Clase 18 - Django - Porfolio (parte II)

**Mejoras en las plantillas**

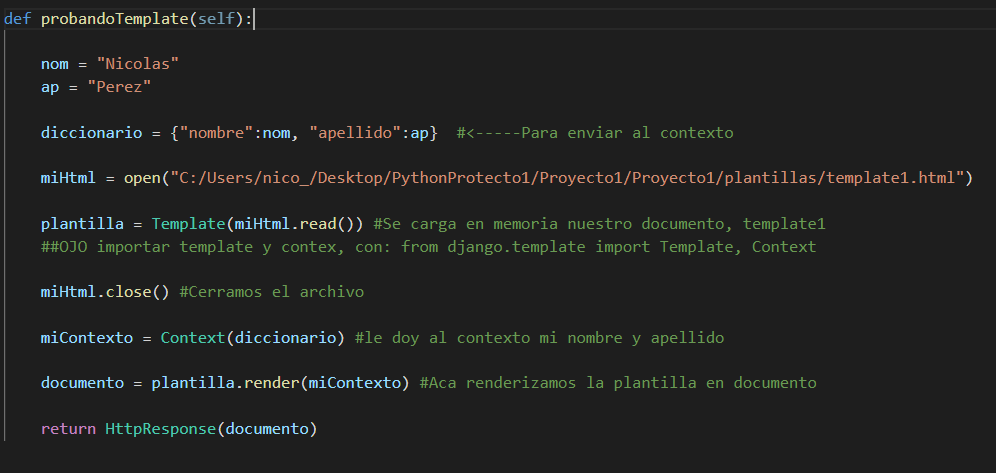
**Variables a las plantillas**

Desde una plantilla podemos hacer referencia a variables de nuestra vista. Para esto debemos darle esa variable al contexto, lo usaremos por primera vez.

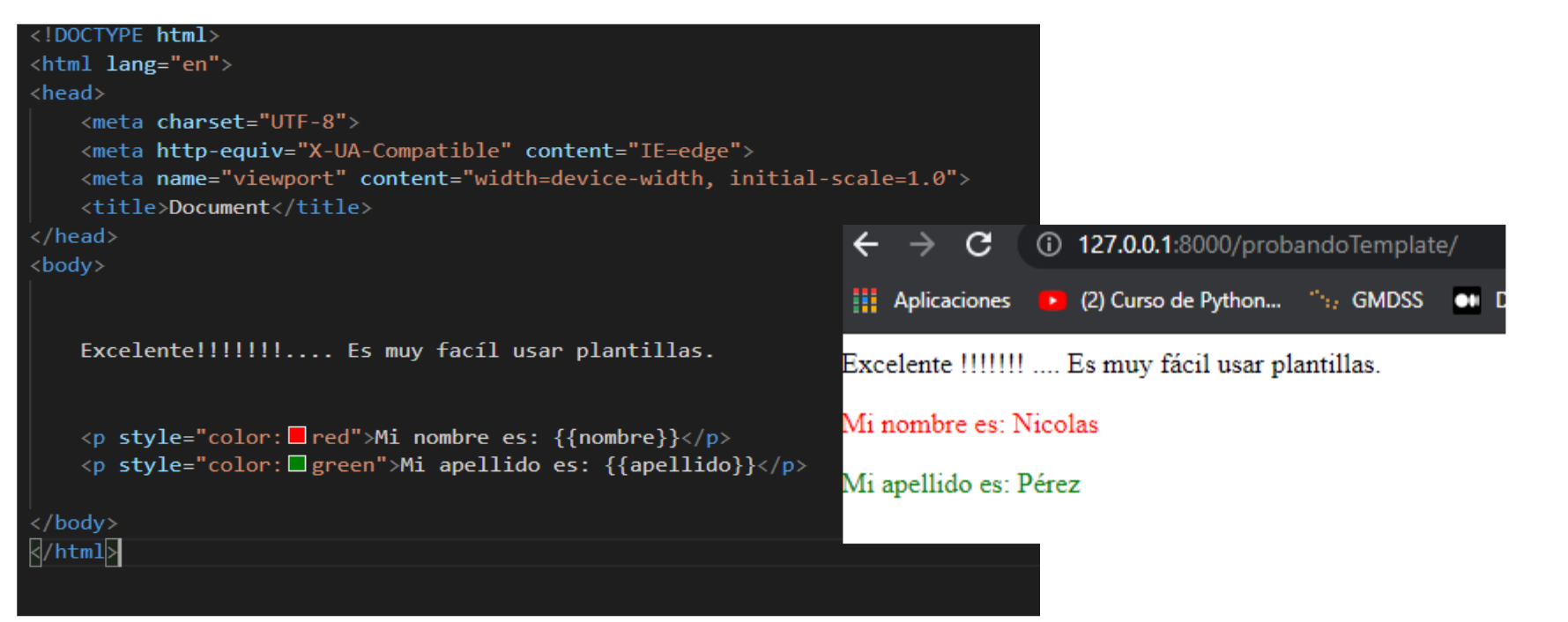
Veámoslo con nuestra función **probandoTemplate** de la clase anterior. El envío de variables al contexto se realizará desde diccionarios de python.

