**Teoria Django**

Santiago Alejandro Gonzalez

26 de febrero de 2021

Zinobe

Tabla de contenido

[Django en Python 1](#_Toc65409587)

[¿Qué es Django? 1](#_Toc65409588)

[¿Qué es un FrameWork? 1](#_Toc65409589)

[¿Para qué sirve Django? 1](#_Toc65409590)

[MVC (Modelo Vista Controlador) 1](#_Toc65409591)

[Modelo: 1](#_Toc65409592)

[Vista: 1](#_Toc65409593)

[Controlador: 2](#_Toc65409594)

[Ventajas del MVC 2](#_Toc65409595)

[MVC (Modelo Vista Contrlador) –> MVC (Model Template View) 3](#_Toc65409596)

[Instalación local vs instalación virtual 3](#_Toc65409597)

[Django y BBDD 4](#_Toc65409598)

[Proyectos en Django 5](#_Toc65409599)

[Creación de proyecto 5](#_Toc65409600)

[Archivo manage.py 5](#_Toc65409601)

[Manage.py help 5](#_Toc65409602)

[Carpeta con el nombre del proyecto 6](#_Toc65409603)

[Archivo \_\_init\_\_.py 6](#_Toc65409604)

[Setting.py 6](#_Toc65409605)

[Urls.py 7](#_Toc65409606)

[Wsgi.py 7](#_Toc65409607)

[Asgi.py 7](#_Toc65409608)

[Activación del proyecto 8](#_Toc65409609)

[Servidores 9](#_Toc65409610)

[Aplicaciones en Django 10](#_Toc65409611)

[Ventajas 10](#_Toc65409612)

[Crear aplicación con Django 10](#_Toc65409613)

[Verificar si la aplicación esta bien está bien 11](#_Toc65409614)

[Archivo models.py 11](#_Toc65409615)

# Django en Python

## ¿Qué es Django?

Django es un Framework web gratuito y de código abierto escrito en Python

## ¿Qué es un FrameWork?

Traduce entorno de trabajo, es un ecosistema conformado por un conjunto de herramientas, librerías y buenas prácticas para crear aplicaciones, en el caso de django aplicaciones web.

## ¿Para qué sirve Django?

Django permite:

* Crear sitios web complejos y sencillos de manera rápida y sencilla
* Hay tareas que son repetitivas, pesadas y comunes en el momento de crear diferentes sitios web. Django viene a facilitar la realización de estas tareas
* Hay código que podemos reutilizar de un sitio web a otro, Django también nos permite esta reutilización de forma sencilla.

## MVC (Modelo Vista Controlador)

### Modelo:

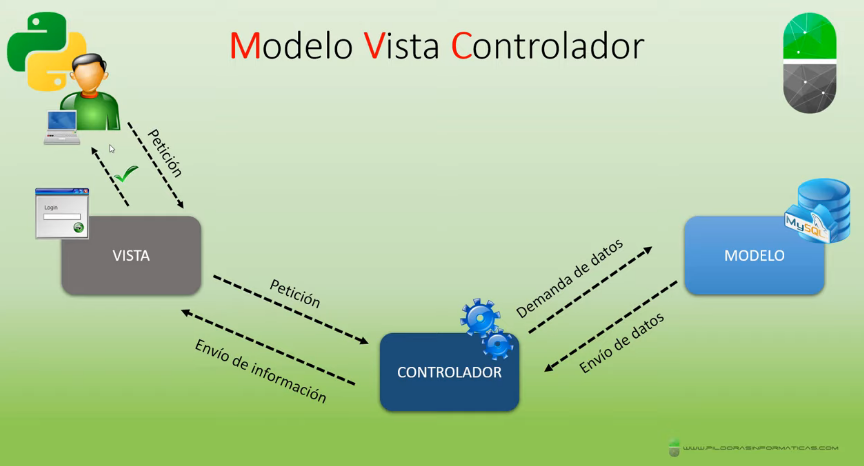
Gestiona los datos, obtiene los datos de la base de datos.

### Vista:

Se encarga de mostrar la información al usuario, lo que el usuario ve, con lo que el usuario interactúa

### Controlador:

Modulo encargado de gestionar todas las comunicaciones existentes entre la vista y el modelo.



## Ventajas del MVC

* Funcional
* Mantenibles
* Escalable: si en el futuro necesito agregar más funciones podre hacerlo de manera sencilla

## MVC (Modelo Vista Contrlador) –> MVC (Model Template View)

Django no hace uso del patrón modelo, vista, controlador si no del patrón model, template, view.

Django cambia:

* Vista –> Template
* Modelo -> Model
* Controlador -> View

Aunque se haya cambiado los nombres la filosofía sigue siendo la misma.

## Instalación local vs instalación virtual



## 

Cabe destacar que cuando estamos desarrollando en un ambiente más profesional, el desarrollo del software se da en tres etapas, la primera el desarrollo, luego las pruebas y luego la producción, así que cuando estamos en la etapa de desarrollo puede que estemos manejando una versión x de Django y de Python y al momento de la producción cambien o se actualicen, pues gracias a la instalación virtual que nos permite desarrollar en varias versiones de Python y Django podemos pasar a la etapa de producción con la misma versión con la que pasamos la etapa de desarrollo y de esta manera no hayan incompatibilidades por el cambio de versiones.

## Django y BBDD

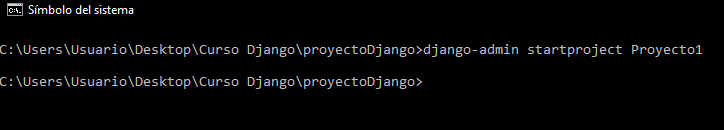


# Proyectos en Django

## Creación de proyecto

Vaya al cmd y ubíquese con cd en la carpeta en la que quiere crear el proyecto, luego lleve a cabo el siguiente comando “django-admin startproject nombredeproyecto”

Ejamplo:



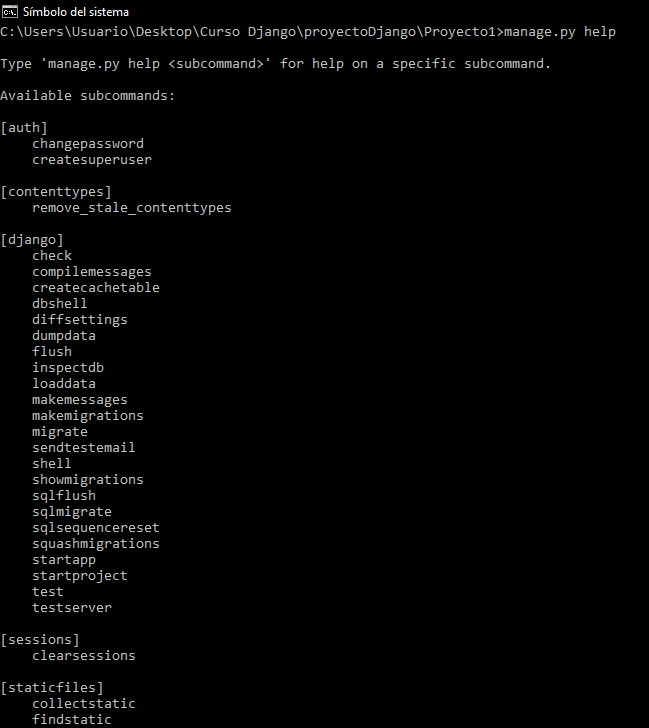
Luego de esto se le creara una carpeta con el nombre del proyecto y dentro de esta carpeta abra una carpeta más con el nombre del proyecto y un archivo llamado “manage.py”

## Archivo manage.py

Este archivo es importantísimo porque es una utilidad de líneas de comando que nos permite interactuar con los proyectos django de varias formas.

