Objetivos

Crear un documento donde se vea un seguimiento del curso gratuito de Udemy sobre Django, donde se recoja el proyecto, los problemas que tengamos y cómo solucionar los fallos que encontremos en el proyecto

¿ Qué es PIP ?

PIP (Pip Instalador de Paquetes o Pip Instalador de Python) es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.

¿ Qué es Python ?

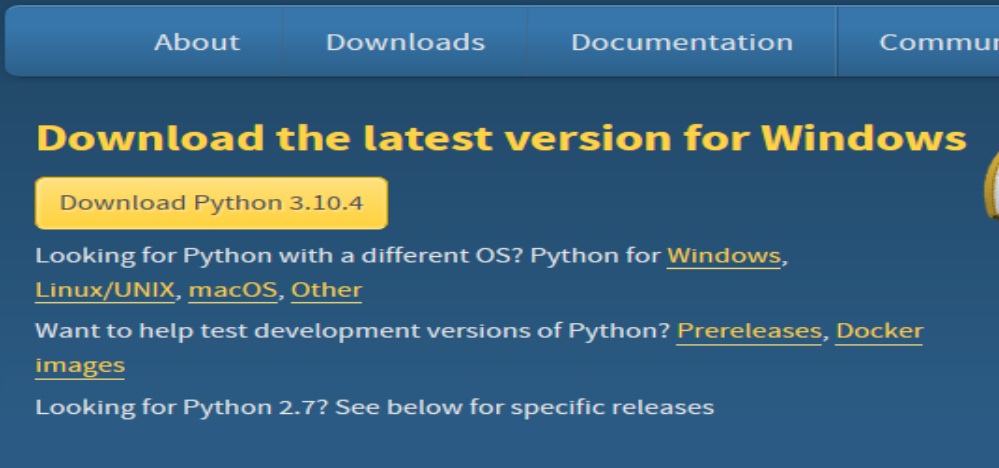
Es un lenguaje de programación interpretado, cuya filosofía hace hincapié en la legibilidad de su código y la orientación a objetos. Es un lenguaje interpretado, dinámico y multiplataforma, administrado por la Python Software Foundation.

¿ Qué es Django ?

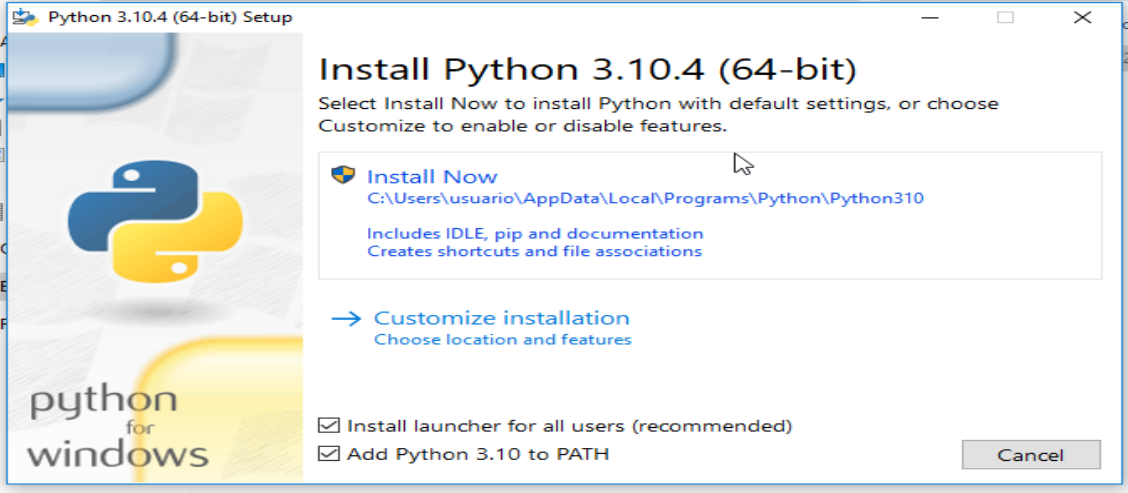
Es un framework (o entorno de trabajo) de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo–vista–controlador (MVC), un patrón de arquitectura de software, que separa los datos y principalmente lo que es la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones. La meta fundamental de Django es facilitar la creación de sitios web complejos.

