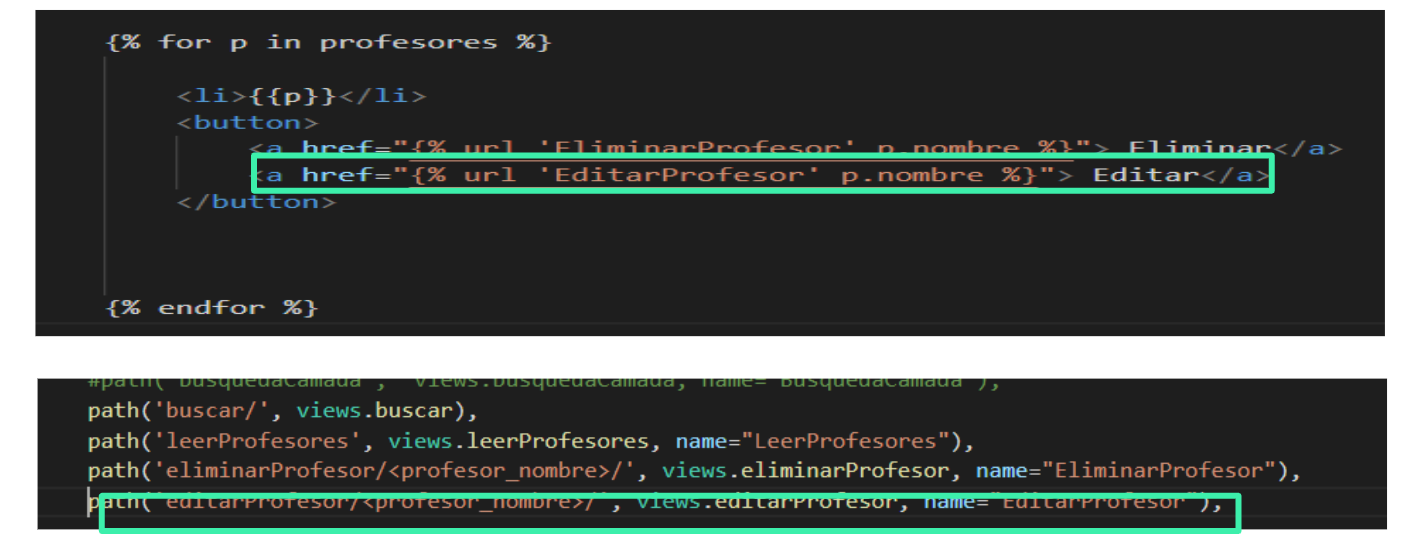
Creamos la vista,

**editarProfesor(request, profesor\_nombre)**

**Ver: Script\_Update.txt**



Luego la url y el template son muy parecidos al agregar



Script: urls.py

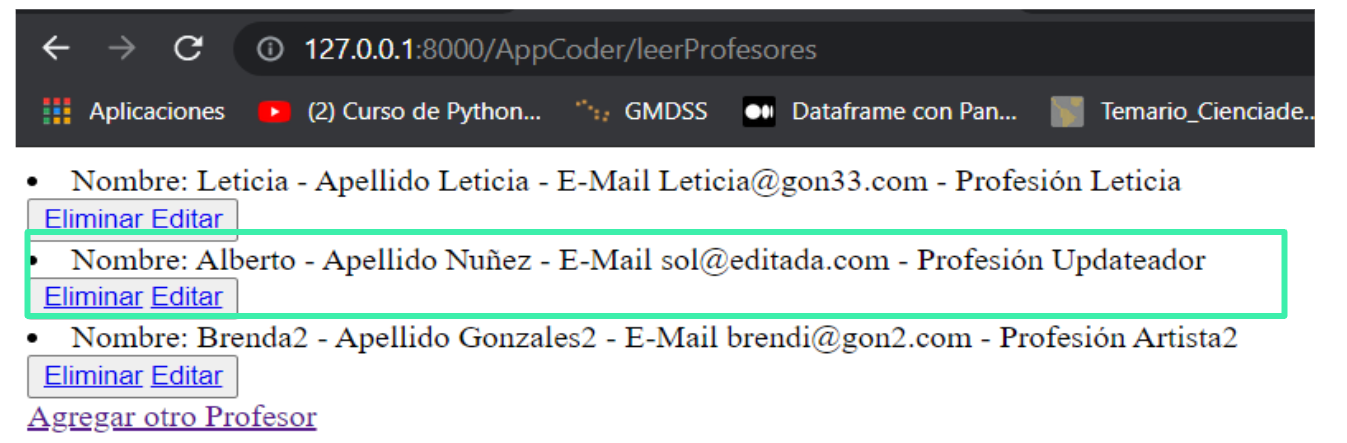
    path('editarProfesor/<profesor\_nombre>/', views.editarProfesor, name="EditarProfesor")

