Entrega Reto 2 parte 2 Bases de datos

Jonathan Betancur

Alejandro Restrepo

Felipe Uribe

Pablo Báez

Algunas indicaciones para conectar django con mysql correctamente

crear el folder donde va a estar todo el proyecto: instrucción: "django-admin startproject (nombreproyecto)"

cd (nombreproyecto)

Crear la app con la que se va a conectar la base de datos

python manage.py startapp (nombre app)

instalar mysqlclient por medio de pip para conectar mysql con django: "pip install mysqlclient"

Configurar settings.py en la sección de: DATABASE, con los datos de la base de datos, ejm:

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.mysql',
      'NAME': 'otro',
      'USER': 'root',
      'PASSWORD': '1234',
      'HOST':'localhost',
      'PORT':'3306',
   }
}
```

Con el comando: "python manage.py inspectdb" se podrá ver la base de datos que está cargando de mysql

y con el comando: "python manage.py inspectdb > (ruta de models.py)" se cargarán todas las clases al archivo models.py

o simplemente con la primera instrucción inspectdb se podrá copiar y pegar en el mismo archivo las clases que saca

Siguiente paso:

Ejecutar el comando: "python manage.py makemigrations" para ejecutar las clases creadas en models.py y que las mande al mysql, después, ejecutar el comando: "python manage.py migrate" para lanzar lo antes dicho

#Nota: cada que se haga algún cambio en el archivo models.py, se tienen que volver a ejecutar las 2 instrucciones previas

'''

Ejemplo Insertando un registro desde la terminal del proyecto hacia mysql:

python manage.py shell #se tiene que abrir un shell de python

from administracion.models import Mesas #se tiene que importar la tabla(modelo) con el que se vaya a trabajar

mesas = Mesas(idmesas=3, color = 'Cafe', tamaño= '60x70cm')

mesas.save() #el save, ejecuta la consulta que es mandada directamente a la base de datos

'''

'''

Ejemplo Actualización de un valor de una tabla

mesas.color= 'Azul'

mesas.save()

'''

'''

Ejemplo Borrado de un registro de una tabla

mesaborrado = Mesas.objects.get(idmesas=3)

mesaborrado.delete()

'''

'''

Ejemplo consultas con where

En la zona que queramos hacer la consulta, hay que definir una función para retonar el string formateado que queramos con los valores de la tabla, ejemplo: con la tabla usuarios

por ejemplo:

def __str__(self):

return 'El usuario con id: %s, se llama %s %s, con celular: %s, correo: %s, nacido en: %s, y es: %s ' % (self.idusuarios, self.nombre, self.apellido, self.celular, self.correo, self.fecha_nacimiento, self.ocupacion_idocupacion)

luego de volver a ejecutar el makemigrations y el migrate, volvemos a la shell de python, importamos la tabla que queramos:from administracion.models import Usuarios

y posteriormente, filtramos por el where que queramos, por ejemplo:

