

## Base De Datos SQLite – Django

Andres Felipe Gonzalez Berdugo

Servicio Nacional De Aprendizaje

Centro Minero sede iteandes

Análisis y desarrollo de software (ADSO)

Ficha:2993648

Andrés Felipe Sandoval Higuera

Sogamoso, Boyacá

Noviembre 2025

## **Contenido**

INTRODUCCION .....	3
Inserción de datos.....	4
Actualización de datos .....	5
Eliminar datos.....	6
Ejercicio 1.....	7
Ejercicio 2.....	9
CONCLUSIONES.....	10

# INTRODUCCION

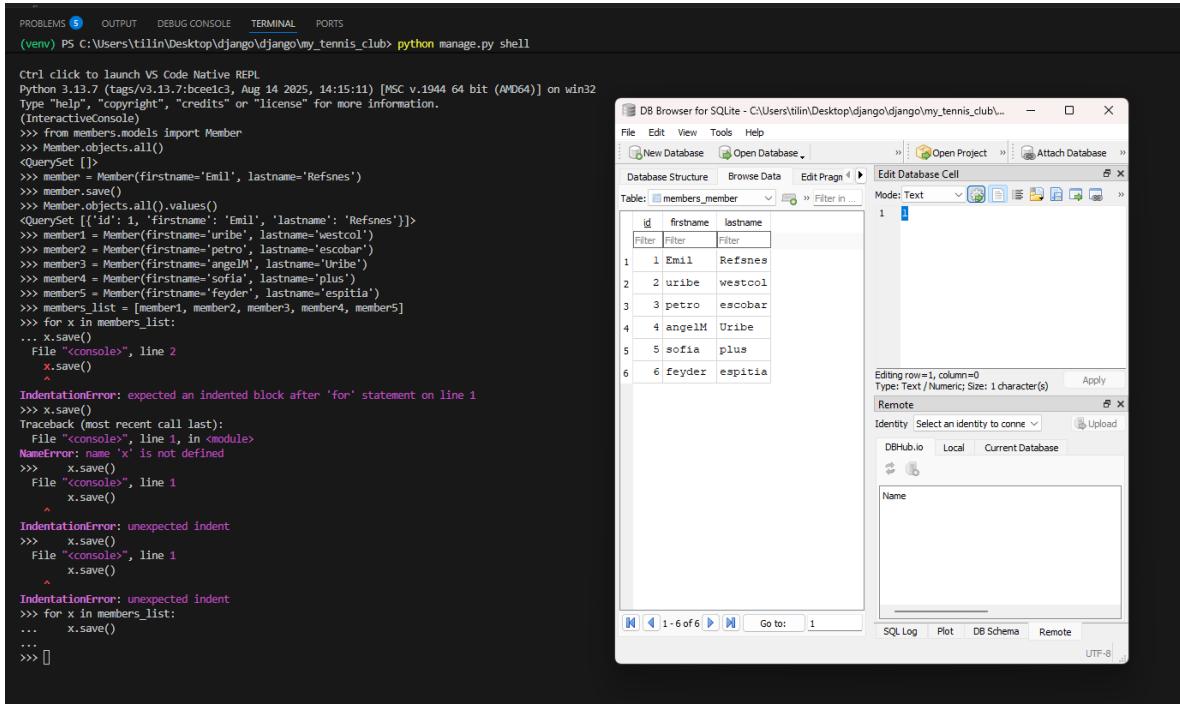
En este trabajo se dará a conocer todo lo visto en base de datos con el Framework Django usando el programa SQLite, mostrando en capturas de pantalla los comandos básicos vistos en clase.

Se verán varios puntos como los son la inserción de datos, consulta de datos, actualización de datos y la eliminación de los mismos.

Con este trabajo pretendo demostrar el buen uso y entendimiento de lo visto en clase.