Por ejemplo, creamos las variables nom y ap, con ellas generamos un diccionario y se lo enviamos al contexto:

Luego en mi plantilla haremos referencia a lo enviado en el diccionario, usando **{{ xxx }}**

****

**¡A no preocuparse si no sabemos HTML, con lo que veamos, será más que suficiente!**

****

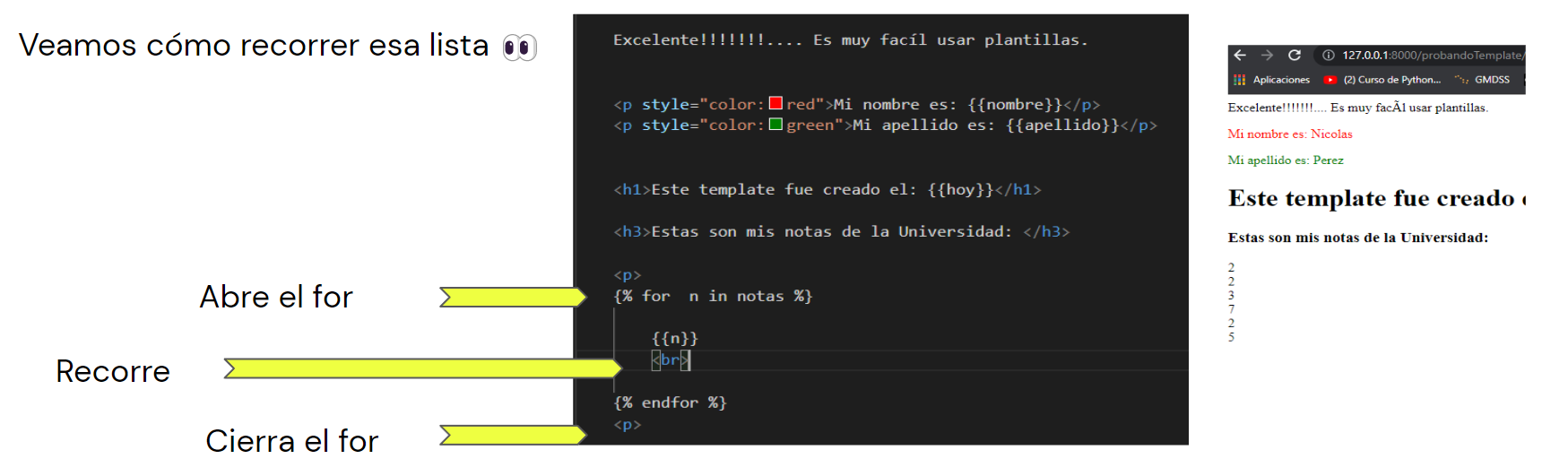
**Bucles y Condicionales**

**Las plantillas nos sorprenden**

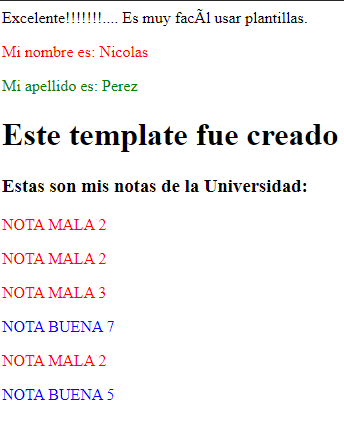
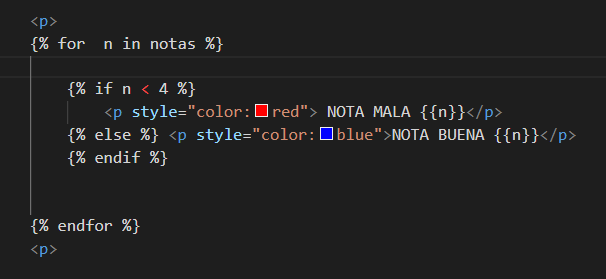
Efectivamente, se puede y es muy sencillo, veamos un ejemplo con una lista:

****

Podemos ejecutar código Python desde la plantilla gracias a Django, usando **{{}}** para variables, o **{% %}** para bucles y condicionales.