### Manage.py help

Si nos ubicamos en la dirección del archivo “manage.py” desde la consola de CMD y ejecutamos el comando “manage.py help” encontraremos todas las instrucciones y todo lo que puede el archivo “manage.py”



## Carpeta con el nombre del proyecto

La carpeta que tiene el mismo nombre del proyecto contendrá los siguientes archivos:

### Archivo \_\_init\_\_.py

Es un archivo necesario para iniciar paquetes en Python, le indica al intérprete de Python que el directorio contiene un módulo, y que debe tratarlo como tal (es decir, hacer que sea posible importar los archivos como parte del módulo).

### Setting.py

Contiene todas las configuraciones del proyecto django. En este archivo en la lista de TEMPLATES en su parte DIR[], dentro de este debe ir la dirección de ubicación de la carpeta que posee los templates o plantillas HTML.

Por otro lado, cunando creamos una aplicación debemos registrar también en el archivo settings, es por eso que cuando se crea una app se debe registrar en la lista INSTALLED\_APPS, simplemente se agrega el nombre de la aplicación y ya es decir “gestionPedidos ”,

### Urls.py

Archivo en el que se almacenan las urls que va a utilizar nuestro proyecto

### Wsgi.py

El concepto clave de la implementación con WSGI es el application invocable que utiliza el servidor de aplicaciones para comunicarse con su código. Se proporciona comúnmente como un objeto nombrado applicationen un módulo de Python accesible al servidor.

Es utilizado tanto por el servidor de desarrollo de Django como en las implementaciones

de WSGI de producción.

WSGI es la interfaz de puerta de enlace del servidor web. Es una especificación que describe cómo un servidor web se comunica con las aplicaciones web y cómo las aplicaciones web se pueden encadenar para procesar una solicitud.

### Asgi.py

Además de WSGI, Django también admite la implementación en ASGI , el estándar emergente de Python para aplicaciones y servidores web asíncronos.

Al igual que WSGI, ASGI le pide que proporcione un applicationinvocable que el servidor de aplicaciones utiliza para comunicarse con su código. Se proporciona comúnmente como un objeto nombrado applicationen un módulo de Python accesible al servidor.

ASGI ( Interfaz de puerta de enlace de servidor asíncrono ) es un sucesor espiritual de WSGI, destinado a proporcionar una interfaz estándar entre servidores web, marcos y aplicaciones de Python con capacidad asíncrona.

## Activación del proyecto

Para activar el proycto es necesario ubicarse en la ubicación del archivo “manage.py” desde la consola CMD y ejecutar el siguiente comando “Python manage.py migrate”



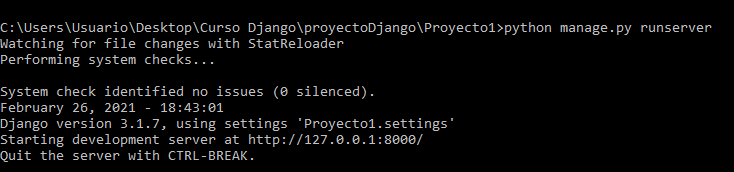
Eso nos creara un archivo sqlite3 en la carpeta del proyecto. Este archivo pertenece a la base de datos.

## Servidores

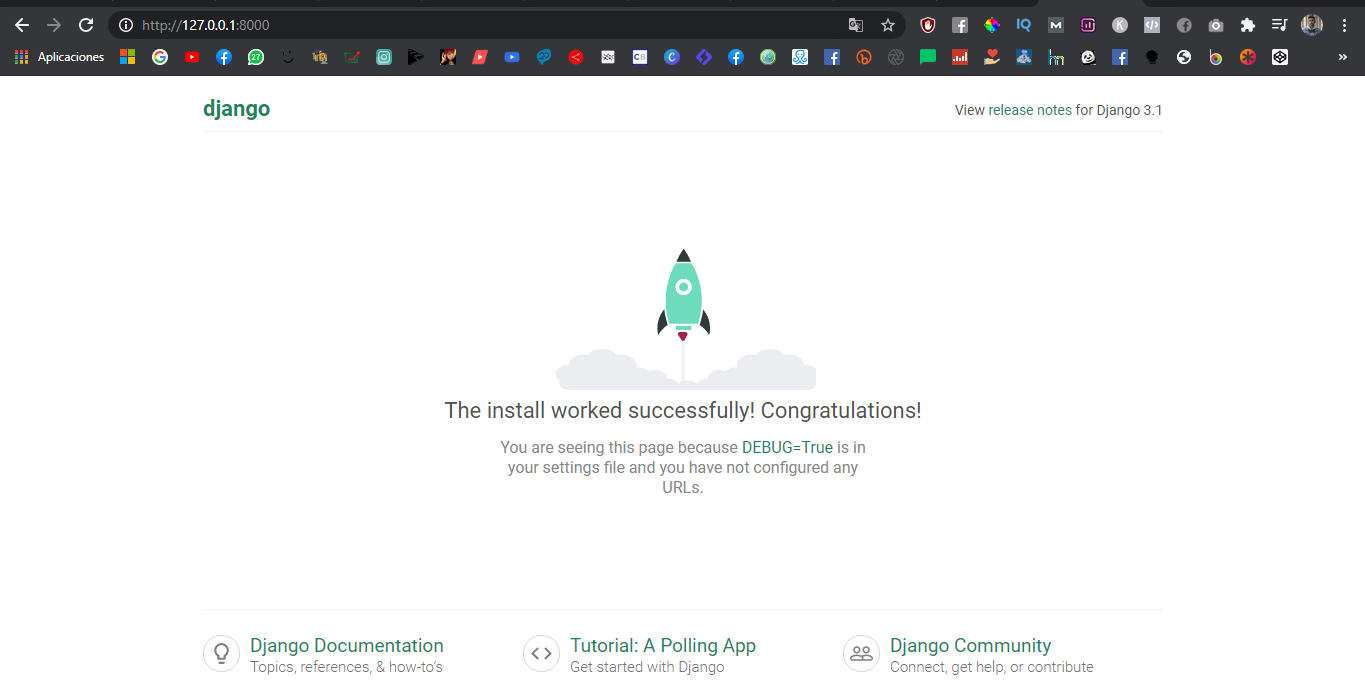
Para poner en ejecución el proyecto es necesario ejecutar un servidor para poder ir a la dirección de la pagina y validar la visualización de nuestro proyecto.

Ahora bien, Django viene con un servidor para hacer pruebas muy ligero, pero que no es recomendable para proyectos serios, porque es un servidor que admite multiples peticiones simultaneas, no admite cargas de trabajo pesadas.

Para ejecutar el proyecto simplemente llevamos a cabo el siguiente comando “Python manage.py runserver”



Si nos dirigimos a la dirección dada, obtendremos lo siguiente:



# Aplicaciones en Django

Es exponer la diferencia entre aplicación y proyecto para django, un proyecto como bien lo vimos es simplemente todo el software y las aplicaciones vendrían siendo los módulos que posee dicho proyecto.

Ejemplo si tenemos un proyecto de tienda virtual, podemos decir que las aplicaciones de este son, Stock de Almacen, Ventas, Pagos, Envios, etc. En realidad estas aplicaciones o modulos dependen del programdor, es decir, el desarrollador es quien define las aplicaciones que necesita.