editarProfesor.html

**Ver: Script\_profesores.txt**



Resultado 💪



**Clases basadas en vistas**

Ya aprendimos lo que es CRUD, crear, leer, modificar y borrar datos de nuestra BD sin utilizar nuestro menú de admin de Django.

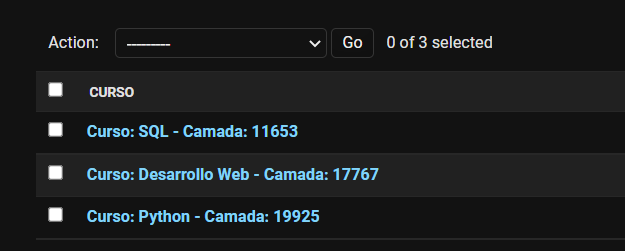
👉 Fue relativamente sencillo y pudimos hacer las 4 tareas con los Profesores, por ende podríamos hacerlo con todas las clases de nuestro modelo de la misma forma.

Pero para que ese trabajo sea aún más sencillo y menos repetitivo aparece el concepto de CBV o clases basadas en vistas, o vistas basadas en clases

(según traducciones).

**¿Listos para saber de qué se trata?**

Para entender este concepto haremos nuevamente CRUD, pero con los Cursos de la BD, así verán cómo se simplifica todo lo que hicimos anteriormente.

****

Todos los ejemplos van a partir de que en nuestra base de datos ya tenemos cargados estos tres cursos.

**Clases basadas en vistas listview**

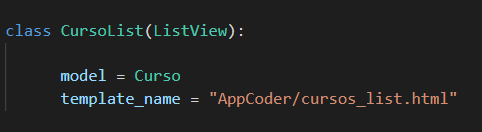
**Clases basadas en vistas**

Ahora haremos una vista que nos permita mostrar a todos los cursos.

Ya sabemos hacerlo, pero la idea es hacerlo simplificando el proceso, para eso usaremos las ListView.

Necesitamos 👇

**from django.views.generic import ListView**



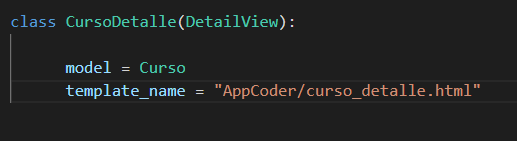
**Clases basadas en vistas detailview**

**Clases basadas en vistas**

Luego accedemos al detalle de ese **Curso**, que ya sabemos cómo hacerlo, pero podemos simplificarlo con las DetailView.

Necesitamos 👇

**from django.views.generic.detail import DetailView**



**Clases basadas en vistas createview**

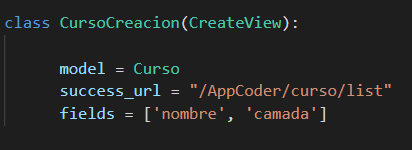
**Clases basadas en vistas**

Ahora pasamos a la creación del curso, usando las CreateView.

Necesitamos 👇

**from django.views.generic.edit import CreateView**

**from django.urls import reverse\_lazy**



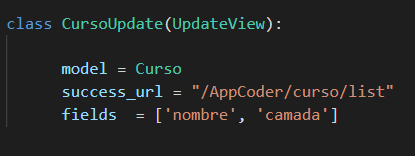
**Clases basadas en vistas updateview**

**Clases basadas en vistas**

Podemos modificar con UpdateView.

