

## Seis. Bases de Datos.

Primero, necesitamos tener en cuenta algunas configuraciones iniciales: necesitamos indicarle a Django qué servidor de base de datos usar y cómo conectarse con el mismo.

Asumimos que haz configurado un servidor de base de datos, lo has activado, y has creado una base de datos en este (por ej. usando la sentencia CREATE DATABASE). SQLite es un caso especial; es este caso, no hay que crear una base de datos, porque SQLite usa un archivo autónomo sobre el sistema de archivos para guardar los datos.

Como con TEMPLATE\_DIRS en los capítulos anteriores, la configuración de la base de datos se encuentra en el archivo de configuración de Django, llamado, por omisión, settings.py. Edita este archivo y busca las opciones de la base de datos:

```
DATABASE_ENGINE = "
```

```
DATABASE_NAME = "
```

```
DATABASE_USER = "
```

```
DATABASE_PASSWORD = "
```

```
DATABASE_HOST = "
```

```
DATABASE_PORT = "
```

Aquí hay un resumen de cada propiedad.

- DATABASE\_ENGINE le indica a Django qué base de datos utilizar. Si usas una base de datos con Django, DATABASE\_ENGINE debe configurarse con un string de los mostrados en la Tabla:
- DATABASE\_NAME la indica a Django el nombre de tu base de datos. Si estás usando SQLite, especifica la ruta completo del sistema de archivos hacia el archivo de la base de datos (por ej. '/home/django/mydata.db').
- DATABASE\_USER le indica a Django cual es el nombre de usuario a usar cuando se conecte con tu base de datos. Si estás usando SQLite, deja este en blanco.
- DATABASE\_PASSWORD le indica a Django cual es la contraseña a utilizar cuando se conecte con tu base de datos. Si estás utilizando SQLite o tienes una contraseña vacía, deja este en blanco.
- DATABASE\_HOST le indica a Django cual es el host a usar cuando se conecta a tu base de datos. Si tu base de datos está sobre la misma computadora que la instalación de Django (o sea localhost), deja este en blanco. Si estás usando SQLite, deja este en blanco.

MySQL es un caso especial aquí. Si este valor comienza con una barra ('/') y estás usando MySQL, MySQL se conectará al socket especificado por medio de un socket Unix. Si estás utilizando MySQL y este valor no comienza con una barra, entonces este valor es asumido como el host. Ejemplo:

```
DATABASE_HOST = '/var/run/mysql'
```

DATABASE\_PORT le indica a Django qué puerto usar cuando se conecte a la base de datos. Si estás utilizando SQLite, deja este en blanco. En otro caso, si dejas este en blanco, el adaptador de base de datos subyacente usará el puerto por omisión acorde al servidor de base de datos. En la mayoría de los casos, el puerto por omisión está bien, por lo tanto puedes dejar este en blanco.

Una vez que hayas ingresado estas configuraciones, compruébalas. Primero, desde el directorio del proyecto que creaste en el Capítulo 2, ejecuta el comando `python manage.py shell`.

Notarás que comienza un intérprete interactivo de Python. Las apariencias pueden engañar. Hay una diferencia importante entre ejecutar el comando `python manage.py shell` dentro del directorio del proyecto de Django y el más genérico `python`. El último es el Python shell básico, pero el anterior le indica a Django cuáles archivos de configuración usar antes de comenzar el shell. Este es un requerimiento clave para hacer consultas a la base de datos: Django necesita saber cuáles son los archivos de configuraciones a usar para obtener la información de la conexión a la base de datos.

Detrás de escena, `python manage.py shell` simplemente asume que tu archivo de configuración está en el mismo directorio que `manage.py`. Hay otras maneras de indicarle a Django qué módulo de configuración usar, pero este subtítulo lo cubriremos luego. Por ahora, usa `python manage.py shell` cuando necesites hacer modificaciones específicas a Django.

Una vez que hayas entrado al shell, escribe estos comandos para probar la configuración de tu base de datos:

```
>>> from django.db import connection
```

```
>>> cursor = connection.cursor()
```

Si no sucede nada, entonces tu base de datos está configurada correctamente. De lo contrario revisa el mensaje de error para obtener un indicio sobre qué es lo que está mal:

- ❖ Error: You haven't set the DATABASE\_ENGINE setting yet.
  - Solución: Configura la variable DATABASE\_ENGINE con otra cosa que un string vacío.
- ❖ Error: Environment variable DJANGO\_SETTINGS\_MODULE is undefined.
  - Solución: Ejecuta el comando `python manage.py shell` en vez de `python`.
- ❖ Error: Error loading \_\_\_\_ module: No module named \_\_\_\_.
  - Solución: No tienes instalado el módulo apropiado para la base de datos especificada (por ej. `psycopg` o `MySQLdb`).
- ❖ Error: \_\_\_\_ isn't an available database backend.

- Solución: Configura la variable DATABASE\_ENGINE con un motor válido descrito previamente. ¿Habrás cometido un error de tipeo?
  
- ❖ Error: database \_\_\_\_\_ does not exist
  - Solución: Cambia la variable DATABASE\_NAME para que apunte a una base de datos existente, o ejecuta la sentencia CREATE DATABASE apropiada para crearla.
  
- ❖ Error: role \_\_\_\_\_ does not exist
  - Solución: Cambia la variable DATABASE\_USER para que apunte a un usuario que exista, o crea el usuario en tu base de datos.
  
- ❖ Error: could not connect to server
  - Solución: Asegúrate de que DATABASE\_HOST y DATABASE\_PORT estén configurados correctamente y que el servidor esté corriendo.

[https://librosweb.es/libro/django\\_1\\_0/capitulo\\_5/configuracion\\_de\\_la\\_base\\_de\\_datos.html](https://librosweb.es/libro/django_1_0/capitulo_5/configuracion_de_la_base_de_datos.html)