Tutorial Construcción una aplicación desde cero

Integrantes:

ARCIA QUINTERO CRISTIAN DAVID

SANCHEZ MORA JUAN JOSE

SAMACA GONZALEZ DANIEL MAURICIO

SOLANO GARCES JOSE ALEJANDRO

Proyecto: My building App

Grupo: LUDOPATAS

Descripción del proyecto: My Building App es una aplicación enfocada en optimizar la gestión administrativa y financiera de copropiedades (conjuntos o edificios).

Lenguaje y framework seleccionado: Python y Django

Librerías de ORM utilizadas: Django ORM

Pasos para la construcción

1) **Descarga de Docker Desktop**

2) **Creación del docker compose**
   a) El docker compose tiene el siguiente contenido:

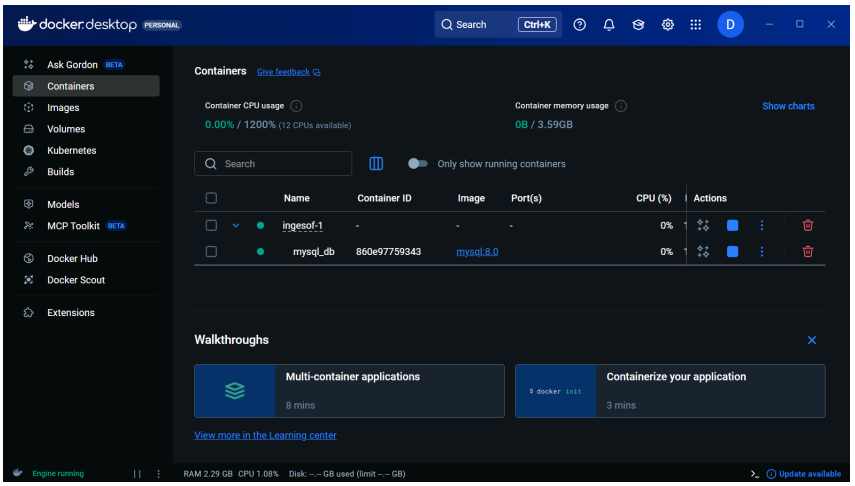
```yaml
version: '3.9'

services:
  mysql-db:
    image: mysql:8.0
    container_name: mysql_db
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: "1234"
    ports:
      - "3307:3306"
    volumes:
      - mysql_data:/var/lib/mysql
    networks:
      - webnet

volumes:
  mysql_data:

networks:
  webnet:
    driver: bridge
```

3) **Creación del .bat y el .sh para levantar los contenedores y servidor de base de datos.**
    a) La aplicación de Docker Desktop se debe ver algo parecido a



4) **Demostración de los datos en la base de datos montada en docker**

    a) Para comprobar el establecimiento de la base de datos en docker, entramos directamente al contenedor y haciendo uso de la terminal, se ejecuta lo siguiente

```shell
docker exec -it mysql_db mysql -u root -p
```

Ingresando la password configurada en el montaje se puede acceder a la consola de mysql y una vez dentro, revisar si las instancias existen, por lo que haciendo una consulta simple se puede ver lo siguiente

```
mysql> use MyBuildingApp
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select * from Usuario
    -> ;
+------------+--------+-----------+------------------+-----------+------------+---------+-------------+
| id_usuario | cedula | nombre    | fecha_nacimiento | correo    | contrasena | celular | rol         |
+------------+--------+-----------+------------------+-----------+------------+---------+-------------+
|          1 |     10 | Pepito123 | 2025-07-12       | pepito123 | 12345      | 322     | Propietario |
|          2 |     11 | Sofia     | 2025-10-14       | carol123  | 12345      | 322     | Residente   |
+------------+--------+-----------+------------------+-----------+------------+---------+-------------+
2 rows in set (0.02 sec)

mysql>
```

5) **Creación del entorno virtual para trabajar en django (opcional pero recomendado)**

    a) Para poder jugar con las diferentes versiones de un framework como lo es Django es recomendable crear un entorno virtual. Este funcionará para aislar un proyecto del resto del sistema y no tener conflicto especialmente en

versiones e incompatibilidades. Para activar este entorno se utilizó el comando :

```shell
python -m venv venv
```

b) No basta con solo crearlo, es necesario activarlo ejecutando el comando:

```shell
.\venv\Scripts\activate
```

6) Instalación del django y de librerias de traducción para el uso de mysql

a) Una vez se tiene el entorno activado (la terminal se muestra con un (venv) al inicio de cada línea) se procede con la instalación de Django.

```shell
pip install django
```

b) Para comprobar la instalación, revisamos la version ejecutando

```shell
django-admin --version
```

c) Como se pretende conectar Django con MySQL en lugar de SQLite que es la que viene por defecto, se necesitará de un conector de MySQL para python, y este se consigue ejecutando