Cabe destacar que pueden haber dos proyectos iguales con las mismas funcionalidades, pero con aplicaciones diferentes, puede que una tenga solo 10 aplicaciones y otra tenga 20 y ambos proyectos hacen el mismo funcionamiento, es por eso que la creación de aplicaciones depende del desarrollador, esto es lo que denominamos modularización (reutilización)

## Ventajas

La ventaja de desarrollar los proyectos con aplicaciones es que permite que el código este mejor distribuido, más organizado, sea más entendible y pueda ser reutilizado por otros proyectos

## Crear aplicación con Django

En la consola cmd nos ubicamos en la dirección del proyecto, específicamente en donde se encuentra el archivo manage.py y ejecutamos el siguiente código: “Python manage.py startapp nombre\_de\_la\_aplicacion”



Al llevar a cabo este código se creará una carpeta con el nombre de la aplicación y esta carpeta contendrá otra carpeta y otros archivos que usaremos para el desarrollo de la aplicación.

### Verificar si la aplicación esta bien está bien

Ejecutar el siguiente código “Python manage.py check nombredeaplicaciòn”



Lo anterior indica que no hay ningún error

### Archivo models.py

En este archivo se creará el modelo de base de datos, cada clase que crees será como una tabla de la base de datos.

# Aclaración BBDD

Es importante aclarar que los siguientes comandos vistos a continuación sirven para cualquier motor de base de datos, es decir, la manera en que se insertan datos, se consultan, se actualizan o se eliminan, son siempre igual independientemente del motor de búsqueda, solo que a continuación divido los procedimientos de acuerdo a dos bases de datos diferentes.

Además, es importante resaltar que cada cambio que se haga en algún modelo (Tabla), ya sea atributos, funciones lo que sea, se debe realizar las respectivas migraciones en consola con los siguientes comandos:

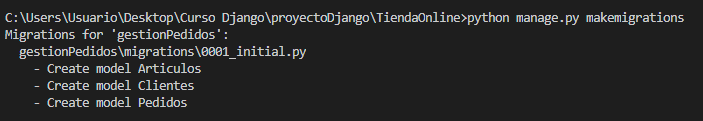
* “Python manage.py makemigrations”
* “Python manage.py migrate”

# Bases de datos SQLite3

Para crear una base de datos en django todo se hace desde el archivo models.py, en donde cada clase representa una tabla de la base de datos.

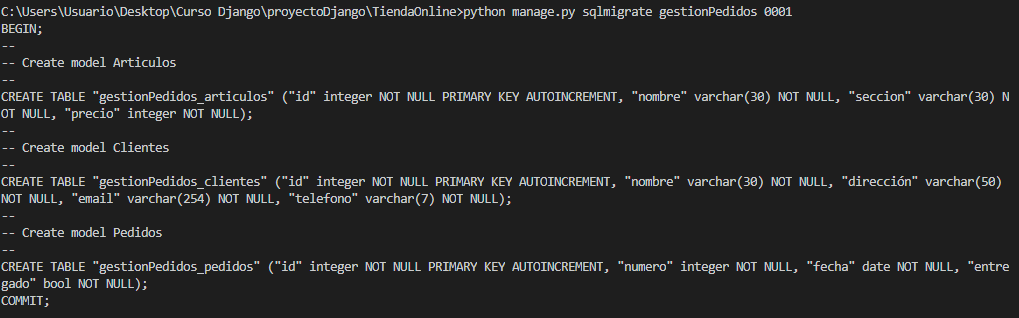
## Creación base de datos

Una vez creada el modelo de base de datos, es necesario crear como tal el archivo sqlite de la base de datos, eso lo hacemos con el siguiente comando “Python manage.py makemigration”

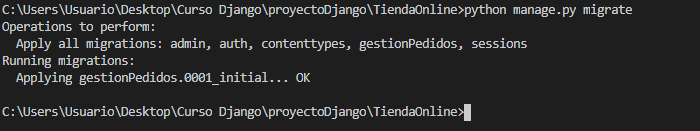


Esto creara la base de datos, pero no se han creado aun las tablas por lo que debemos llevar a cabo la ejecución de un código para que genere ese código SQL, el código es “Python manage.py sqlmigrate nombre\_de\_la\_aplicaciòn numero\_de\_la\_migraciòn”

El número de la aplicación se encuentra en la ejecución del anterior código, es el número que esta junto a \_initial.py, en este caso es 0001.

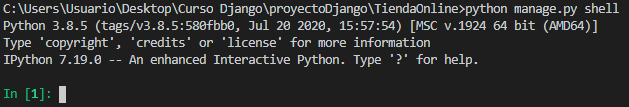


Luego de que se han generado en código sql de las tablas, necesitamos insertar esas tablas en nuestra base de datos, esto se hace con el siguiente código. “Python manage.py migrate”



## Inserción de datos desde consola

Para poder manipular la base de datos desde consola, lo primero que debemos hacer es abrir la consola, por lo que insertamos el siguiente código “python manage.py shell”



Antes de hacer la debida inserción de una dato debemos importar el modelo o la tabla de la cual queremos insertar datos, para eso llevamos a cabo el siguiente código dentro del Shell “from nombre\_de\_la\_aplicación.models import nombre\_modelo\_o\_tabla”



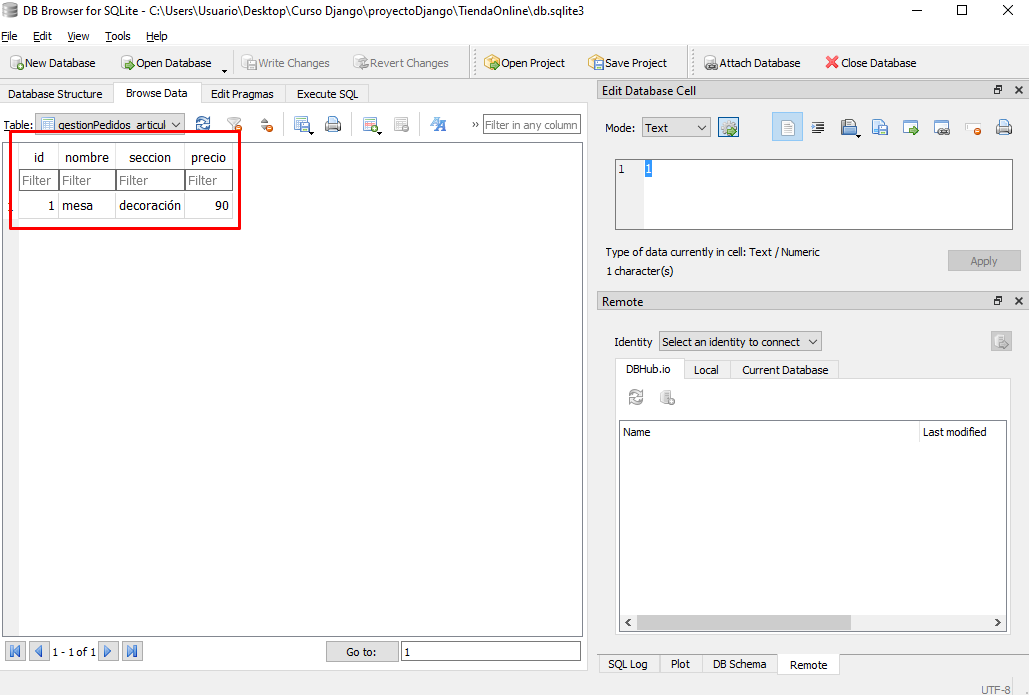
Ahora bien, para hacer la inserción simplemente creamos una variable y la igualamos al modelo entre paréntesis y el nombre de los atributos de la tabla igualados a su correspondiente valor, ejemplo:



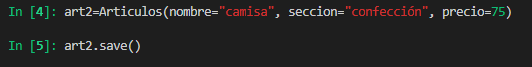
Luego simplemente guardamos para que la inserción se refleje en la tabla.