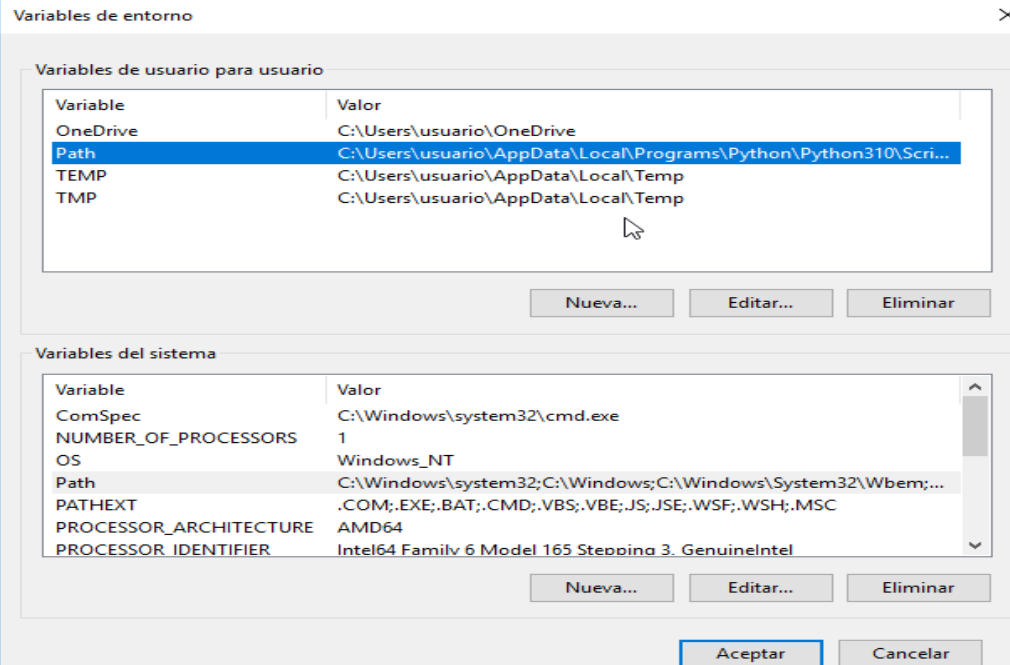
1.Instalación

Descargamos la versión más moderna de python en este caso python 3.10.4 que la podremos descargar en el siguiente enlace <https://www.python.org/downloads/>

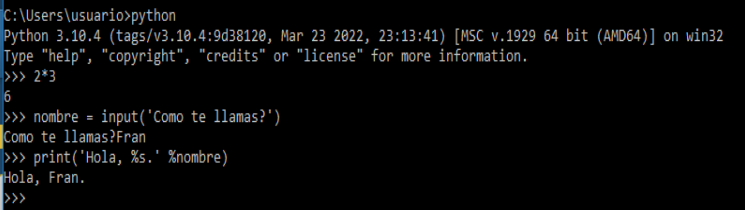


Al seleccionar la opción “Ass Python 3.10 to PATH” se nos creará la variable de entorno

