crear entorno

python -m venv venv
source venv/bin/activate

pip install "Django>=5,<6" PyMySQL

***el proyecto se llama mi_proyecto incluir el punto .

django-admin startproject mi_proyecto .

***la app se llama productos

python manage.py startapp productos

mi_proyecto `/settings.py`:

```
INSTALLED_APPS = [
    # ...
    "productos",
]
```

“Shim” para que Django use PyMySQL como MySQLdb

Crea/edita mi_proyecto `/__init__.py` (misma carpeta que `settings.py`) y agrega:

```
import pymysql
pymysql.install_as_MySQLdb()
```

Conexión por host/puerto (universal)

mi_proyecto/settings.py:

"NAME": "miproyecto", va el nombre de la base de datos creada en mysql workbench

CREATE SCHEMA `miproyecto` DEFAULT CHARACTER SET utf8 ;

```
DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.mysql",
```

```
        "NAME": "miproyecto",
        "USER": "root",
        "PASSWORD": "root",
        'OPTIONS': {
            'unix_socket': '/Applications/MAMP/tmp/mysql/mysql.sock',
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        }
    }
}
```

4) Modelo ORM pedido (en `productos/models.py`)

Requisitos del enunciado:

- `nombre`: **único**
- `descripcion`: texto largo
- `precio`: **decimal con 2 decimales**
- `stock`: entero
- `fecha_creacion`: auto al guardar

```
from django.db import models

class Producto(models.Model):
    nombre = models.CharField(max_length=100, unique=True)
    descripcion = models.TextField()
    precio = models.DecimalField(max_digits=10, decimal_places=2)
    stock = models.IntegerField()
    fecha_creacion = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.nombre} (${self.precio})"
```

5) Migraciones

```
python manage.py check
python manage.py makemigrations productos
python manage.py migrate
```

6) CRUD en Django Shell (crear, leer, actualizar, borrar)

Abre shell:

```
python manage.py shell
```

Dentro del shell:

```
from decimal import Decimal
from productos.models import Producto

# a) CREAR por shell
producto = Producto(
    nombre = "Desodorante",
    descripcion = "Olor chocolate",
    precio = 1000,
    stock = 10
)
producto.save()


Producto.objects.create(
    nombre = "Shampoo de Mascota",
    descripcion = "Shampoo para perritos y gatitos",
    precio = 500.00,
    stock = 5
)

producto = Producto(
```

```
    nombre = "laca",
    descripcion = "para fijar tu peinado",
    precio = 8000,
    stock = 35
)
producto.save()

Producto.objects.create(
    nombre = "crema de cuerpo",
    descripcion = "propiedades nutritivas",
    precio = 6000,
    stock = 6
)

#listar por shell
#Obtener TODOS → Producto.objects.all()


#Obtener una lista de productos WHERE → Producto.objects.filter()
producto1 = Producto.objects.get(id=1) #nombre = "Desodorante"
print(producto1, producto1.descripcion, producto.precio, producto1.stock)

#actualizar por shell
producto1.precio = 989

#guardar el nuevo precio
producto1.save()
#borrar producto
producto1.delete()
```

Revisar productos en workbench

Clic derecho sobre `productos_producto` → **Select Rows - Limit 1000**.

Se abrirá un editor con la consulta:

```
SELECT * FROM productos_producto;
```

salir de shell con
exit()

7) (Opcional) Crear superusuario y probar admin

```
python manage.py createsuperuser
python manage.py runserver




http://127.0.0.1:8000/admin/
```

superusuario

user: root

pass: root

usuario@gmail.com

Para salir de forma correcta de un proyecto se ejecuta en la terminal lo siguiente:

CTRL+C → cerrar el server.

deactivate → salir del entorno virtual.

Cerrar la ventana de terminal si ya terminaste.