Ahora si observamos en la base de datos, debe haberse creado la fila.



Inserción de otro elemento:



Otra forma más simplificada de insertar datos:



Otro artículo:



## Actualizar registro desde consola

Si debemos cambiar o actualizar uno de los campos de alguno de los registros, hacemos lo siguiente.

Debemos escoger la variable con la que insertamos el dato y hacemos lo siguiente:



## Borrar registro desde consola

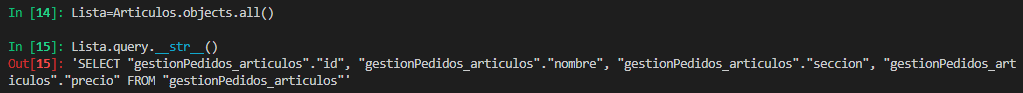
Si queremos borrar un dato de una tabla hacemos lo siguiente, creamos una variable la igualamos al modelo o nombre de la tabla.objects.get() y dentro de los paréntesis colocamos el criterio por el que vamos a eliminar el dato, es decir, vamos a borrar el datos por su id o por alguno de sus otros atributos, luego simplemente damos enter y volvemos a escribir la variable.delete() y ya se habrá eliminado. cabe destacar que si alguno otro dato posee el mismo criterio de eliminación que estamos dantos, también se eliminara. Ejemplo:





## Select desde consola

Para listar todos los elementos de una tabla, de nuevo es necesario crear una variable, que va a hacer una lista y se iguala a lo siguiente… lista=nombre\_tabla.objects.all(). Ejemplo:

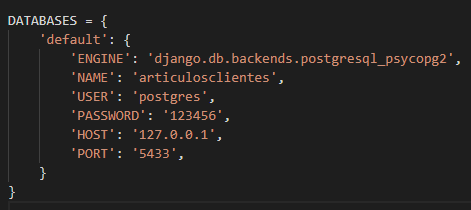


# Base de datos PostgreSQL

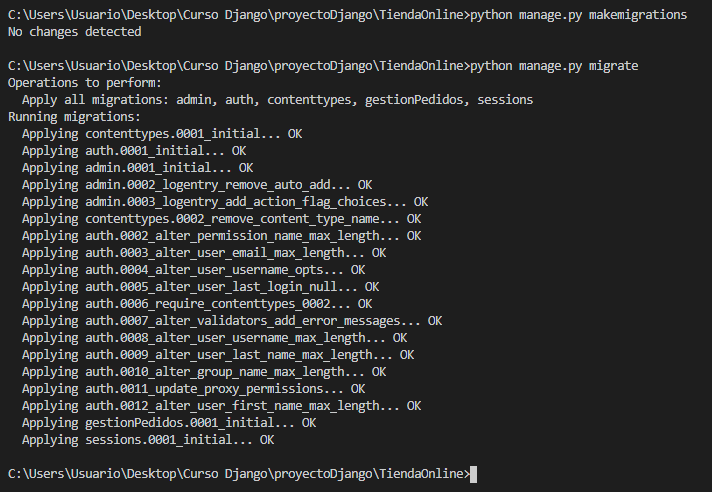
Para conectar a django con el motor de base de datos de PostgreeSQL, es necesario instalar una librería, dicha librería es la siguiente.



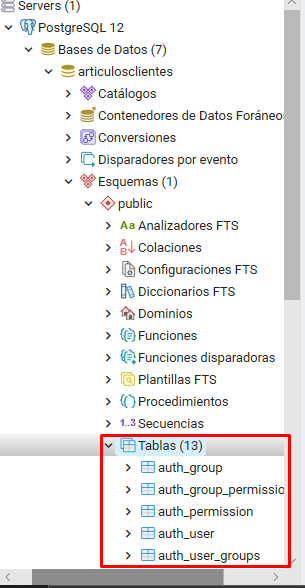
Luego de instalarla debemos configurar el archivo settings en la lista DATABASES, de la siguiente manera:



Lo siguiente que debemos hacer son las correspondientes migraciones. Ejecutamos los siguientes dos códigos que ya conocemos, “Python manage.py makemigrations” y luego “Python manage.py migrate”. Si no hay ningún problema les debe salir lo siguiente, pero si les sale un error deben verificar los valores que asignaron a la lista DATABASES del archivo settings.py

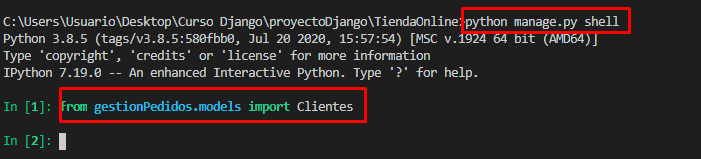


De esta manera, podrás observar que se han creado las diferentes tablas en la base de datos de PostgreSQL.

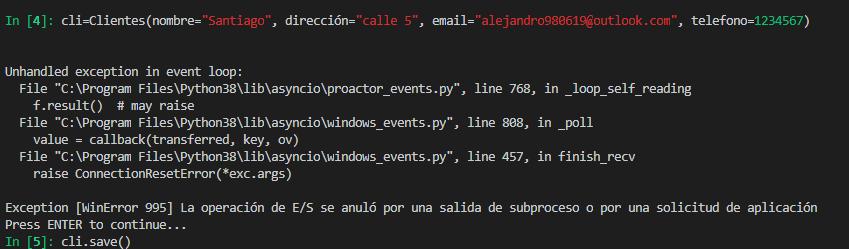


## Insertar registro PostgreSQL

Para insertar un registro lo primero que hacemos es abrir el Shell, como ya sabemos “Python manage.py shell” y luego importamos el modelo con “from gestionPedidos.models import Articulos”



Ahora bien, para insertar dato lo hacemos de la misma manera en que lo hacíamos cuando usábamos sqlite, asignamos una variable con el nombre de la tabla y en paréntesis el nombre de las atributos igualados por lo que queremos asignar en cada campo. Los campos deben estar separados por comas, y luego simplemente volvemos a colocar el nombre de la variable y. save(). Ejemplo



Validando en PostgreSQL



## Consultas con Where

### Consulta sencilla un solo criterio:

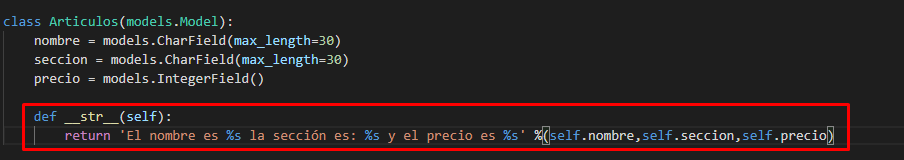
A continuación vamos a hacer una consulta que me traiga los artículos solo de una sección en específico, para eso ejecutamos el siguiente comando.