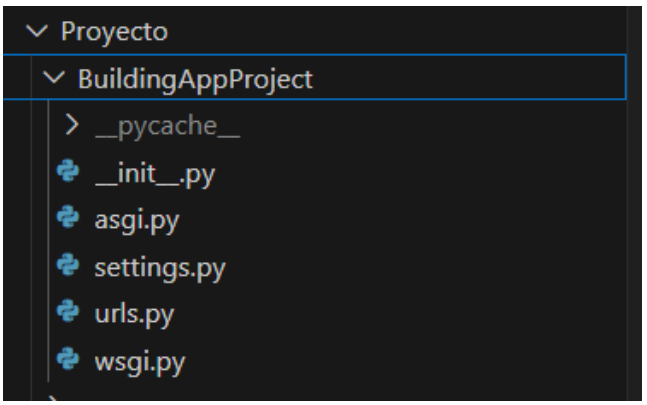
```shell
pip install mysqlclient
```

7) Creación del proyecto de django

a) Una vez todo está listo se procede con la creación del proyecto

```shell
django-admin startproject BuildingAppProyect
```

b) Al ejecutar el comando anterior, se crean algunos archivos que constituyen toda la configuración predeterminada del proyecto, algo similar a:



8) Modificación del archivo settings para referenciar la base de datos creada

a) Recordando que ya fue ejecutado el script de creación de la base de datos y que esta reside en un contenedor de docker con una imagen de MySQL, es necesario configurar el proyecto para que haga referencia a esta base de datos y no a la que tiene por defecto, por lo que en el archivo Proyecto\BuildingAppProyect/settings.py se modifica la sección de DATABASES de la siguiente manera:

```shell
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME'    : 'MyBuildingApp',   # Nombre exacto de tu base
        'USER'    : 'root',
        'PASSWORD': '1234',
        'HOST'    : '127.0.0.1',       # Porque tu servicio Docker está en localhost
        'PORT'    : '3307',            # Porque expusiste 3307:3306
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        },
    }
}
```

Con esto la configuración del proyecto en cuanto a base de datos es suficiente

9) Creación de un super usuario en Django

a) Para el uso adecuado de Django se necesita la creación de un super usuario. Este va tener todos los permisos para ejecutar un CRUD básico y en general para gestionar todo el proyecto.

```Shell
python manage.py createsuperuser
```

b) Para comprobar que todo salió bien y con el fin de levantar el servidor por primera vez se puede ejecutar

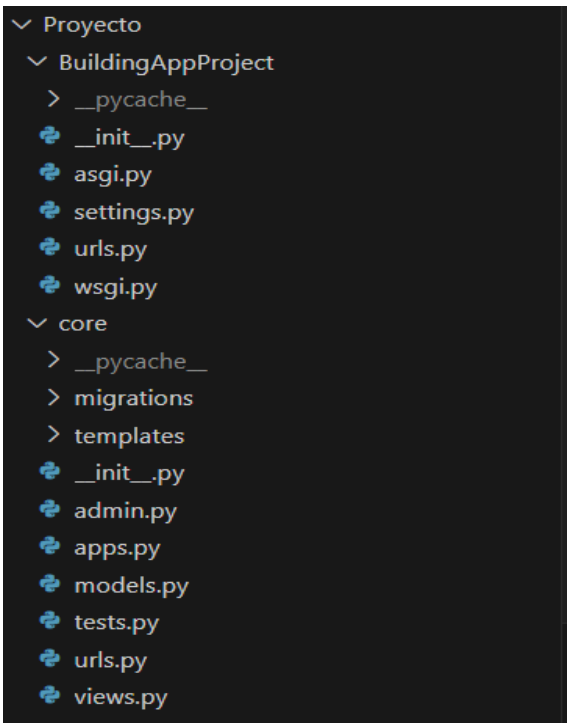
```Shell
python manage.py runserver
```

10) Creación de una app de Django (core)

a) Django estructura los proyectos mediante aplicaciones independientes (apps), cada una encargada de una funcionalidad concreta del sistema. Para este caso crearemos una app que se encargará de mostrar registros de la base de datos en la interfaz gráfica.

```Shell
python manage.py startapp core
```

Una vez ejecutado el comando anterior se debería tener una estructura como la siguiente:



Nota: Hasta este punto los archivos de url, templates y migración en core no necesariamente tienen que estar disponibles.

11) Configuración del app en el settings del proyecto

a) Posterior a la creación de la app, se requiere añadirla en la app del proyecto, esto se hace alterando el archivo de Proyecto\BuildingAppProyecto\settings.py y poniendo esta nueva app en la sección de apps, esta sección tendría que quedar de la siguiente manera:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core',
]
```

12) Migración de los datos a la app de modelos

a) Para hacer uso del ORM de Django y que se establezca una correcta conexión con la base de datos que ya se inicializó, primero hago que Django cree sus tablas internas con el fin de utilizar todos sus recursos.