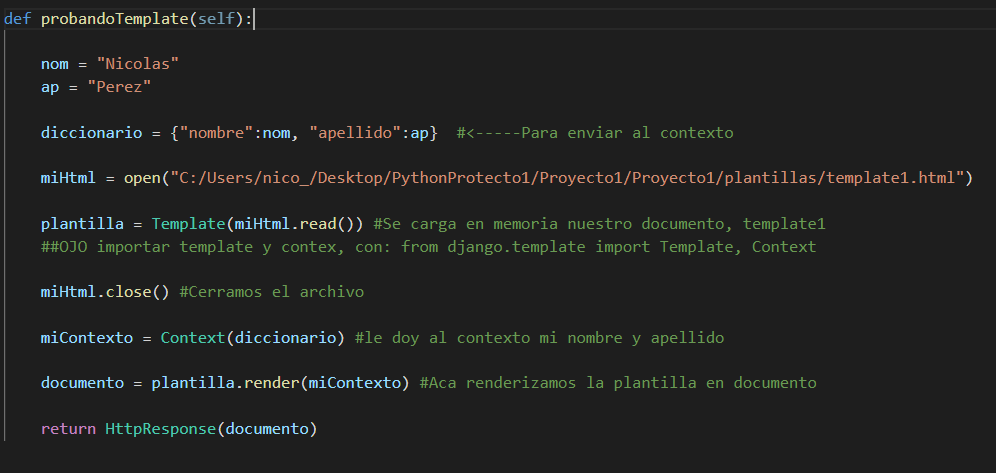
Ahora le agregaremos un condicional, si la nota es menor que 4 en rojo, en otro caso en azul:

****

Más TAG: Built-in template tags and filters | [docs.djangoproject.com](https://docs.djangoproject.com/en/3.2/ref/templates/builtins/)

**Cargadores de plantillas**

Si tuvieras que cargar muchas plantillas, hacer esto sería horrible:

****

Para solucionar ésto, aparece el concepto de **Cargador** con el que podrás manejar de forma más ordenada las plantillas.

**Cargamos plantillas desde Django**

**¿Cómo hacemos esto?** Guardamos todas las plantillas en la misma carpeta y se lo avisamos a Django, en nuestro caso la carpeta se llamaba plantillas.

1. Primero en las vistas, importamos el cargador:

**from django.template import loader**

1. Cargamos la plantilla **pero sin el open**, para eso abrimos el archivo **settings.py,** buscamos la lista TEMPLATES y en DIRS[] ponemos la ruta, en este caso:

**C:/Users/nico\_/Desktop/PythonProtecto1/Proyecto1/Proyecto1/plantillas/**

**3.** Volvemos a la vista y llamamos a nuestra plantilla con:

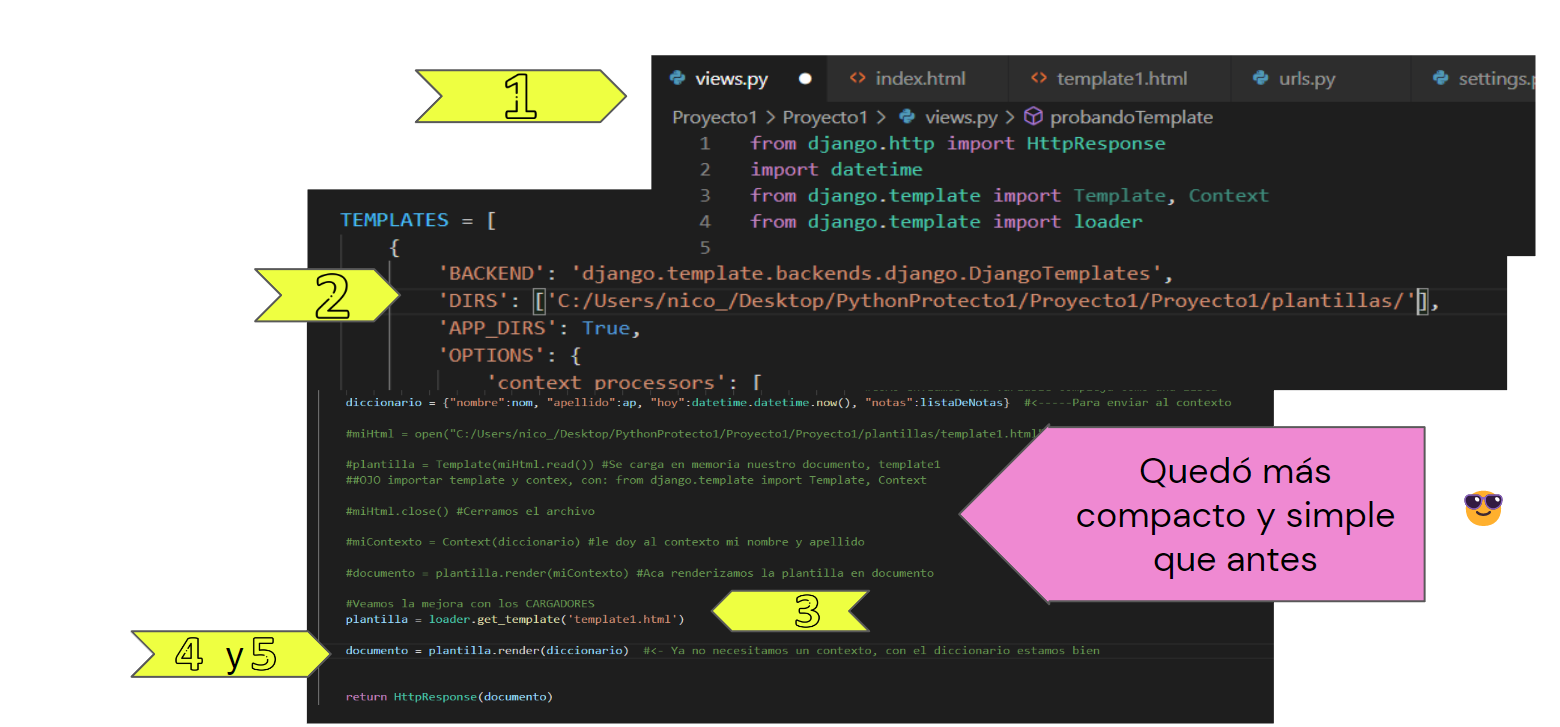
**loader.get\_pemplate(‘plantilla1.html)**