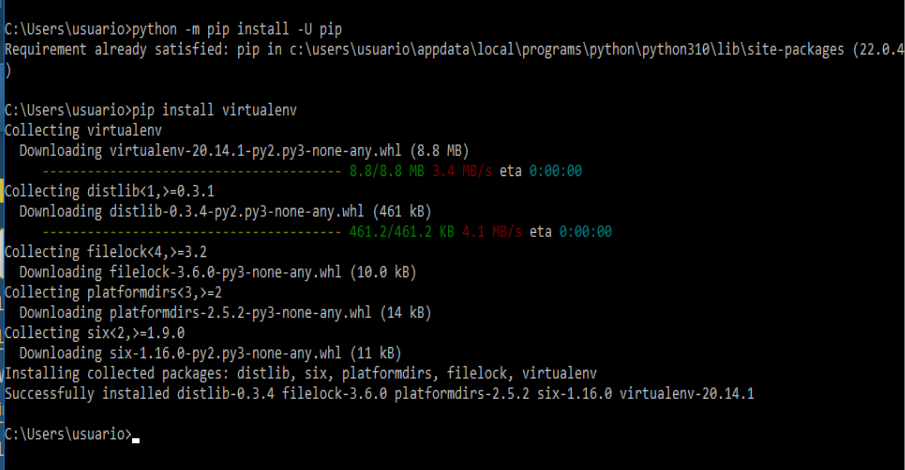




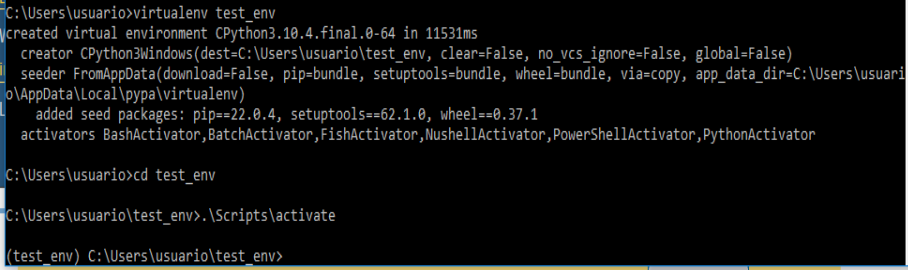
Nos vamos a símbolo del sistema y ejecutamos python para ver si la instalacion se a echo correctamente



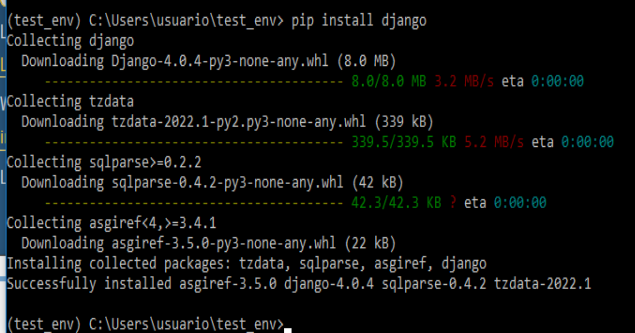
En esta versión de python ya viene instalado pip por lo que el siguiente paso es instalar Virtual Enviroment. Para ello, ejecutamos el siguiente comando “pip install virtualenv”



Instalamos Virtual Enviroment con la finalidad de poder instalar Django dentro de él, para ello creamos uno usando el comando “virtualenv test\_env”, no metemos dentro y lo activamos ejecutando “.\Scripts\activate”



Una vez hecho esto, ya podremos instalar Django, para ello utilizaremos el comando “pip install django”

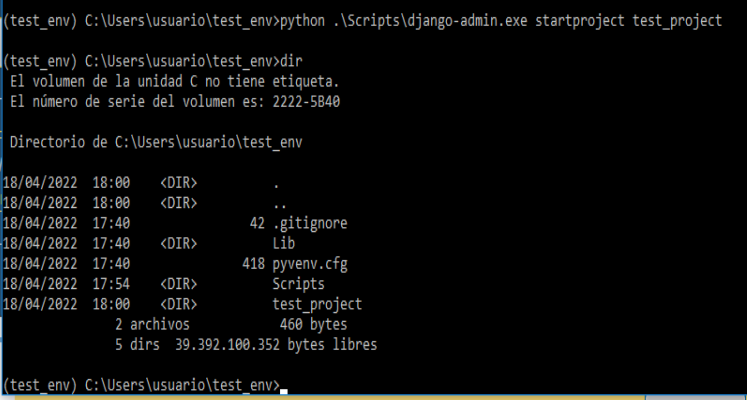


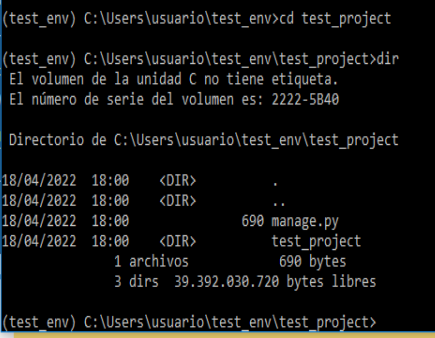
Vamos a crear un proyecto nuevo con Django, vamos a escribir lo siguiente ‘python

.\Scripts\django-admin.py startproject test\_project’. Con esto indicamos que queremos

iniciar un nuevo proyecto con su nombre. Para asegurarnos de su correcta instalación,

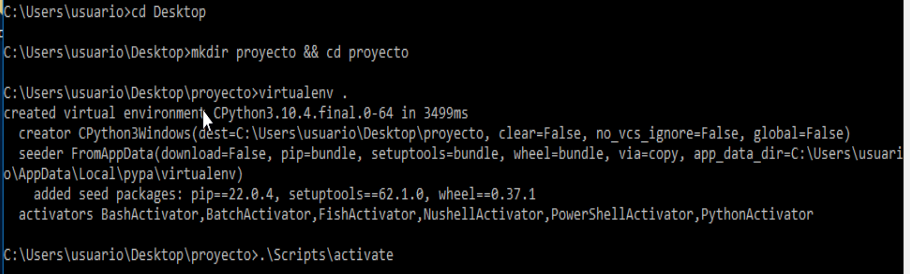
podemos hacer un listado del contenido de nuestro Virtual Enviroment.



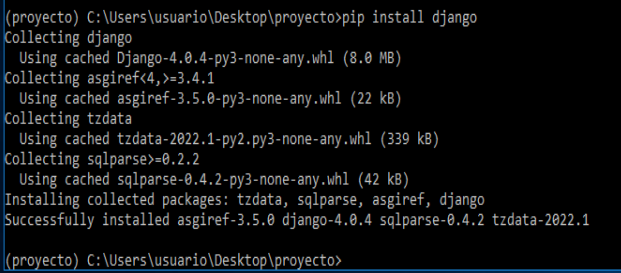


2. Empezar proyecto nuevo

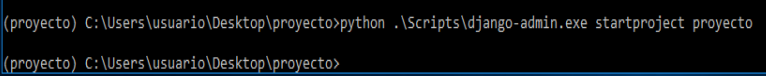
Para empezar un proyecto nuevo, abrimos nuestro cmd y creamos un nuevo directorio (en mi caso voy a crear una carpeta llamada “proyecto” en el escritorio). Entramos en ella e instalamos el Virtual Enviroment. Después, lo activamos como hicimos en el punto anterior.



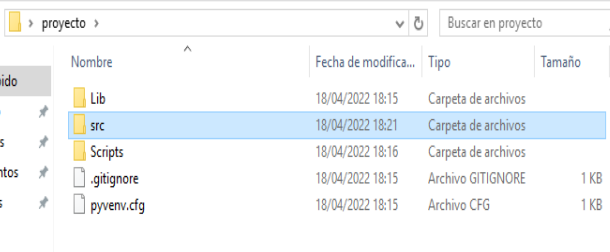
A continuación instalaremos Django, utilizamos el comando “pip install django”



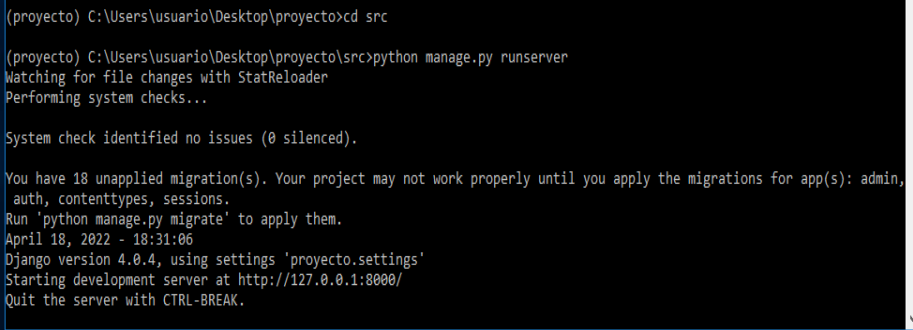
Creamos el nuevo proyecto para ello usamos “.\Scripts\django-admin.py startproject proyecto”



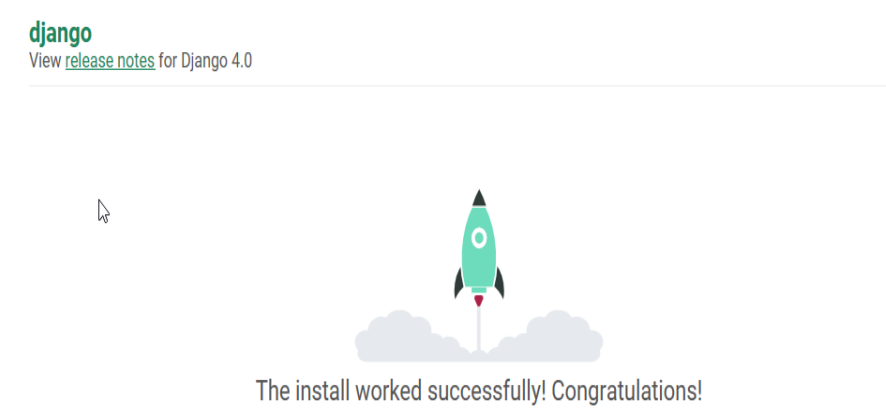
Una vez creado el proyecto nos iremos a Escritorio y dentro de la carpeta proyecto cambiaremos el nombre de la carpeta proyecto por src para no crear confusiones.