# Base de datos SQLite Django

## Inserción de datos



The screenshot shows a development environment with two main windows. On the left, the VS Code terminal window displays Python code being run against a Django database. The code creates a `Member` object with various attributes and saves it to the database. On the right, the DB Browser for SQLite application shows a table named `members\_member` with six rows of data.

```
Ctrl click to launch VS Code Native REPL
Python 3.13.7 (tags/v3.13.7-bcceed3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from members.models import Member
>>> Member.objects.all()
<QuerySet []>
>>> member = Member(firstname='Emil', lastname='Refsnes')
>>> member.save()
>>> Member.objects.all().values()
<QuerySet [{'id': 1, 'firstname': 'Emil', 'lastname': 'Refsnes'}]>
>>> member1 = Member(firstname='uribe', lastname='westcol')
>>> member2 = Member(firstname='petro', lastname='escobar')
>>> member3 = Member(firstname='angelM', lastname='Uribe')
>>> member4 = Member(firstname='sofia', lastname='plus')
>>> member5 = Member(firstname='reyder', lastname='espitia')
>>> members_list = [member1, member2, member3, member4, member5]
>>> for x in members_list:
...     x.save()
File "<console>", line 2
    x.save()
...
IndentationError: expected an indented block after 'for' statement on line 1
>>> x.save()
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'x' is not defined
>>>     x.save()
  File "<console>", line 1
      x.save()
  ^
IndentationError: unexpected indent
>>>     x.save()
  File "<console>", line 1
      x.save()
  ^
IndentationError: unexpected indent
>>> for x in members_list:
...     x.save()
...
>>> []

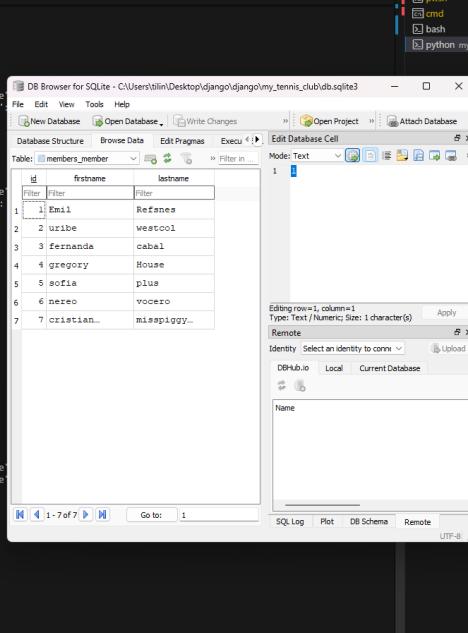
```

	id	firstname	lastname
1	1	Emil	Refsnes
2	2	uribe	westcol
3	3	petro	escobar
4	4	angelM	Uribe
5	5	sofia	plus
6	6	reyder	espitia

Se insertaron los datos a la tabla de datos previamente creada con “name” y “lastname”, esto se hizo mediante una variable “x” para insertar los datos

# Actualización de datos

```
(venv) PS C:\Users\tilin\Desktop\django\my_tennis_club> python manage.py shell
django.db.utils.OperationalError: database is locked
>>> x = Member.objects.all()[5]
>>> x.firstname
'feyder'
>>> x.firstname = "nero"
>>> x.save()
>>> Member.objects.all().values()
<QuerySet [{"id": 1, "firstname": "Emil", "lastname": "Refsnes"}, {"id": 2, "firstname": "uribe", "lastname": "westcol"}, {"id": 3, "firstname": "Urube"}, {"id": 4, "firstname": "fernanda", "lastname": "cabal"}, {"id": 5, "firstname": "sofia", "lastname": "plus"}, {"id": 6, "firstname": "nero", "lastname": "espitia"}, {"id": 7, "firstname": "espitia"}, {"id": 8, "firstname": "vocero"}]>
>>> Member.objects.all().values()
<QuerySet [{"id": 1, "firstname": "Refsnes"}, {"id": 2, "firstname": "uribe", "lastname": "westcol"}, {"id": 3, "firstname": "Urube"}, {"id": 4, "firstname": "fernanda", "lastname": "cabal"}, {"id": 5, "firstname": "sofia", "lastname": "plus"}, {"id": 6, "firstname": "nero", "lastname": "espitia"}, {"id": 7, "firstname": "espitia"}, {"id": 8, "firstname": "vocero"}]>
>>> x = Member.objects.all()[3]
>>> x.firstname
'angeIM'
>>> x.firstname = "gregory"
>>> x.save()
>>> x = Member.objects.all()[3]
>>> x.lastname
'Urube'
>>> x.lastname = "House"
>>> x.save()
>>> Member.objects.all()[2]
>>> x.firstname
'petro'
>>> x.firstname = "fernanda"
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.lastname
'cabal'
>>> x.lastname = "cabal"
>>> x.save()
>>> Member.objects.all().values()
<QuerySet [{"id": 1, "firstname": "Emil", "lastname": "Refsnes"}, {"id": 2, "firstname": "uribe", "lastname": "westcol"}, {"id": 3, "firstname": "Urube"}, {"id": 4, "firstname": "fernanda", "lastname": "cabal"}, {"id": 5, "firstname": "sofia", "lastname": "plus"}, {"id": 6, "firstname": "nero", "lastname": "vocero"}, {"id": 7, "firstname": "espitia"}, {"id": 8, "firstname": "vocero"}]>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```



ID	firstname	lastname
1	Emil	Refsnes
2	uribe	westcol
3	fernanda	cabal
4	gregory	House
5	sofia	plus
6	nero	vocero
7	christian...	misspiggy...

Por medio de los comandos que serán presentados en un momento se puede editar la información de las casillas designando a que usuario o en que casilla se van a hacer cambios

```
>>> from members.models import Member

>>> Member.objects.all()

>>> x = Member.objects.all()[4]

>>> x.lastname

>>> x.lastname = "cabal"
```

## Eliminar datos

```
>>> Member.objects.all().values()
<QuerySet [{"id": 1, "firstname": "Emil", "lastname": "Refsnes"}, {"id": 2, "firstname": "uribe", "lastname": "westcol"}, {"id": 3, "firstname": "petro", "lastname": "escobar"}, {"id": 4, "firstname": "angelM", "lastname": "angelm"}, {"id": 5, "firstname": "sofia", "lastname": "plus"}, {"id": 6, "firstname": "nereo", "lastname": "vocero"}, {"id": 7, "firstname": "cristian\n", "lastname": "misspiggy\n"}]>
>>> Member.objects.all()[3]
>>> x.firstname
'angelm'
>>> x.firstname = "gregory"
>>> x.save()
>>> x = Member.objects.all()[3]
>>> x.lastname
'Uribe'
>>> x.lastname = "House"
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.firstname
'petro'
>>> x.firstname = "fernanda"
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.lastname
'escobar'
>>> x.lastname = "cabal"
>>> x.save()
>>> Member.objects.all().values()
<QuerySet [{"id": 1, "firstname": "Emil", "lastname": "Refsnes"}, {"id": 2, "firstname": "uribe", "lastname": "westcol"}, {"id": 3, "firstname": "nereo", "lastname": "vocero"}, {"id": 4, "firstname": "cristian\n", "lastname": "misspiggy\n"}, {"id": 5, "firstname": "sofia", "lastname": "plus"}, {"id": 6, "firstname": "fernanda", "lastname": "cabal"}]>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>> x = Member.objects.all()[4]
>>> x.delete()
(1, {'members.Member': 1})
>>> Member.objects.all()[3]
>>> x.delete()
(1, {'members.Member': 1})
>>> x = Member.objects.all()[2]
>>> x.delete()
(1, {'members.Member': 1})
>>> x.save()
>>> ]
```

Como se puede evidenciar en el recuadro rojo se uso el comando “x.delete()” el comando es similar al de editar caracteres

```
>>> x = Member.objects.all()[4]
>>> x.delete()
```

# Ejercicio 1

Se tienen que agregar 2 nuevas columnas las cuales serán el teléfono y fecha de entrada las cuales se agregarán mediante código en el archivo “models.py” como lo podemos apreciar en la página w3schools compartida en clase, luego de hacer los cambios en el código de “models.py” tendremos que realizar las migraciones a la base de datos para que esta se actualice

```
my_tennis_club/members/models.py :
```

```
from django.db import models

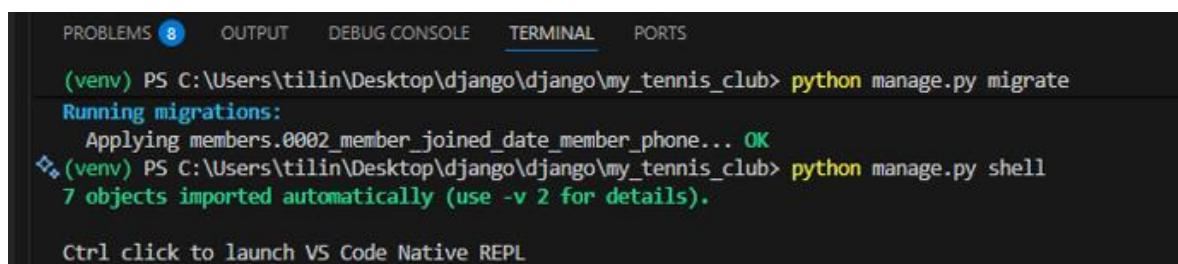
class Member(models.Model):
    firstname = models.CharField(max_length=255)
    lastname = models.CharField(max_length=255)
    phone = models.IntegerField()
    joined_date = models.DateField()
```

As you can see, we want to add `phone` and `joined_date` to our Member Model.

This is a change in the Model's structure, and therefore we have to make a migration to tell Django that it has to update the database:

```
python manage.py makemigrations members
```

Luego se usará el código Python `manage.py migrate` con el cual sabremos si migraron de manera correcta las columnas anteriormente creadas



```
PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) PS C:\Users\tilin\Desktop\django\django\my_tennis_club> python manage.py migrate
Running migrations:
  Applying members.0002_member_joined_date_member_phone... OK
❖ (venv) PS C:\Users\tilin\Desktop\django\django\my_tennis_club> python manage.py shell
7 objects imported automatically (use -v 2 for details).