**4.** Luego renderizamos:

**render(contexto)**

**5.** El contexto debe dejar de serlo, ahora **debe** ser un diccionario.

Veamos entonces qué fue todo esto…

****

**MODELO**

**¿Qué sabemos de Django?**

Hemos visto que Django se basa en el **patrón MTV**, ya conocemos los **Templates (lo que se ve)**, **las View (la información que se pasa al template)**, ahora nos falta saber qué es el **modelo.**

**¿Qué es el MODELO?**

**El modelo es la parte de nuestro proyecto que almacena, borra, modifica y manipula el caudal principal de los datos**, apoyándose en alguna bb.dd.

De base de datos no debemos saber mucho gracias a Django 😉

En nuestro caso usaremos la base de datos integrada **SQlite** que viene integrada y simplificada.

**Proyecto vs Aplicación**

PARA RECORDAR

Ya sabemos realizar un proyecto de Django, pero Django hace una distinción entre proyecto y aplicación. Puntualmente un proyecto es todo, dentro del proyecto habrá varias aplicaciones, donde cada aplicación tendrá su función.

**Proyecto vs Aplicación**

Es fundamental tratar de crear aplicaciones dentro del proyecto para que el programa sea entendible y fácilmente manipulable.

**¿Cómo crear una aplicación?**

Crearemos la aplicación AppCoder 😎

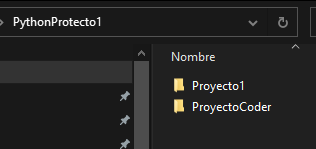
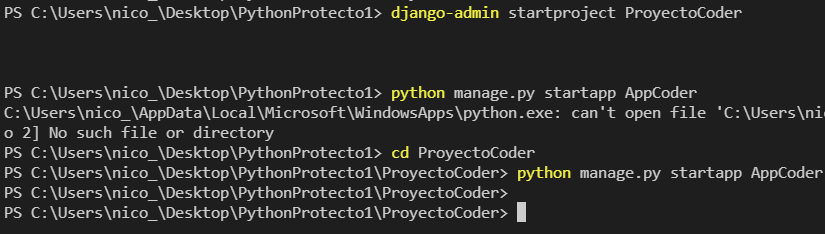
Dentro de esa aplicación, tendremos a los **estudiantes**, a los **Profesores** y por ejemplo los **Entregables**.

* **Estudiantes** (nombre, apellido, email)
* **Profesor** (nombre, apellido, email, profesión)
* **Entregable** (nombre, fechaDeEntrega,entregado)
* **Curso** (nombre, camada (“ o comisión”))

Esto último es el MODELO de nuestra app, es la información que manejará nuestro proyecto por medio de nuestra APP.

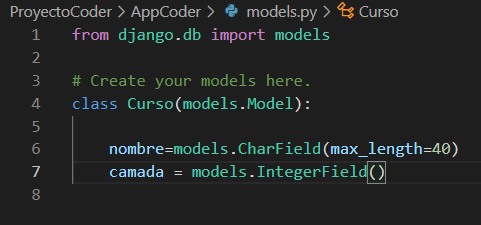
Empecemos con un proyecto nuevo que se llamará **ProyectoCoder**, ese proyecto tendrá una **AppCoder** y dentro de esa app tendremos los modelos para **Estudiantes**, **Profesor**, **Entregable** y **Curso**.