Necesitamos 👇

**from django.views.generic.edit import UpdateView**



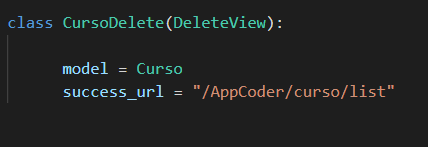
**Clases basadas en deleteview**

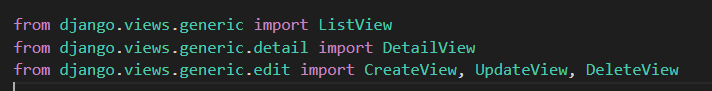
**Clases basadas en vistas**

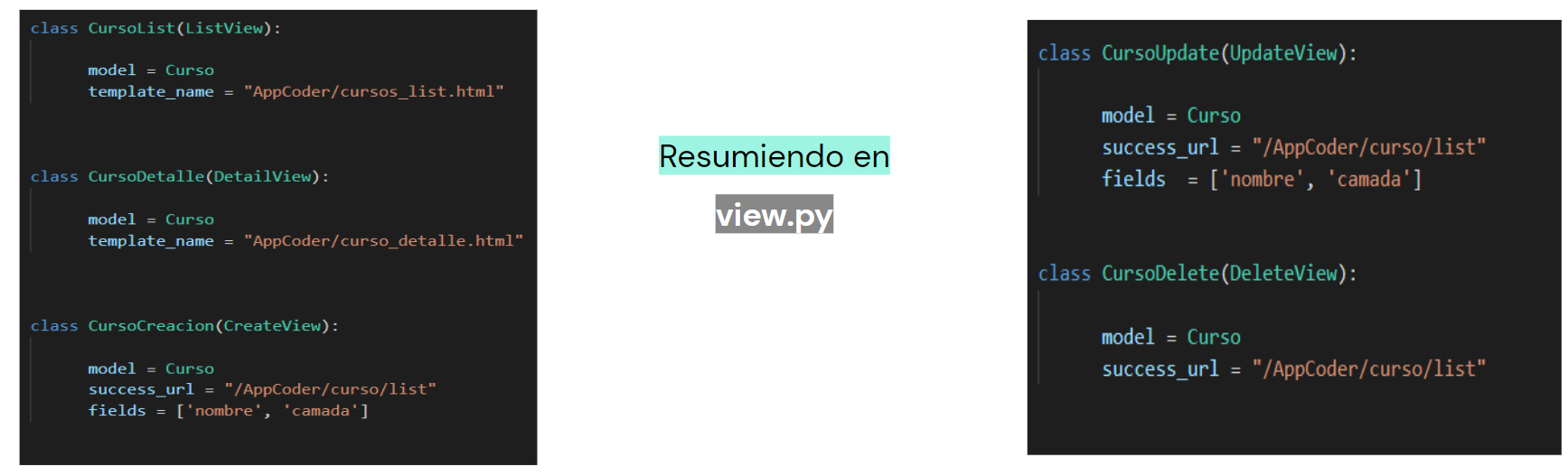
¿Saben que nos falta?, exacto, el DeleteView.

Necesitamos 👇

**from django.views.generic.edit import DeleteView**

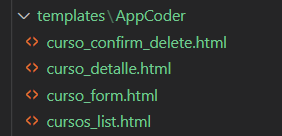
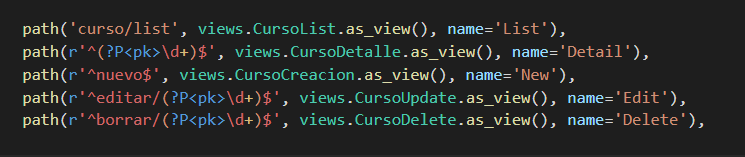