Cambiamos al directorio src y ejecutamos el servidor de desarrollo de nuestro entorno de desarrollo, para ello utilizamos el comando “python manage.py runserver”



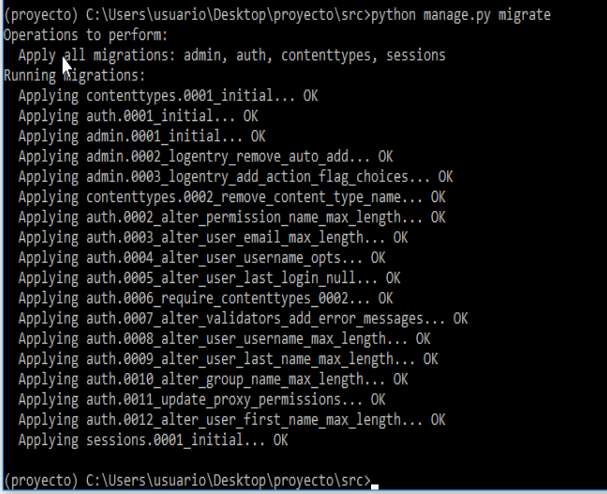
A continuación en el navegador ponemos “http://127.0.0.1:8000”, y podemos comprobar que funciona correctamente.



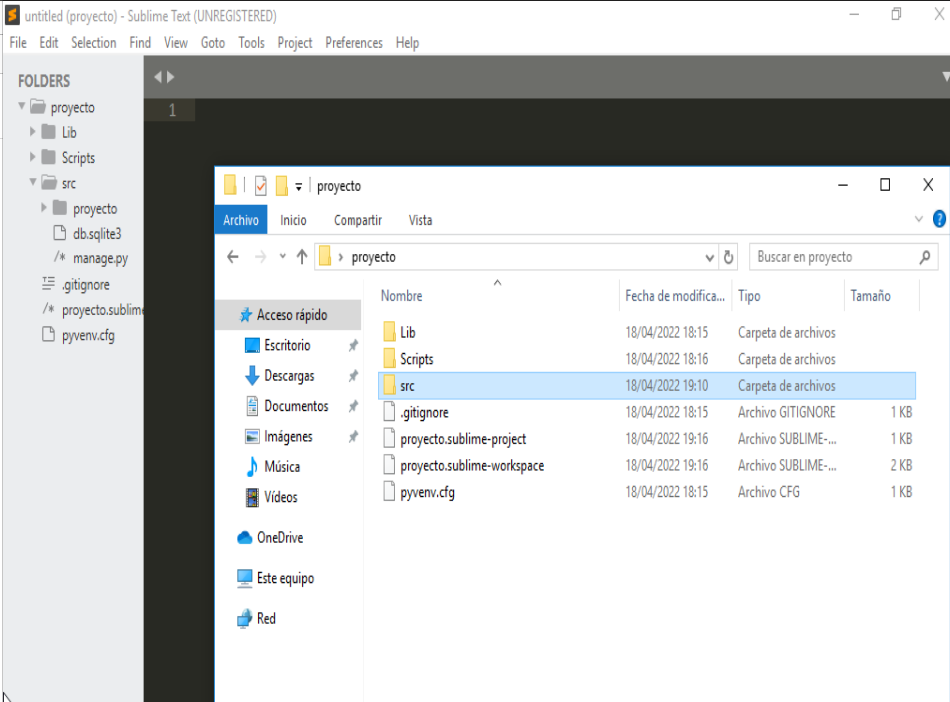
3. Primera migración

Ya tenemos nuestro proyecto funcionando. Ahora podemos hacer las migraciones. Las migraciones nos sirven para comunicarnos con nuestra base de datos, es el vínculo entre la base de datos y nuestro proyecto.

En nuestro cmd, cerramos el servidor (Ctrl + D) y escribimos ‘python manage.py migrate’.