Ctrl click to launch VS Code Native REPL
```

Una vez hecho esto se proceden a agregar los datos de los usuarios a las 2 nuevas columnas, en mi caso en el código se aprecian errores por mala sintaxis al momento de realizarse, sintaxis que corregí allí mismo

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

```
(venv) PS C:\Users\tillin\Desktop\django\my_tennis_club> python manage.py migrate
Running migrations:
  Applying members.0002_member_joined_date_member_phone... OK
7 objects imported automatically (use -v 2 for details).

Ctrl click to launch VS Code Native REPL
Python 3.13.7 (tags/v3.13.7-bce1c3, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> from members.models import Member
>>> x = Member.objects.all()[0]
>>> x.phone = 3285512
>>> x.joined_date = '2025-11-13'
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.phone = 334534
>>> x.joined_date = '2025-11-13'
>>> x = Member.objects.all()[6]
Traceback (most recent call last):
  File "<console>", line 1, in <module>
    File "C:\Users\tillin\Desktop\django\venv\lib\site-packages\django\utils\functional.py", line 356, in __call__
      return self.result_cache[0]
     ~~~~~^M
IndexError: list index out of range
>>> x = Member.objects.all()[3]
>>> x.phone = 3234345
>>> x.joined_date = '2025-11-13'
>>> x = Member.objects.all()[4]
>>> x.phone = 314134234
>>> x.joined_date = '2025-11-13'
>>> x = Member.objects.all()[4]
>>> x.phone = 356546346
>>> x.joined_date = '2025-11-13'
>>> x = Member.objects.all()[5]
Traceback (most recent call last):
  File "<console>", line 1, in <module>
    File "C:\Users\tillin\Desktop\django\venv\lib\site-packages\django\utils\functional.py", line 356, in __call__
      return self.result_cache[0]
     ~~~~~^M
IndexError: list index out of range
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.phone = 323414
>>> x.joined_date = '2025-11-13'
>>> x.save()
>>> x = Member.objects.all()[1]
>>> x.phone = 3123451
>>> x.joined_date = '2025-11-13'
>>> x.save()
>>> x = Member.objects.all()[3]
>>> x.phone = 323434
>>> x.joined_date = '2025-11-13'
>>> x.save()
>>> []

```

DB Browser for SQLite - C:\Users\tillin\Desktop\django\my\_tennis\_club\db.sqlite3

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Undo

Open Project Attach Database

Database Structure Browse Data Edit Pragmas Execute SQL

Table: members\_member Mode: Text

	id	firstname	lastname	joined_date	phone
1	1	Emil	Refsnes	2025-11-13	3285512
2	2	uribe	westcol	2025-11-13	3123451
3	6	nereo	vocero	2025-11-13	323414
4	7	cristi...	misspig...	2025-11-13	323434
5	8	fernanda	cabal	2025-11-13	356546346

Editing row=1, column=0  
Type: Text/Numeric; Size: 1 character(s) Apply

Remote Select an identity to conn: DBHub.io Local Current Database

Name

SQL Log Plot DB Schema Remote UTF-8

Como resultado final se logra evidenciar en la tabla de datos que se ingresaron correctamente los datos en cada columna

## Ejercicio 2

En este caso debemos agregar 3 nuevas columnas a la base de datos, entonces se hará de la manera explicada antes, son prácticamente los mismos códigos lo único que cambia en este caso son nombres y variables ya que por mi parte los nombres de las nuevas columnas a las cuales voy a a agregar información en la base de datos son (salario, email y peso)

The terminal window shows the following command sequence:

```
django > my_tennis_club > members > models.py > 0003_member_email_member_peso_member_salario.py 3. U
djang... > my_tennis_club > members > models.py > Member
1   from django.db import models
...
3   # Create your models here.
4   class Member(models.Model):
5       firstname = models.CharField(max_length=255)
6       lastname = models.CharField(max_length=255)
7       phone = models.IntegerField(null=True)
8       joined_date = models.DateField(null=True)
9       salario = models.FloatField(null=True)
10      email = models.EmailField(null=True)
11      peso = models.FloatField(null=True)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