**Clases basadas en vistas URLs**

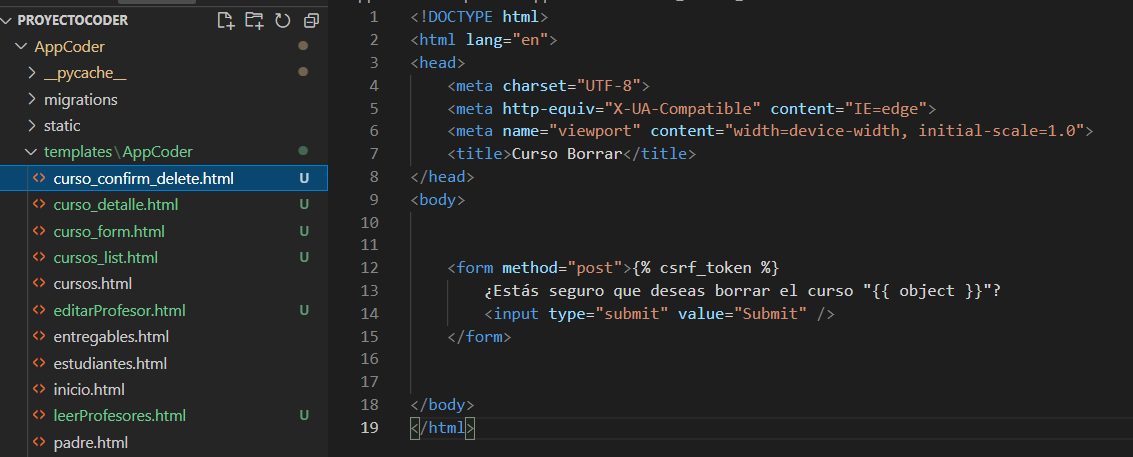
**CBV - urls - template**



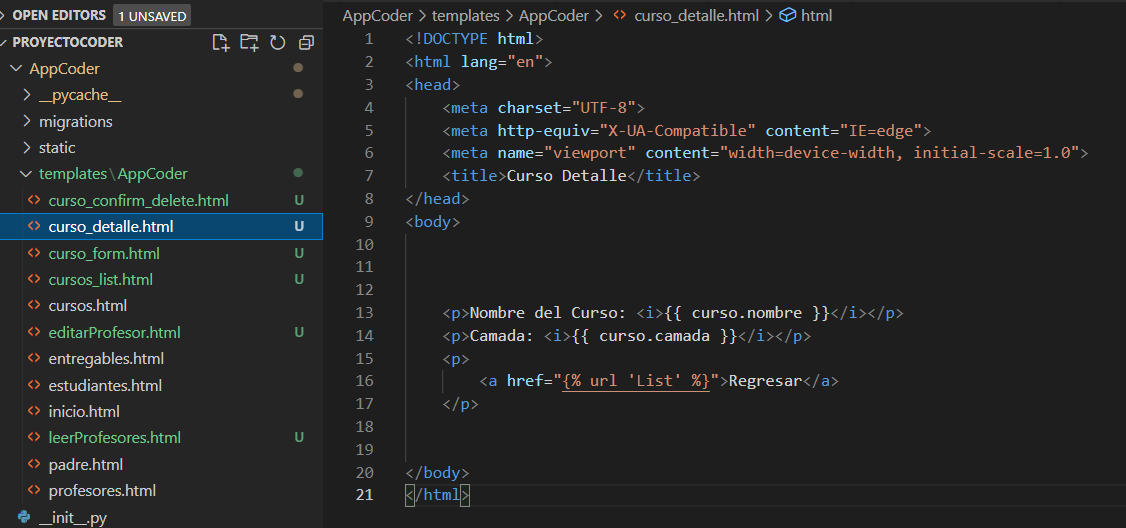
**Clases basadas en vistas**

**Plantillas**

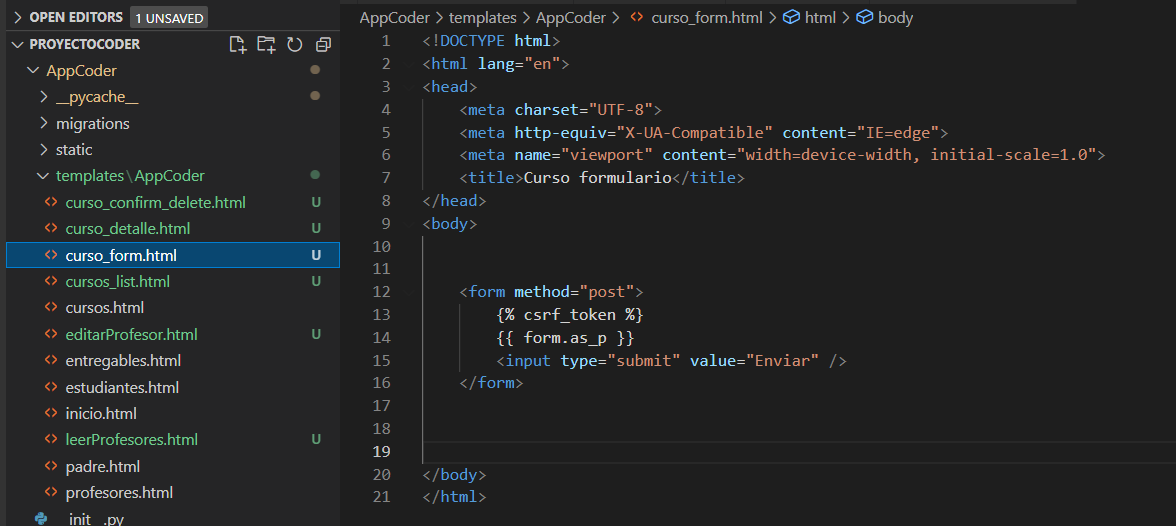