“nombre\_de\_tabla.objects.filter(nombre\_de\_campo=”valor\_de\_busqueda”)”

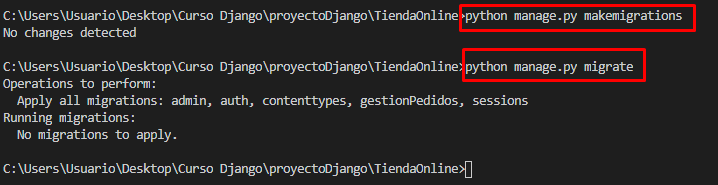


Como observamos la respuesta no es exactamente la que esperábamos , nos ha devuelto un Queryset con dos datos que correspondes a los artículos de la sección de deportes.

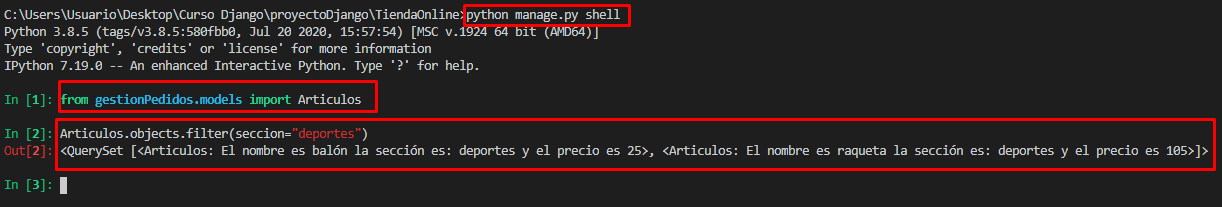
Para que podamos ver la información de manera correcta debemos agregar una función str dentro de la clase de la tabla o del modelo. Hacemos lo siguiente:



Luego de nuevo volvemos a realizar el makemigrations y el migrate, para guardar cambios del modelo.



Y de nuevo entramos al Shell y de nuevo importamos la tabla o modelo y hacemos la consulta y nos deberá salir algo como esto:



### Consulta con dos criterios:

Vamos a hacer una consulta en la que me traiga el nombre=”mesa” y que sea de la sección=”decoracion”.



### Consultas de “>” y “<”:

A continuación haremos consultas jugando con el precio de los artículos, de sin “>” o ”<” que cierto valor.

Para escribir “>” en el Shell se debe hacer de la siguiente manera \_\_gte =, lo cual daría el siguiente código:



Ahora bien, para el operador “<”, el código es: “\_\_lte=”

### Consultas de rango, entre dos valores:

Utilizamos el comando “\_\_range(“valor rango uno”,”valor rango dos”)”. Ejemplo:



### Ordenar registros

Vamos a realizar una consulta que me traiga todos los artículos mayores a 30 pero que este ordenado de manera ascendente según el precio.



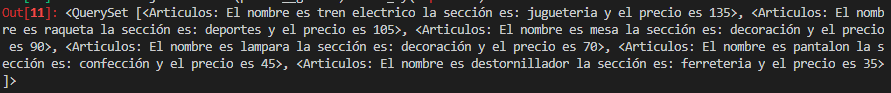
Resultado:



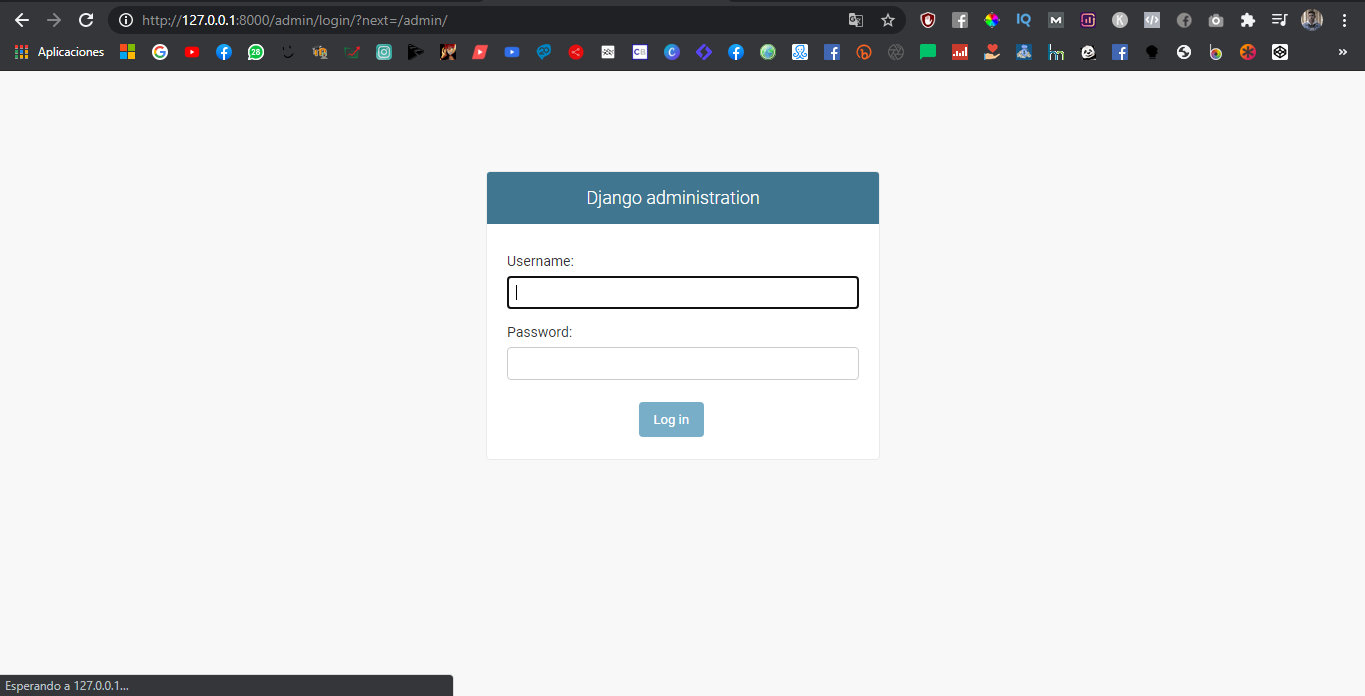
Ahora vamos a ordenarlos de manera descendente. Para esto agregamos un menos dentro del order\_by(), es decir un menos en el atributo por el que queremos ordenar los resultados.



Resultado:



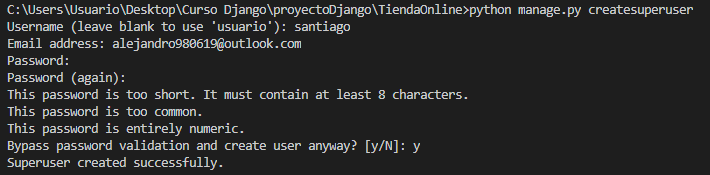
# Panel de Administración



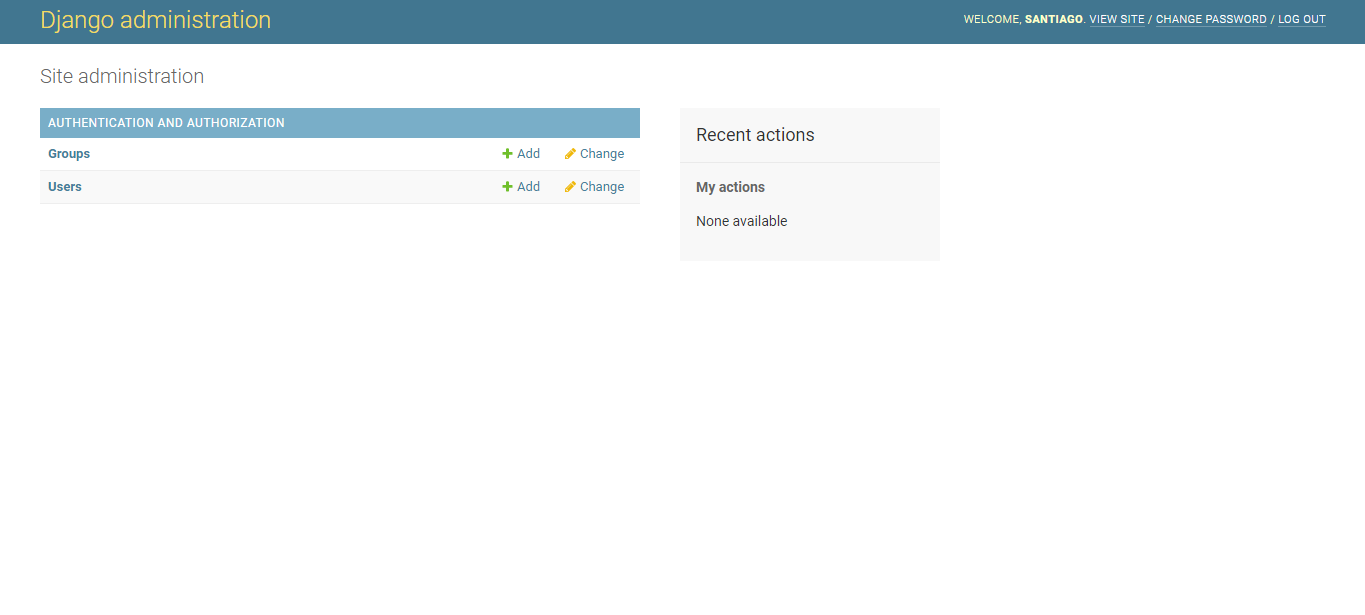
## Crear súper usuario

Para crear un súper usuario o un usuario administrador del plan de administración hacemos lo siguiente:

Ingresamos el siguiente código “Python manage.py createsuperuser” y luego simplemente nos empezaran a salir varias opciones, la primera es una opción que nos da un nombre de usuario automático pero tenemos la opción de cambiarlo si deseamos, luego nos pide un correo y una contraseña, en este caso la contraseña es: 123456, como es muy básica nos arroja un mensaje de que la contraseña no es segura, sin embargo nos da la opción de yes para dejarla tal cual.



Ingresando al Panel de Administración



## Importando Modelos:

Lo primero que debemos hacer en el archivo admin.py de nuestra aplicación es importar los modelos que usaremos en el Panel de Administración.

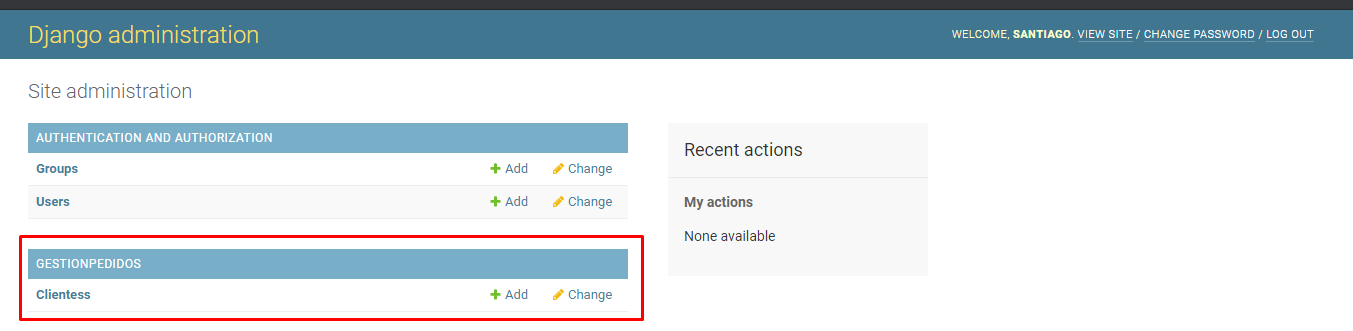


Luego registramos el modelo para que aparezca en el Panel de Administración

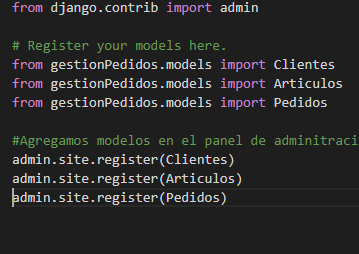


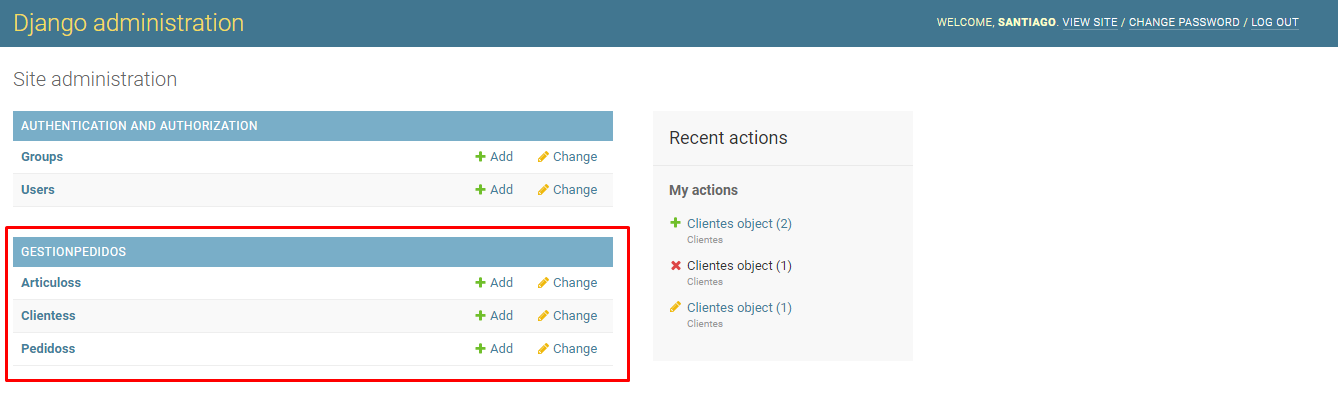
Ahora ejecutamos el servidor, entramos como administradores y observaremos lo siguiente:

Cabe destacar que Django siempre agrega un “s” al nombre del modelo o tabla, esto para que se va de manera plural y para tenerlo en cuenta.

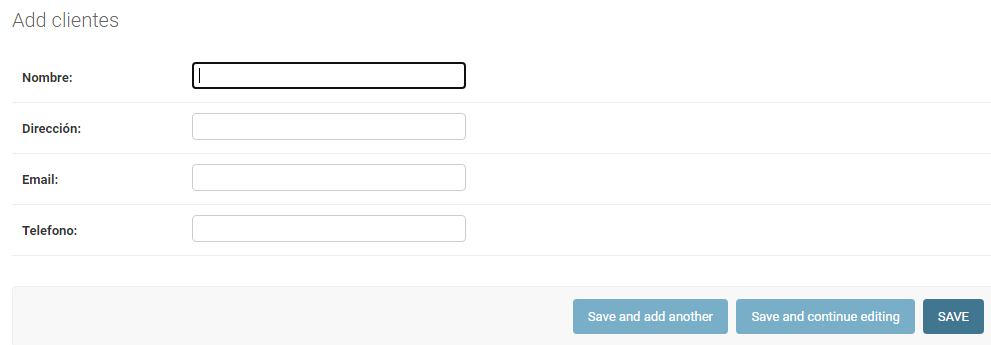


Agregando el resto de modelos:

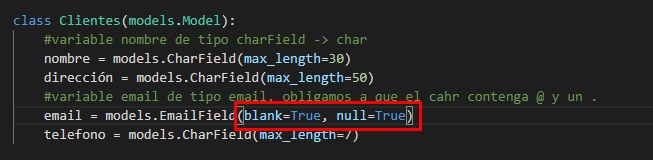




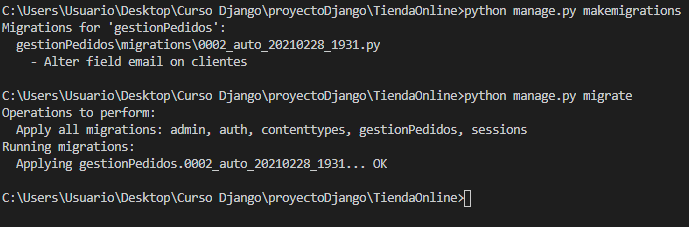
Ahora ya podemos agregar, eliminar y modificar clientes, artículos y pedidos. Es importnate resaltar que si nosotros agregamos un dato de alguno de los modelos, por ejemplo cliente, debemos dar en add y nos saldrán los campos para crear el nuevo cliente



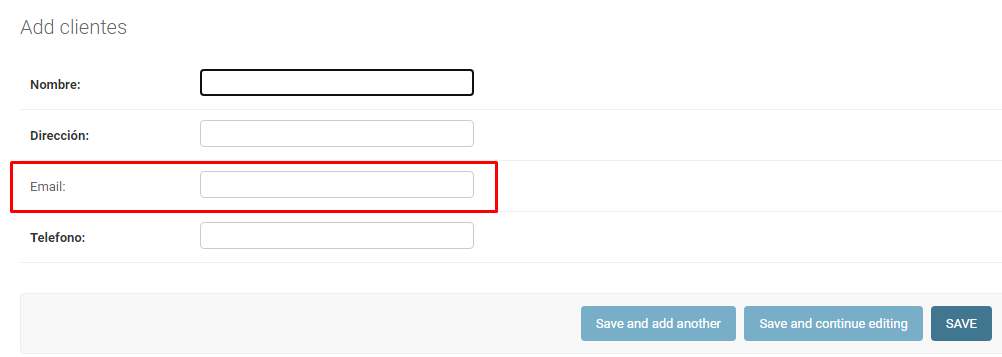
Si agregamos los datos y le damos en save, efectivamente el nuevo cliente se va a cargar a la base de datos y se va a crear. Pero lo importante aquí es que si podemos ver todos los campos están en negrilla, eso indica que es obligatorio ingresar información en cada uno de ellos, así que si queremos que alguno de los campos sea opcional, por ejemplo el email de un cliente debemos agregar lo siguiente en atributo email del modelo Cliente:



Por supuesto hacemos las respectivas migraciones.

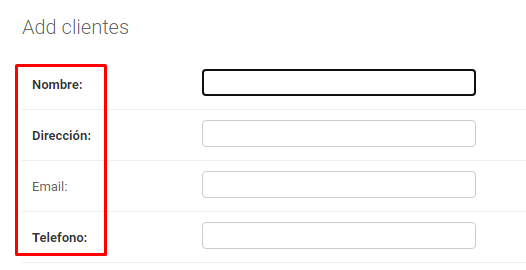


Al realizar dicho cambio observamos que al agregar un nuevo cliente, el campo de email ya no aparece en negrilla, indicando que no es obligatorio ingresar ese dato.



## Cambiar nombre de los campos de las tablas – solo Panel de Administración

Como podemos observar cuando queremos agregar un nuevo dato de alguna tabla, los nombres de los campos son colocados de acuerdo a los nombres de los atributos de la tabla en la base de datos.



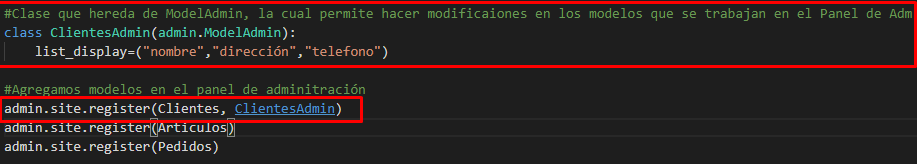
Si nosotros queremos cambiar estos nombres del Panel de Administración, podemos hacer lo siguiente, agregar el siguiente campo al atributo de la tabla o del modelo cuando lo estamos creando “verbose\_name=”Nombre\_de\_camp\_en\_Panel\_de\_Adminitración””



## Agregar campos en la tabla de Panel de Administración

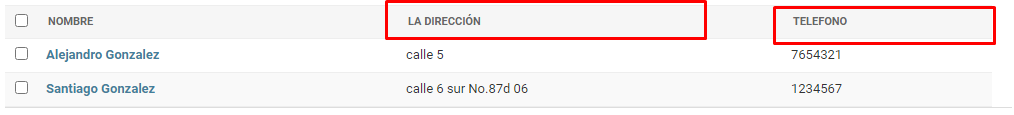
Si queremos que se visualicen más campos al momento de listar cualquiera de los modelos o tablas desde el Panel de Administración, hacemos lo siguiente:

Primero nos ubicamos en el archivo admin.py y creamos una clase que se encargada de modificar los campos de la tabla en el Panel, dicha clase debe heredar de admin.ModelAdmin, la cual es la que nos permite hacer dichas modificaciones, luego iría el siguiente código: “list\_display()” y dentro de los paréntesis entre comillas y separados por comas irían tal cual los nombres de los campos del modelo que queremos que aparezcan en la tabla del Panel de Administración. Ademas cuando carguemos el modelo con el código: “admin.site.register()” ademas de agregar dentro de los paréntesis con comillas el nombre del modelo, debemos agregar el nombre de la clase que contiene la configuración de los nombres de las columnas de la tabla.



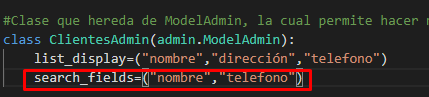
Luego simplemente hacemos las respectivas migraciones e iniciamos el servidor.

Como observamos se han agregado los campos:



## Barra de búsqueda

Si queremos agregar una barra de busqueda para los modelos del Panel de Adminitración agregamos lo siguiente, haremos uso de la misma clase que utilizamos arriba y simplemente agregamos lo siguiente: “search\_fields()” y dentro del paréntesis con comillas y separados por coma agregamos los nombres de los campos por lo que queremos hacer busqueda.



Ahora si listamos clientes en el Panel de Administración, podremos observar cómo se a agregado una barra de búsqueda, con la que podremos buscar datos por nombre y teléfono.