(venv) PS C:\Users\tilin\Desktop\django\my\_tennis\_club> #una gatica que le gusta el mamb...  
(venv) PS C:\Users\tilin\Desktop\django\my\_tennis\_club> python manage.py makemigrations members  
Migrating to 0003...  
members/migrations/0003\_member\_email\_member\_peso\_member\_salario.py  
+ Add field email to member  
+ Add field peso to member  
+ Add field salario to member  
(venv) PS C:\Users\tilin\Desktop\django\my\_tennis\_club> python manage.py migrate  
Operations to perform:  
 Apply all migrations: admin, auth, contenttypes, members, sessions  
Running migrations:  
 Applying members.0003\_member\_email\_member\_peso\_member\_salario... OK  
(venv) PS C:\Users\tilin\Desktop\django\my\_tennis\_club> python manage.py shell  
7 objects imported automatically (use -v 2 for details).

Ctrl click to launch Code Native REPL  
Python 3.13.7 (Tags/v3.13.7:b7cbe1c, Aug 14 2025, 14:15:11) [MSC v.1944 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
(InteractiveConsole)  
>>> from members.models import Member  
>>> Member.objects.all()  
<QuerySet [Member: Member object (1), Member: Member object (2), Member: Member object (6), Member: Member object (7), Member: Member object (8)>|>>> []

The DB Browser for SQLite window shows the 'members\_member' table with columns: id, firstname, lastname, joined\_date, phone, email, peso, and salario. The data is as follows:

id	firstname	lastname	joined_date	phone	email	peso	salario
1	Emil	Refsnes	2025-11-13	3285512	yaesoulultimo@gmail.com	67.0	2500000.0
2	uribe	westcol	2025-11-13	3123451	estonoesalg0@gmail.com	92.0	9000000.0
3	nero	vocero	2025-11-13	323414	alomsd@email.com	83.0	1500000.0
4	cristi	misspig	2025-11-13	323434	algo123@email.com	70.0	2500000.0
5	fernanda	cabal	2025-11-13	356546346	aigo@email.com	63.0	5000000.0

Aquí podemos evidenciar como se crearon las nuevas columnas para guardar información nueva, como dicho anteriormente (peso, email y salario)

The terminal window shows the following Python code:

```
>>> x.save()
>>> x = Member.objects.all()[3]
>>> x.peso
>>> x.peso = "78"
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.peso
>>> x.peso = "83"
>>> x.save()
>>> x = Member.objects.all()[1]
>>> x.peso
>>> x.peso = "92"
>>> x.save()
>>> x = Member.objects.all()[0]
>>> x.peso
>>> x.peso = "67"
>>> x.save()
>>> x = Member.objects.all()[4]
>>> x.email
>>> x.email = "algo@email.com"
>>> x.save()
>>> x = Member.objects.all()[3]
>>> x.email
>>> x.email = "algo123@email.com"
>>> x.save()
>>> x = Member.objects.all()[2]
>>> x.email
>>> x.email = "algoalgo@email.com"
>>> x.save()
>>> x = Member.objects.all()[1]
>>> x.email
>>> x.email = "estonopesalg0@gmail.com"
>>> x.save()
>>> x = Member.objects.all()[0]
```

The DB Browser for SQLite window shows the same 'members\_member' table with the new data added:

id	firstname	lastname	joined_date	phone	email	peso	salario
1	Emil	Refsnes	2025-11-13	3285512	yaesoulultimo@gmail.com	67.0	2500000.0
2	uribe	westcol	2025-11-13	3123451	estonoesalg0@gmail.com	92.0	9000000.0
3	nero	vocero	2025-11-13	323414	alomsd@email.com	83.0	1500000.0
4	cristi	misspig	2025-11-13	323434	algo123@email.com	70.0	2500000.0
5	fernanda	cabal	2025-11-13	356546346	aigo@email.com	63.0	5000000.0

Como se logra evidenciar se agregaron los datos a sus respectivas columnas de su respectivo usuario

## CONCLUSIONES

- Con este proyecto se pudo evidenciar un buen manejo de la plataforma Django para la gestión del proyecto
- se demostró buena practicidad en la manipulación y manejo de la base de datos
- Este proyecto funciona mucho para aprender los temas básicos de bases de datos con el Framework Django