Usuarios.objects.filter(idusuarios=1)

RESULTADO:<QuerySet [<Usuarios: El usuario con id: 1, se llama Jonathan Betancur Espinosa, con celular: 3043910313, correo: Jonathanbetancurespinosa@gmail.com, nacido en: 2003-11-02, y es: Ocupacion object (2) >]>

'''

Configuraciones para mostrar panel de administración

Crear superusuario para acceder a las funciones de insertar, modificar, eliminar y actualizar usuarios en las tablas:

python manage.py createsuperuser: esto, si las tablas quedaron bien insertadas, pedirá unos campos (nombre, correo y contraseña) con la que se podrá acceder al panel de administración de django por la ruta: localhost:8000/admin, si no deja, el error puede ser porque dice que la tabla auth_user no tiene admite campo: last-login nulo, simplemente hay que cambiarlo para que por defecto sea nulo.

Algunos errores posteriores:

1.no se puede insertar un registro o modificarlo porque la columna "name" está vacía:https://stackoverflow.com/questions/55007406/django-migration-error-field-name-doesnt-have-a- default-value

Opciones varias en el panel de administración:

1. Configurar algún campo para que no sea requerido su valor en la inserción desde el panel de admin:

En el campo de la clase que se requiera, en los paréntesis, hay que poner esto: blank=True, null=True

2. Configurar algún titulo de un campo para que no sea tal cual el que esté en las tablas:

En el campo de la clase que se requiera, en los paréntesis, hay que poner: verbose_name="nombre que quiera"

3. Configurar campos que se quieren mostrar en las tablas: se creará una clase dentro del archivo admin.py (y previamente habiendo importado las clases del archivo models.py) el cual llamará al modelo correspondiente, y podremos especificar cuales campos queremos agregar con: list_display, ejemplo: class Iluminacionadmin(admin.ModelAdmin):

list_display= ("idiluminacion", "color", "origen", "regulacion", "ubicacion")

4.Configurar con cuales campos se harán las búsquedas de registros en el panel de administración entre cada tabla:

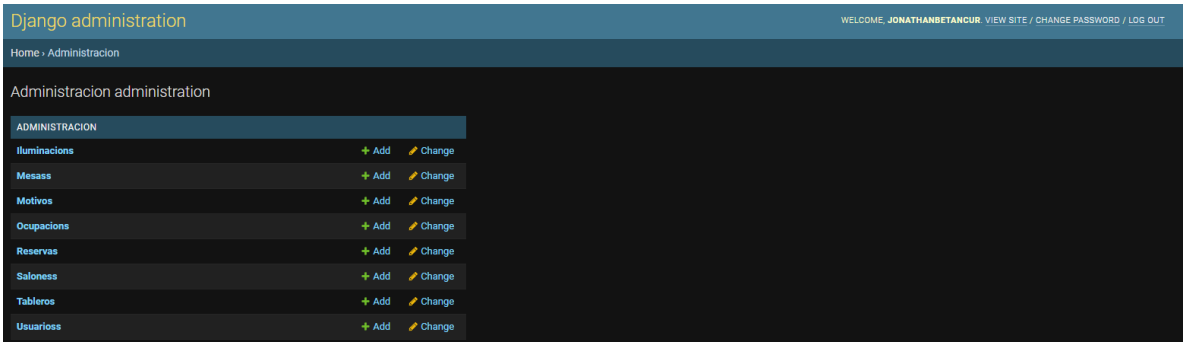
El procedimiento está encadenado con el anterior, pero solo agregaremos: search_fields=(""), y agregaremos los campos con los cuales idenfiticará las búsquedas, ejemplo:

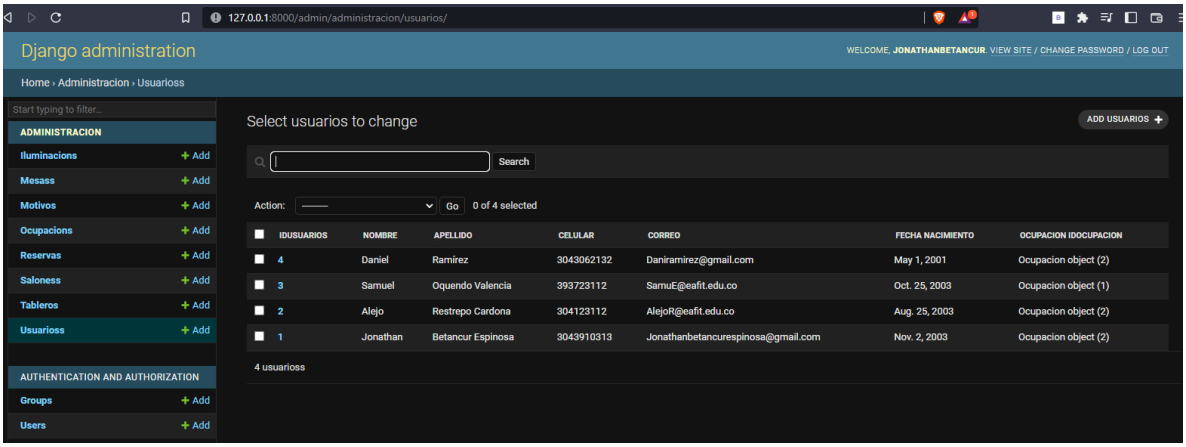
class Mesasadmin(admin.ModelAdmin):

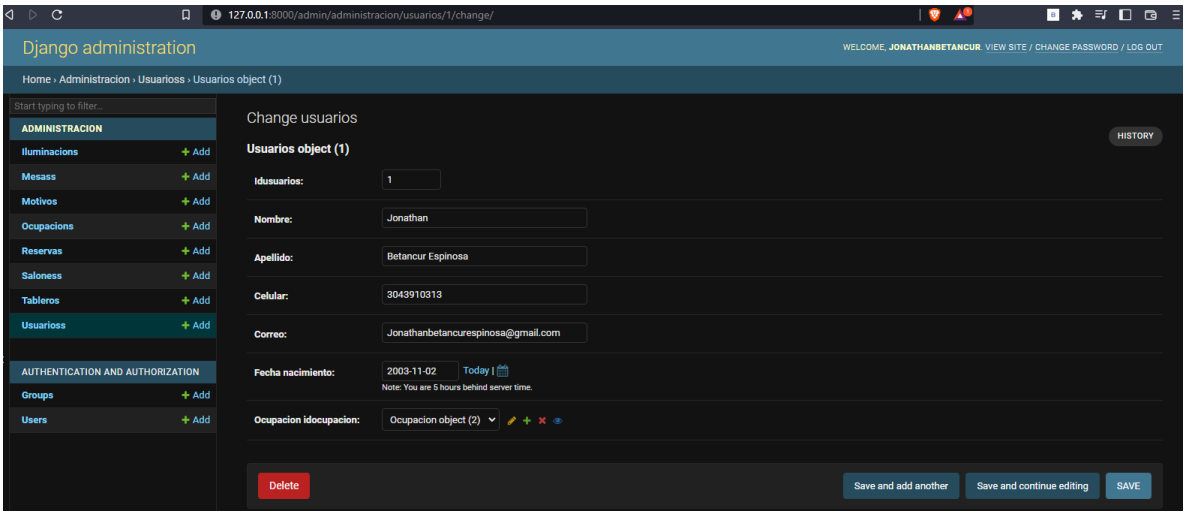
list_display= ("idmesas", "color", "tamaño")

search_fields= ("idmesas", "color", "tamaño")

Algunas imágenes de la vista del panel de administración resultante:







Con lo anterior ya hecho, podremos actualizar, insertar, consultar, y eliminar los registros de las tablas que se requieran, por lo que cumple el CRUD

Eliminar Migraciones

Restablecer Python manage.py migrate --fake-initial