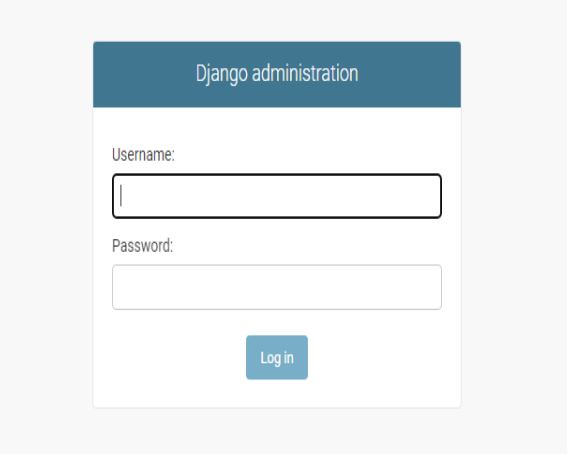


A continuación, abrimos nuestro editor de texto (en mi caso Sublime Text) y vamos a trabajar con nuestro proyecto. Pinchamos en ‘Project’ y añadimos la carpeta del proyecto ‘proyecto’. Una vez abierto, lo guardamos dentro de nuestra carpeta ‘raíz’.

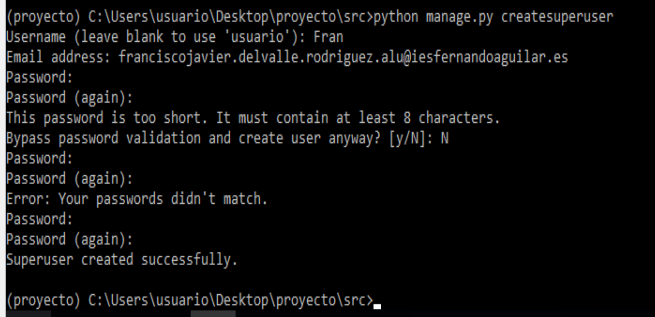


4. Creación de Superusuarios y Administración de Django.

Los proyectos de Django vienen automáticamente con una interfaz ya escrita. Volvemos a activar nuestro servidor y recargamos nuestro navegador web.

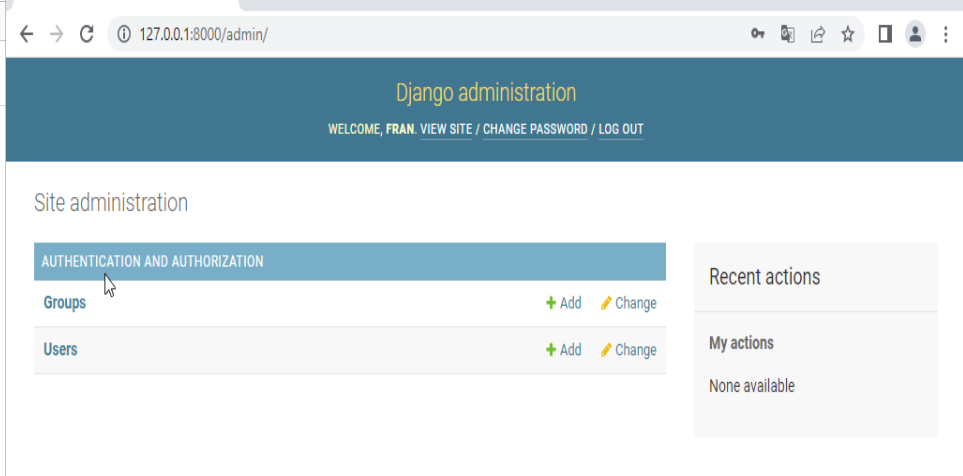


Para poder iniciar sesión dentro de Django, cerramos el servidor y vamos a crear un superusuario, capaz de entrar. Para ello, ponemos en nuestro cmd ‘python manage.py createsuperuser’ y añadimos nuestro usuario, email y contraseña.