1. Crear proyecto: **django-admin startproject ProyectoCoder**
2. Entramos al proyecto con cd ProyectoCoder
3. Creamos la app: **python manage.py startapp AppCoder**

****

**Crear el modelo**

Vamos a **models.py** que se encuentra en nuestra app, **AppCoder** y creamos la estructura de nuestros modelos, por medio de clases, empezaremos por **Curso**:

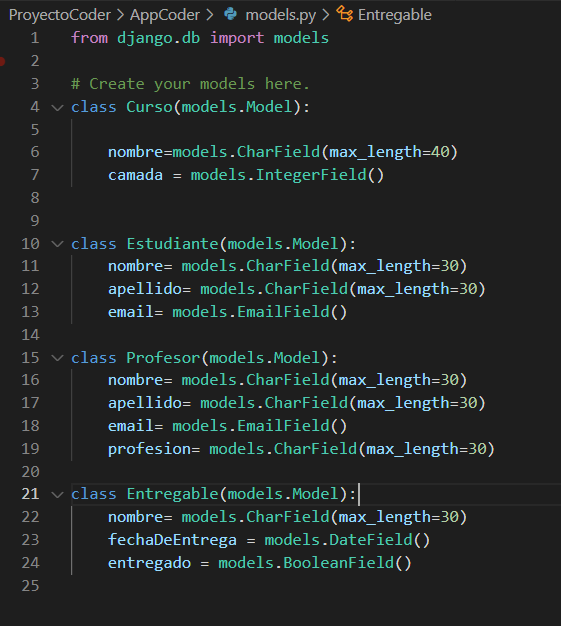
****

Sigamos con las otras:

Ya tenemos la estructura de nuestra

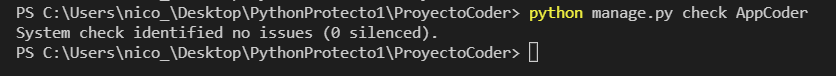
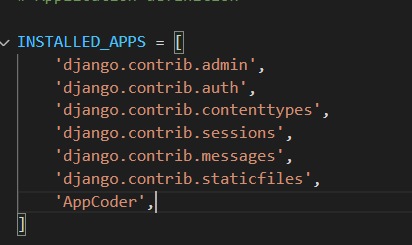
app. Entonces debemos informar a Django.

* Vamos a **settings,**
* **INSTALLED\_APPS** y
* **Agregamos nuestra APP.**

****

Es buen momento para ver si hemos realizado todo bien hasta ahora:

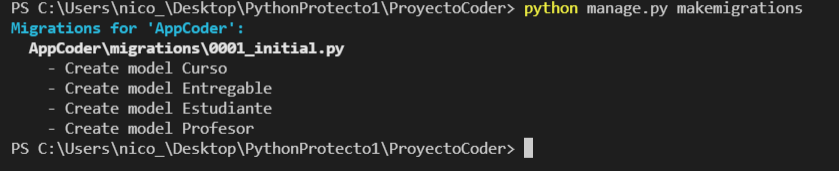
**python manage.py check AppCoder**

****

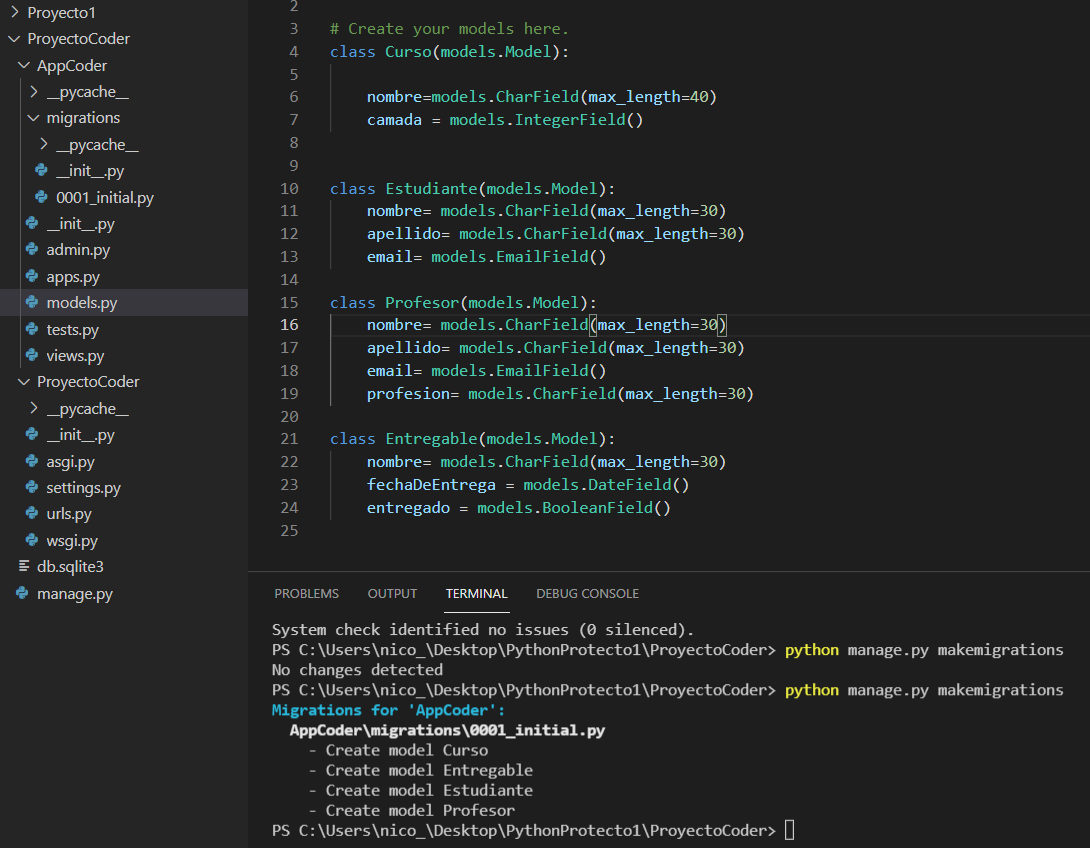
**Crear el modelo - BB.DD**

Llegó la hora de transformar nuestros modelos en base de datos:

**python manage.py makemigrations**



**¡Ya se ha creado la BB.DD!**

****

Ya tenemos la BD, pero aún está vacía, debemos generar su estructura, para eso:

1. **python manage.py sqlmigrate AppCoder 0001**

(Eso nos dará muchas líneas en código sql)

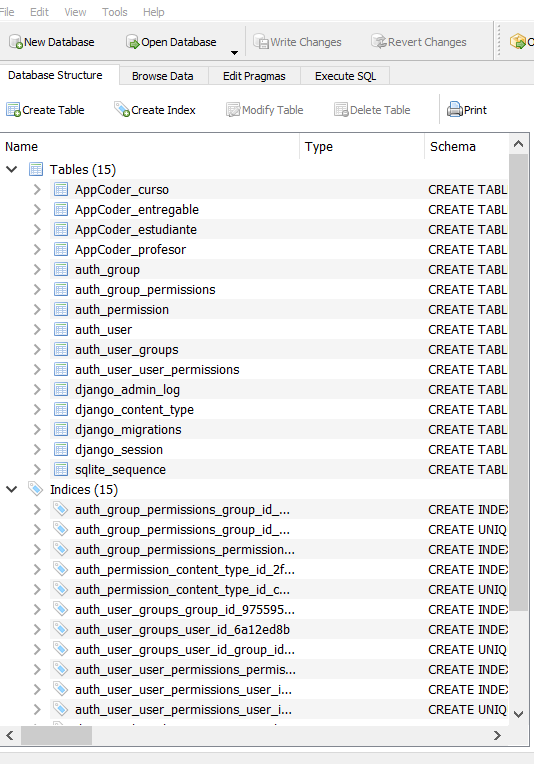
1. **python manage.py migrate**

(Con esto, esas líneas de sql impactan en nuestra base de datos)

**DR Browser**

Hasta ahora deberían confiar en nosotros, porque no tienen manera de saber si de verdad se creó la base de datos.

Para saberlo y verlo de una forma más tangible, instalaremos **DB Browser**, que nos permite acceder a la información de una base de datos **SQLite**.