**CBV - Delete**

****

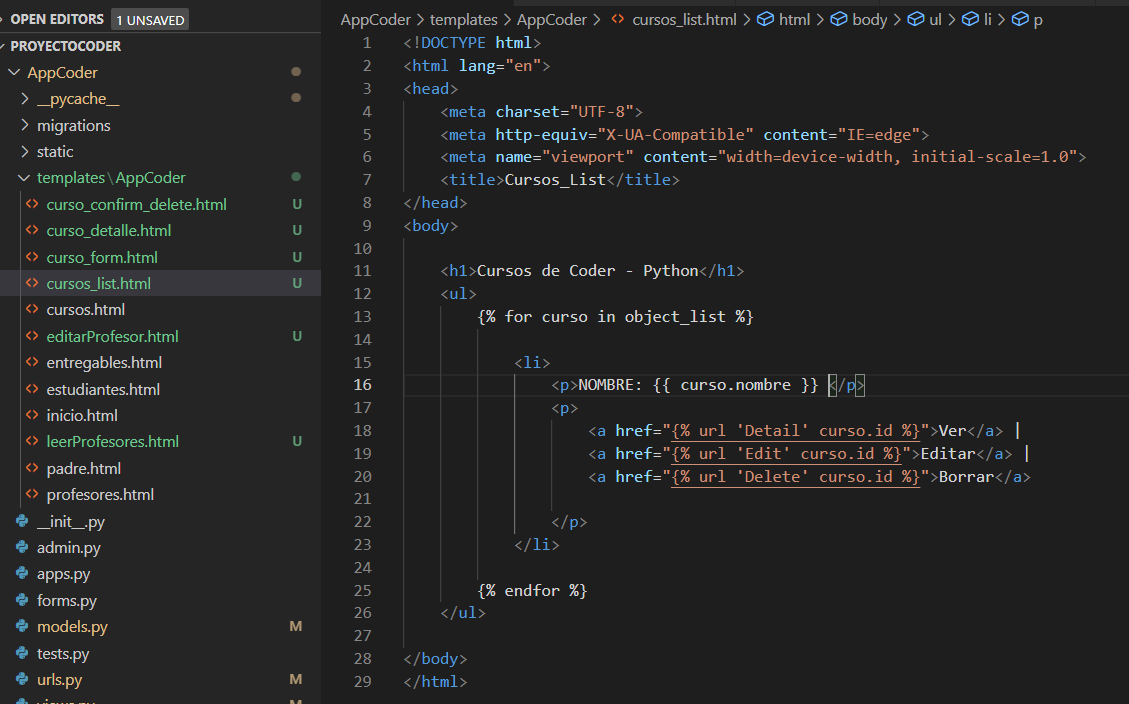
**CBV - Detalle**

****

**CBV - Formulario**

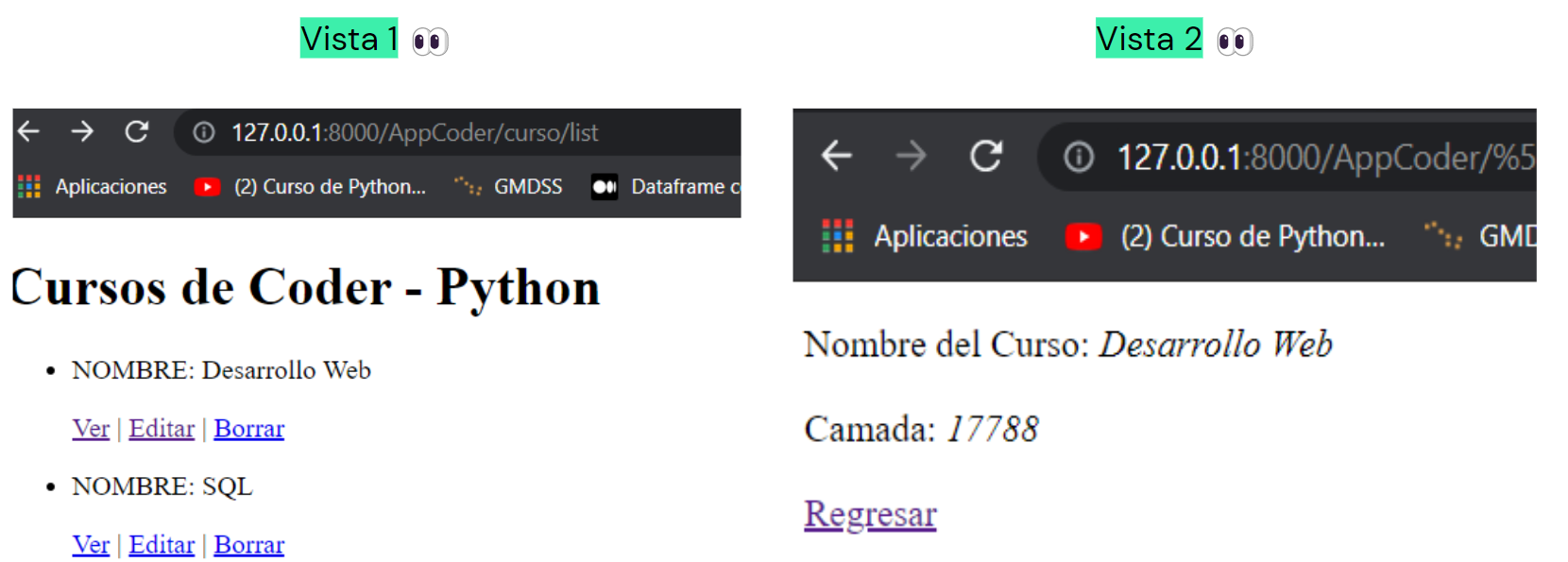
****

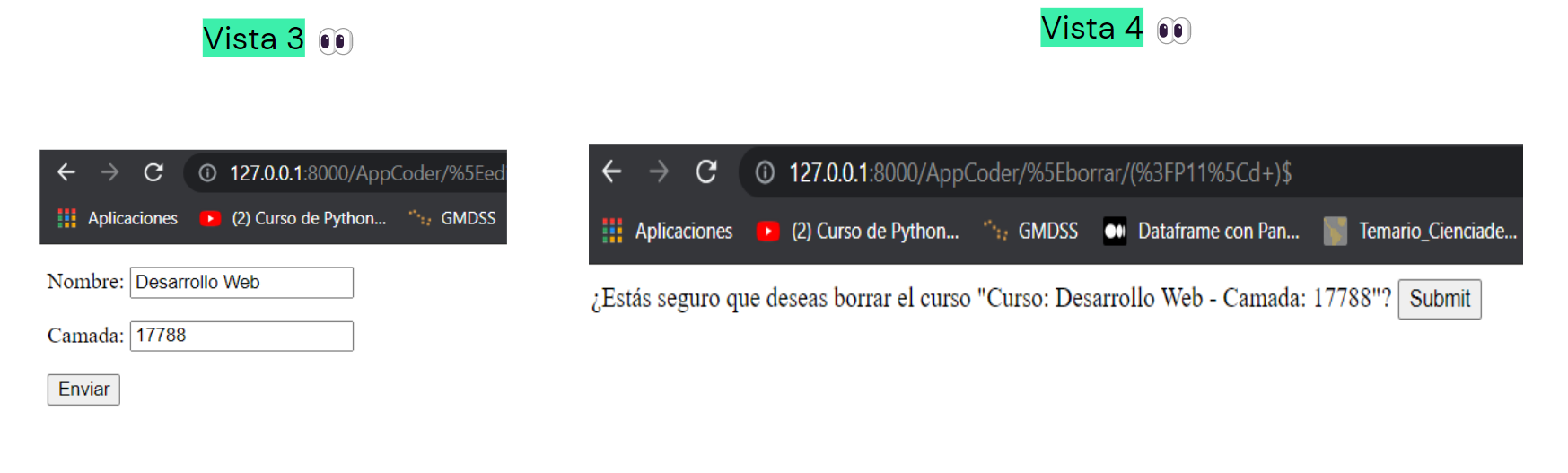
**CBV - Lista**

****

**Clases basadas en vistas de acción**

**Clases basadas en vistas**

****

****

¡Increíble! 👏

Hicimos CRUD y FORM de una forma muy simple y abreviada.

Gracias a las vistas basadas en clases, utilizando herramientas View.