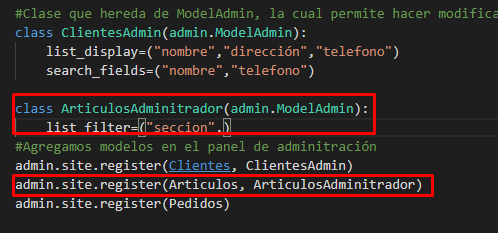


## Aplicando filtros a columnas de las tablas

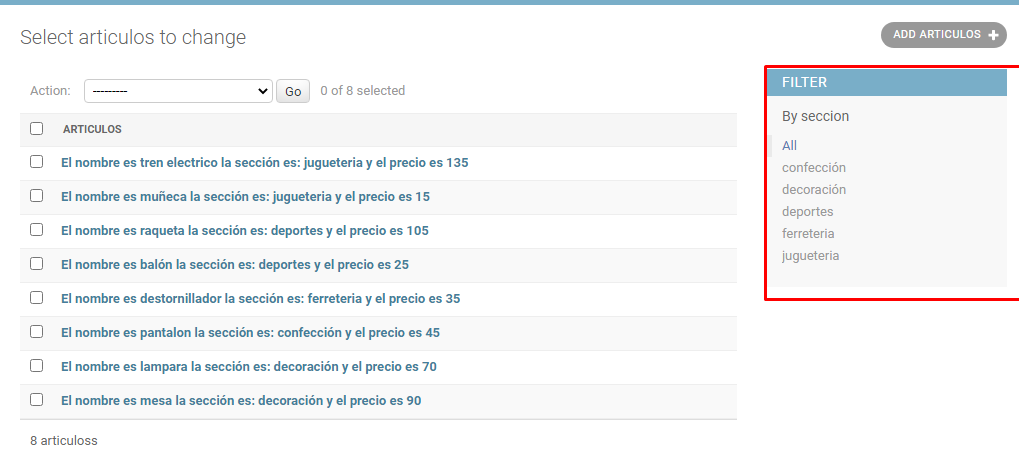
Si deseamos agregar la opción de que el administrador pueda filtrar los datos de una tabla por alguno de sus atributos, hacemos lo siguiente:

Creamos una clase con la que vamos a editar los campos del modelo, la clase dentro de los parámetros debe contener la herencia de “admin.ModelAdmin” y dentro de la clase agreamos lo siguiente “list\_filter()” dentro de los paréntesis, con comillas y separados por comas los nombres de los campos a los que queremos agregarle un filtro. Cabe destacar que para este código es necesario siempre, antes del cierre del paréntesis poner una coma.

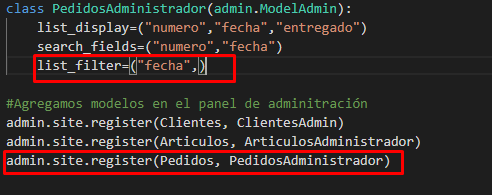
Luego, agregamos el nombre de la clase dentro de los parámetros de la carga del módulo y nos debería quedar algo así:



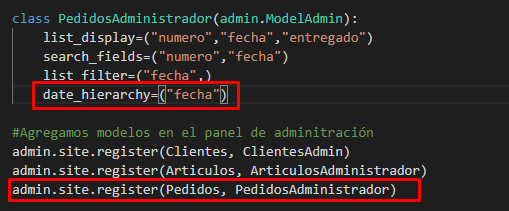
Ahora si observamos el modelo dentro del Panel de Adminitración, observamos que se ha agregado una tabla de filtros por los campos que hemos puesto, en este caso el filtro es solo por sección del artículo.



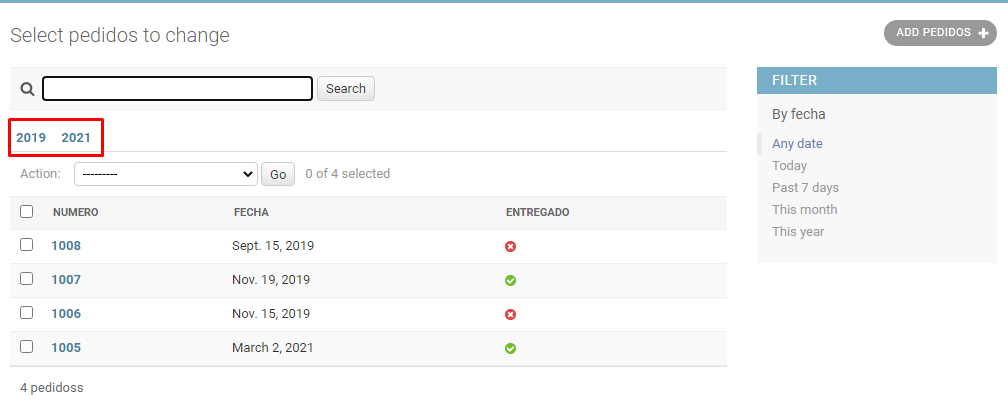
Agregando un filtro para el módulo de pedidos en el atributo fecha



Agregando otro tipo de filtros para fecha:

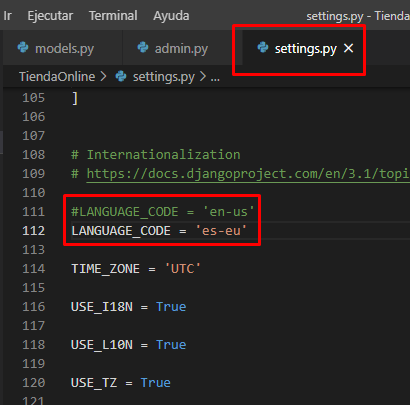


Obtendremos lo siguiente, otro filtro:



## Cambiar idioma del Panel de Administración

Lo que debemos hacer es simplemente ubicarnos en el archivo setting.py y buscar la línea de código al final que dice LANGUAGE\_CODE = “en-us”, simplemente cambiamos lo que está dentro del paréntesis por la codificación del idioma que queramos, en mi caso voy a cambiarlo a español:



## Usuarios

### Es staff:

Cuando un usuario tiene activo la opción staff, esto indica que el usuario puede entrar al Panel de Administración, puede existir un usuario con esta opción activa pero puede que no tenga permisos para hacer todos los cambios en el Panel de Administración

### Superusuario:

El superusuario es el usuario que puede modificar y acceder a todos los elementos de la interfaz administrativa

### Activo/no Activo

Los usuarios no activos son los usuarios que no pueden acceder a ninguna URL del sitio web Django que requiera una autenticación, no solo el panel de administración, también puede haber otros sitios web que requieran autenticación.

### Agregar, Modificar y Eliminar Usuarios

En el panel de administración podemos crear un usuario, al que deberemos asignarle un nombre de usuario, una contraseña y luego los permisos que anteriormente vimos, permisos de activo, de superusuario y de staff.

También podemos modificar los permisos e información de cualquier usuario simplemente seleccionando al usuario al que queremos realizarle el cambio desde el Panel de Administración listando todos los usuarios.

Por ultimo también podemos eliminar a cualquier usuario, simplemente dando en la opción delete o eliminar

## Grupos

Imaginemos que tenemos 500 usuarios en nuestra aplicación y de esos 500 necesitamos darle a 100 de esos usuarios los mismos permisos, en ese caso para no tener que ir asignando permisos a cada usuario, se crean los grupos

### Crear grupo

Para crear un grupo simplemente nos vamos a lista de grupos del Panel de Administración, damos en añadir grupo, colocamos el nombre del grupo y asignamos las diferentes funciones que tendrán los usuarios que hagan parte de dicho grupo.

# Envió de correo

Para enviar correos lo primero que se debe hacer es buscar un servidor de correo que permita usar sus parámetros para que desde Django poder enviar los correos, esos parámetros se deben especificar en el archivo settings.py, puede ser en la parte final de código. Las configuraciones pueden ser las siguientes:

Juan pablo velandia

4 grpos 3-6 personas

Cly managment

Desembolso

Core

Microservicios

Línea de crédito

Microcrédito

Maquina de estados

Etapas

Solicitud

Desembolsos

Visual

Linux

Cliente VPN

Workbench

Postman

Rubi==2.7

Sg bases de datos