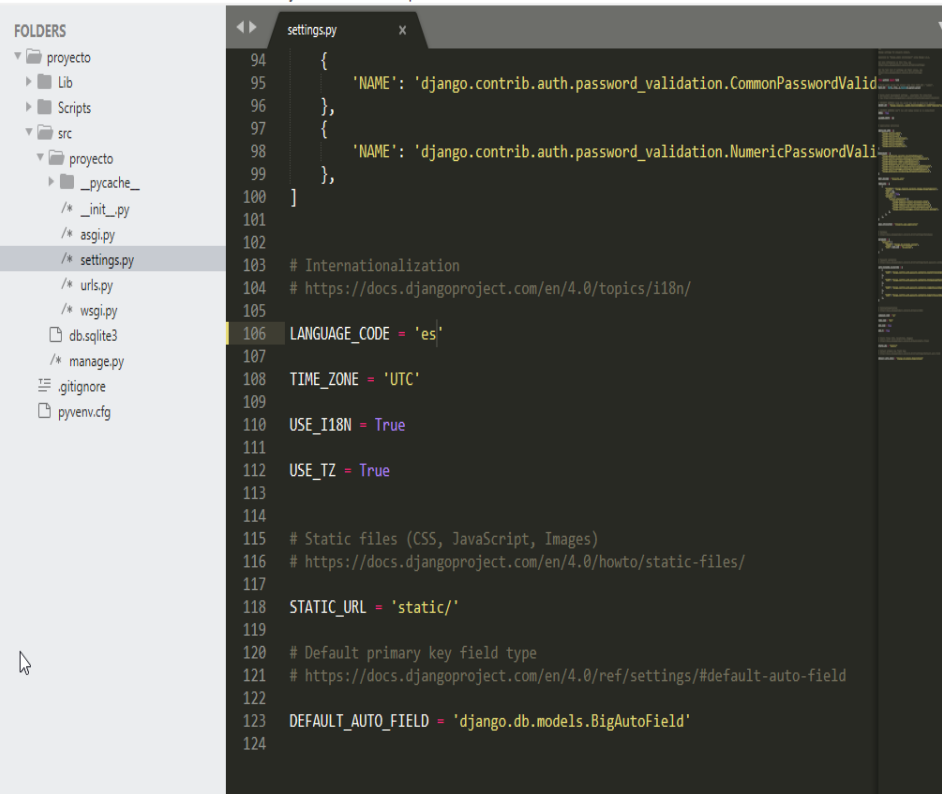


Volvemos a iniciar el servidor y ya podemos acceder a nuestra página. Esta es la página

principal de la interfaz administrativa.



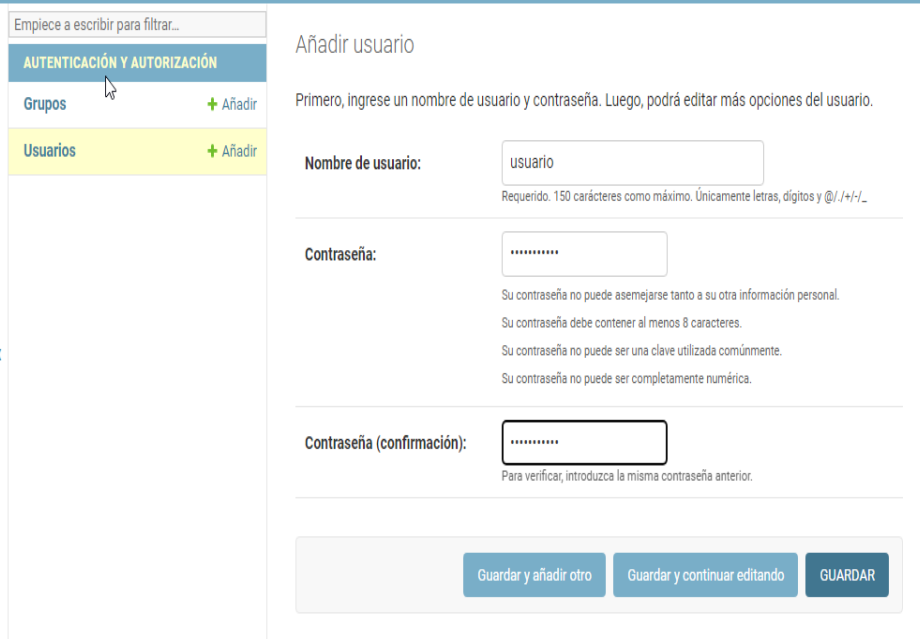
Cambiamos el idioma para ello en el fichero settings.py modificaremos la línea que ponga “LANGUAGE\_CODE = ‘en-us’ por LANGUAGE\_CODE= ‘es''' y recargamos la página.



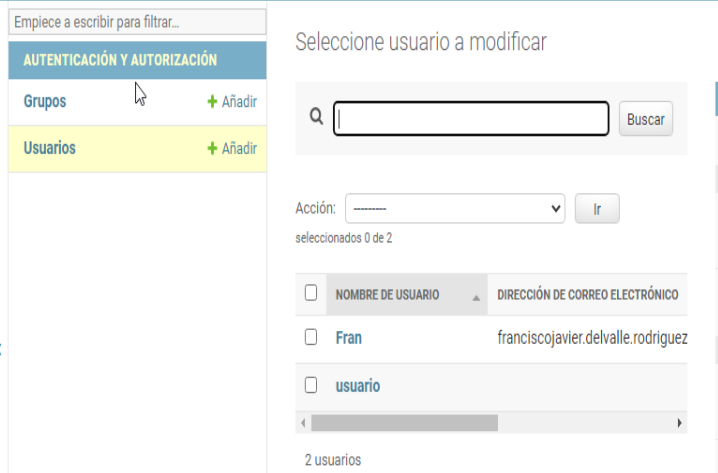


Si, por ejemplo, queremos crear otro usuario, podemos entrar en la sección de ‘Usuarios’

y ‘Añadir usuario’

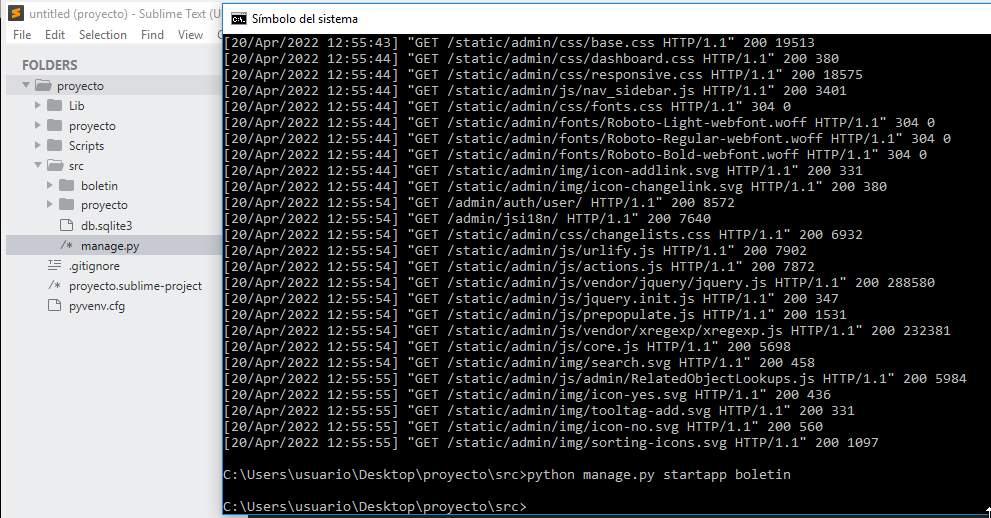


Si volvemos a ‘Usuarios’ podremos ver que ya tenemos el usuario creado. El cual no tendrá todos los permisos como nuestro superusuario.

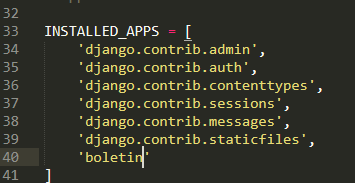


5. Primera aplicación

Para crear una aplicación tendremos que cerrar el servidor y ejecutamos el siguiente comando “python manage.py startapp boletin”, y vemos que se nos a creado la aplicación

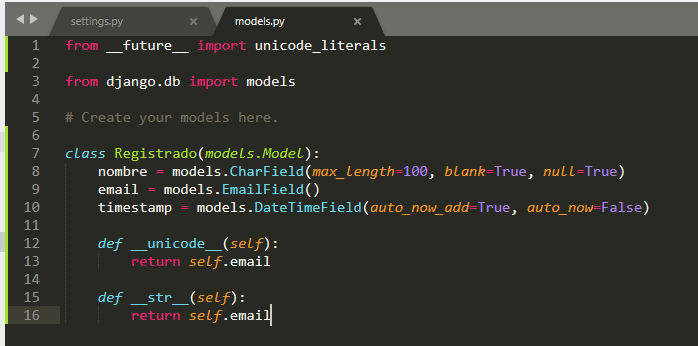


A continuación la registramos la apliación en el fichero setting.py



6. Primer modelo

Vamos a crear un registro de la gente que va a recibir nuestro boletin. Para ello escribimos dentro de ‘models.py’ lo siguiente:



Por último, realizamos la migración, para ello escribimos el siguiente comando “python manage.py makemigrations”, que buscara todas las modificaciones que hemos hecho en nuestro modelo y va a empaquetarlas. Despues, con el comando “python manage.py migrate” se comunica con nuestra base de datos