****

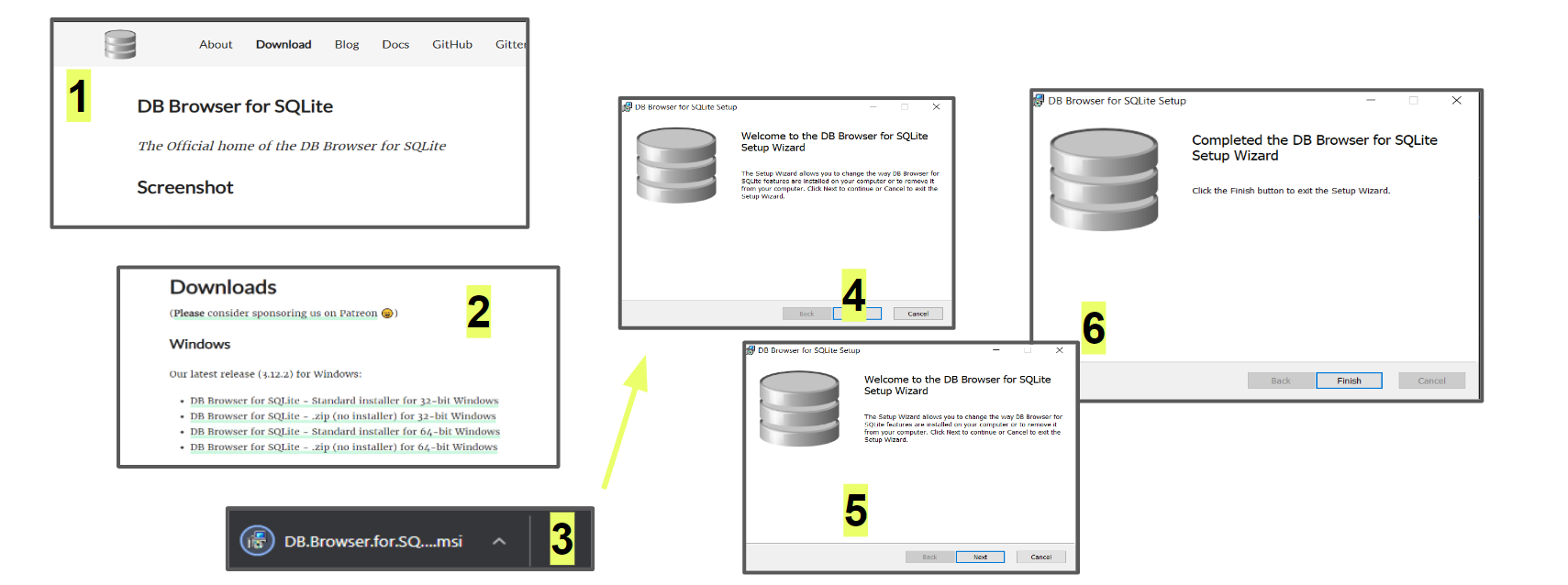
Instalarlo es muy sencillo, solo siguiente, siguiente, siguiente… etc.

**Tutorial de instalación DB Browser:**

DB Browser for SQLite | [sqlitebrowser.org](https://sqlitebrowser.org/)

Lo abrimos y vemos nuestra base de datos

**DB Browser - Instalación**



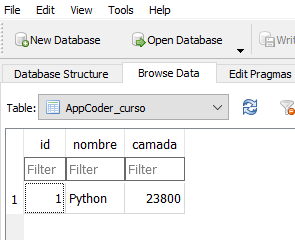
**Agregar información a mi BD**

**Agregar información a mi BB.DD**

* Agregaremos algún registro, por ejemplo agregaremos algún curso.
* Vamos a la consola y escribimos **python manage.py shell** para pararnos en la consola.
* Luego vamos a importar el modelo, en este caso la clase Curso.

- **from AppCoder.models import Curso**

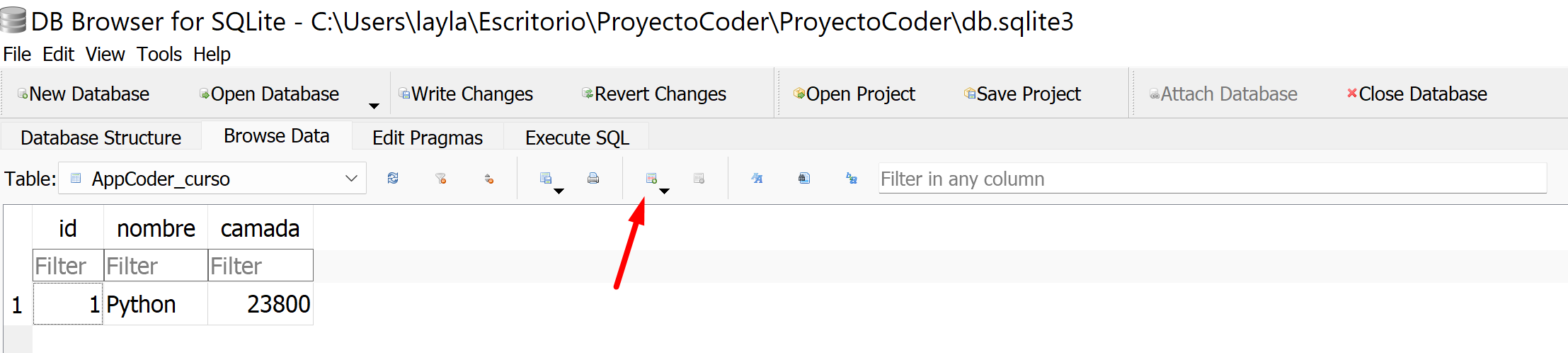
* Pasamos a crear un Curso:
* **curso = Curso(nombre="Python", camada=23800)**
* Luego enviamos ese curso a la BD: **curso.save()**

****

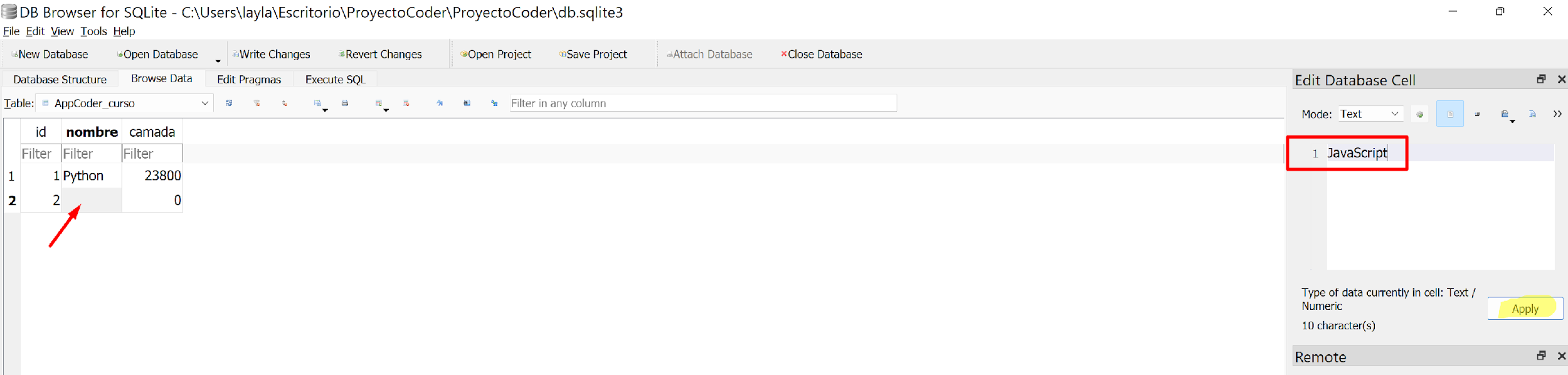
Ya probamos desde la consola que todo funciona, ahora miremos cómo quedaría una vista que guarda datos y luego los muestra en la web. Sería lo mismo, pero solo haciendo una vista, y modificando el archivo urls.py.

****

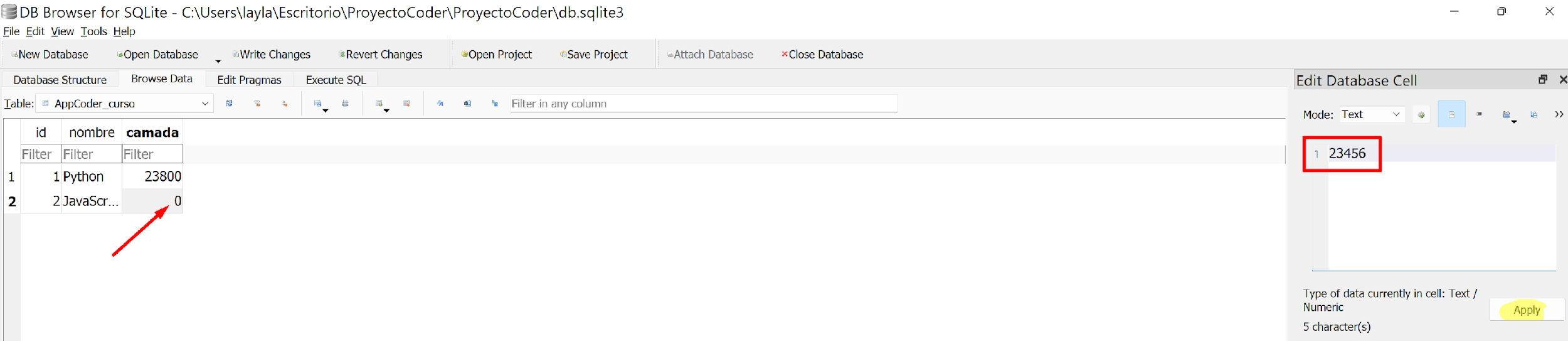
**Otra manera de agregar información a mi BD**

****

Agreguemos ahora la camada del curso de JavaScript. De la siguiente manera:



Hacemos el mismo procedimiento pero ahora con el nro de la Camada:



Hacemos persistir los cambios ☺