```shell
python manage.py migrate
```

b) Ahora se generan modelos a partir de la base de datos del proyecto y se pone en core\models

```shell
python manage.py inspectdb > core/models.py
```

Para este paso, en core/models.py se debería estar mostrando un archivo con la “traducción” de cada una de las tablas de la base de datos, además de las tablas internas de Django. Algo parecido a lo siguiente:

```python
from django.db import models

# Nota: Los modelos de Django (auth_*, django_*) se dejan tal cual,
# ya que son gestionados internamente por el framework.

class Apartamentos(models.Model):
    id_apartamento = models.AutoField(primary_key=True)
    interior = models.IntegerField(blank=True, null=True)
    torre = models.IntegerField()
    numero = models.IntegerField()
    id_propietario = models.ForeignKey('Usuario', models.CASCADE, db_column='id_propietario',
    n_habitaciones = models.IntegerField()
    n_banos = models.IntegerField()
    area_total = models.DecimalField(max_digits=10, decimal_places=2)
    clasificacion = models.CharField(max_length=50, blank=True, null=True)
    deposito = models.IntegerField(blank=True, null=True)
    parqueadero = models.IntegerField(blank=True, null=True)

    class Meta:
        managed = False
        db_table = 'Apartamentos'
        unique_together = (('torre', 'numero'),)
```

Nota: Es importante que el atributo “managed” en class Meta de cada una de las tablas sea falso ya que esto le indica a Django que no cree que las tablas si no que utilice las que ya están.

Es importante revisar con cuidado cada uno de los modelos que genera Django ya que las políticas en cuanto a llaves primarias compuestas y atributos auto incrementados que no sean la llave primaria de la tabla pueden ser conceptos que desencadenan errores difíciles de depurar.

Nota: Se recomienda dividir los modelos en archivos individuales para una mejor organización y fácil modificación y mantenimiento.

13) Definir funciones de views para leer usuarios de la base de datos

a) Se recrea una función para leer todos los registros disponibles en la tabla de Usuarios. Como ya se tiene toda la configuración anterior, en este punto podemos hacer uso del ORM de Django. Una función que cumple con este objetivo se puede asemejar a la siguiente:

```Shell
from django.shortcuts import render
from django.http import HttpResponse
from .models import Usuario

# Clase Singleton para manejar acceso centralizado
class UsuarioManager:
    _instance = None

    def __new__(cls):
        if cls._instance is None:
            cls._instance = super().__new__(cls)
        return cls._instance

    def listar_todos(self):
        return Usuario.objects.all()

# Vista que utiliza el Singleton
def listar_usuarios(request):
    manager = UsuarioManager()
    usuarios = manager.listar_todos()
    contexto = {'usuarios': usuarios}
    return render(request, 'core/listar_usuarios.html', contexto)
```

14) Configuración del template para el HTML

a) Para dar funcionalidad a la app y con el fin de comprobar la comunicación entre la interfaz visual y la base de datos que reside en el contenedor de docker se pretende mostrar diferentes registros de Usuarios. Para esto es necesario crear un archivo HTML ubicado en Proyecto\Core\templates\core\listar_usuarios.html y configurarlo de la siguiente manera:

```HTML
<!DOCTYPE html>
```

```
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Lista de Usuarios</title>
</head>
<body>
    <h1>¡Conexión Exitosa! Usuarios en la Base de Datos</h1>

    <ul>
        {% for usuario in usuarios %}
            <li>
                <strong>Nombre:</strong> {{usuario.nombre}} |
                <strong>Cédula:</strong> {{usuario.cedula}} |
                <strong>Correo:</strong> {{usuario.correo}} |
                <strong>Permisos:</strong>
{{usuario.get_permission}}

            </li>
        {% empty %}
            <li>No hay usuarios registrados en la base de
datos.</li>
        {% endfor %}
    </ul>
</body>
</html>
```

15) Establecimiento de una URL en core

a) Se necesita conectar la función que se definió anteriormente con una URL para que esta última se pueda ejecutar en el momento del despliegue. La configuración se hace en el archivo ubicado en Proyecto\Core\urls.py y es tan sencillo como poner:

Shell

```
from django.urls import path
from . import views

urlpatterns = [
```

```
    path('usuarios/', views.listar_usuarios, name='lista_de_usuarios'),
]
```

16) Cambiar las URL propias del proyecto

a) Ahora se requiere conectar la URL del proyecto con la de la app.

```
Shell
urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('core.urls')),
]
```

17) Correr el servidor y redirigir para la vista de usuarios

a) Una vez con todo configurado se puede proceder a correr el servidor entrando a la carpeta donde se encuentre el manage.py y ejecutando el siguiente comando

```
Shell
python manage.py runserver
```

Esto debería mostrar en terminal algo como la siguiente:

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
October 24, 2025 - 23:57:57
Django version 5.2.7, using settings 'BuildingAppProject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Al dar click en el enlace al servidor y redirigir esta dirección a la de /api/usuarios/ se deberá mostrar una interfaz como la siguiente: