Django — Conecta tu proyecto con la base de datos MySQL

Dante Maximilano Flores Moreno Follow
Nov 2, 2018 · 7 min read

Django es un framework orientado al desarrollo de aplicaciones web en Python. Generalmente cuando queremos aprender a utilizar esta herramienta, los tutoriales están diseñados para trabajar con la base de datos de SQLite de manera predeterminada. SQLite es un gestor de base de datos reconocido por su buen rendimiento, portabilidad y fácil instalación (literalmente no se necesita instalar). Sin embargo, SQLite sólo es considerado a la hora de realizar proyectos con bases de datos sencillas y pequeñas, que no requieran soportar una gran cantidad de peticiones.

Cuando me encontraba desarrollando una aplicación en Django, junto con mis compañeros de proyecto nos dimos que era necesario utilizar una base de datos que pudiera cubrir nuestras necesidades, las cuales SQLite no cubría.

Es por eso que nos decidimos a utilizar MySQL, esto debido a muchos aspectos tales como su capacidad de manejar grandes bases de datos (hasta 8 TB), su sistema de seguridad encriptado para acceder a ellas, su facilidad de instalación en distintos sistemas operativos, y por supuesto, es gratuito.

Una vez decidida la base de datos que utilizaremos, me percaté de que leer la documentación del sitio puede resultar un poco confuso y complicado de entender. Para conectar un proyecto de Django con una base de datos MySQL es necesario tener instaladas una serie de librerías, en la documentación mencionan que se requiere instalar *MySQL connector/python*. Yo tardé mucho en entender cómo instalarlo y realizar las configuraciones necesarias con esa herramienta, así que decidí buscar otras opciones en distintos foros y encontré una librería de Python llamada *pymysql* que me funcionó como lo necesitaba.

Es por eso que en éste artículo les voy a mostrar cómo conectar su aplicación de Django con MySQL de manera local, paso a paso, utilizando la librería de *pymysql*.

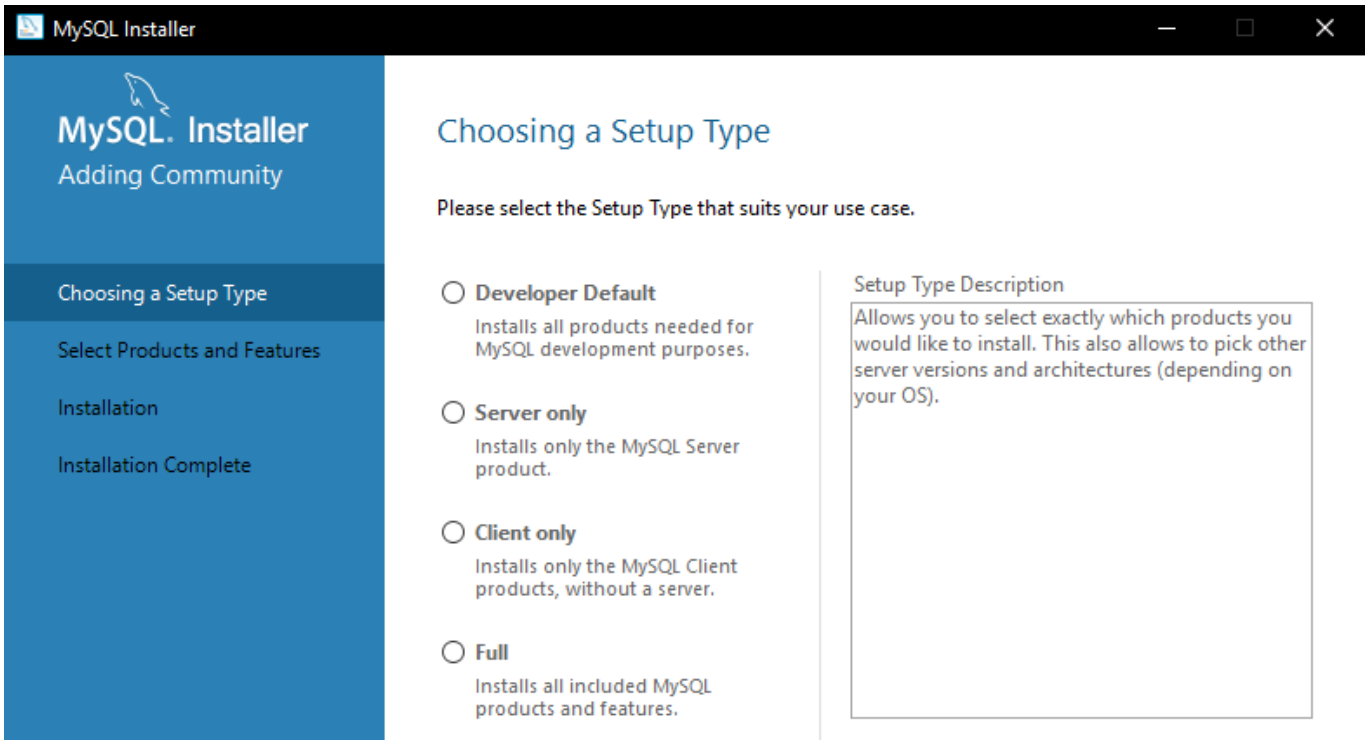
Cabe aclarar que en éste artículo se trabajará con las siguiente configuración. De cualquier manera les dejo los links de dónde pueden descargar e instalar las dependencias:

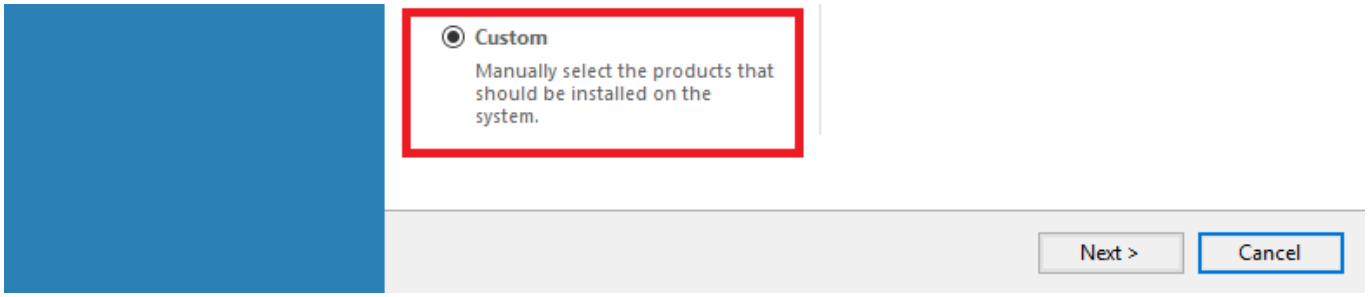
- Django 2.1.2 (instalación Windows)
- Python 3.6.6 (instalación)
- MySQL 8.0.13 (instalación)
- SO: Windows 10

PASO 1.- Descargar e instalar MySQL en tu equipo

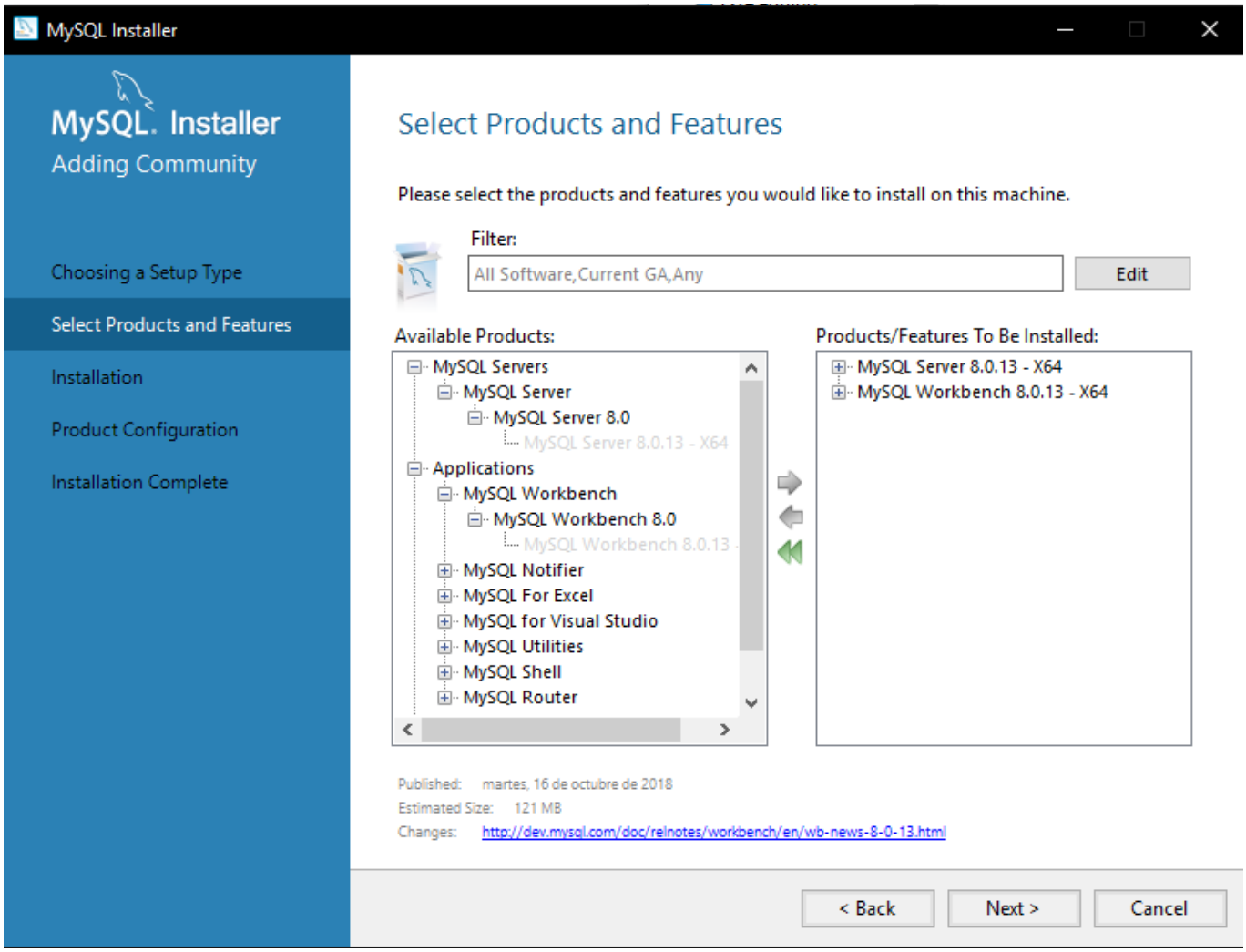
Para conectar tu base de datos a tu aplicación, es necesario tener antes montado en tu equipo MySQL server. Para hacer ésto sólo es necesario descargar el instalador que se encuentra dando clic aquí.

Cuando abramos el instalador, nos va a preguntar el tipo de configuración que queremos instalar. Nosotros sólo ocuparemos MySQL server y el Workbench, así que vamos a seleccionar la opción de *custom* y daremos en *next*:





Una vez hecho esto nos pedirá los productos que necesitemos instalar, nosotros seleccionaremos sólo el SQL Server y el Workbench:

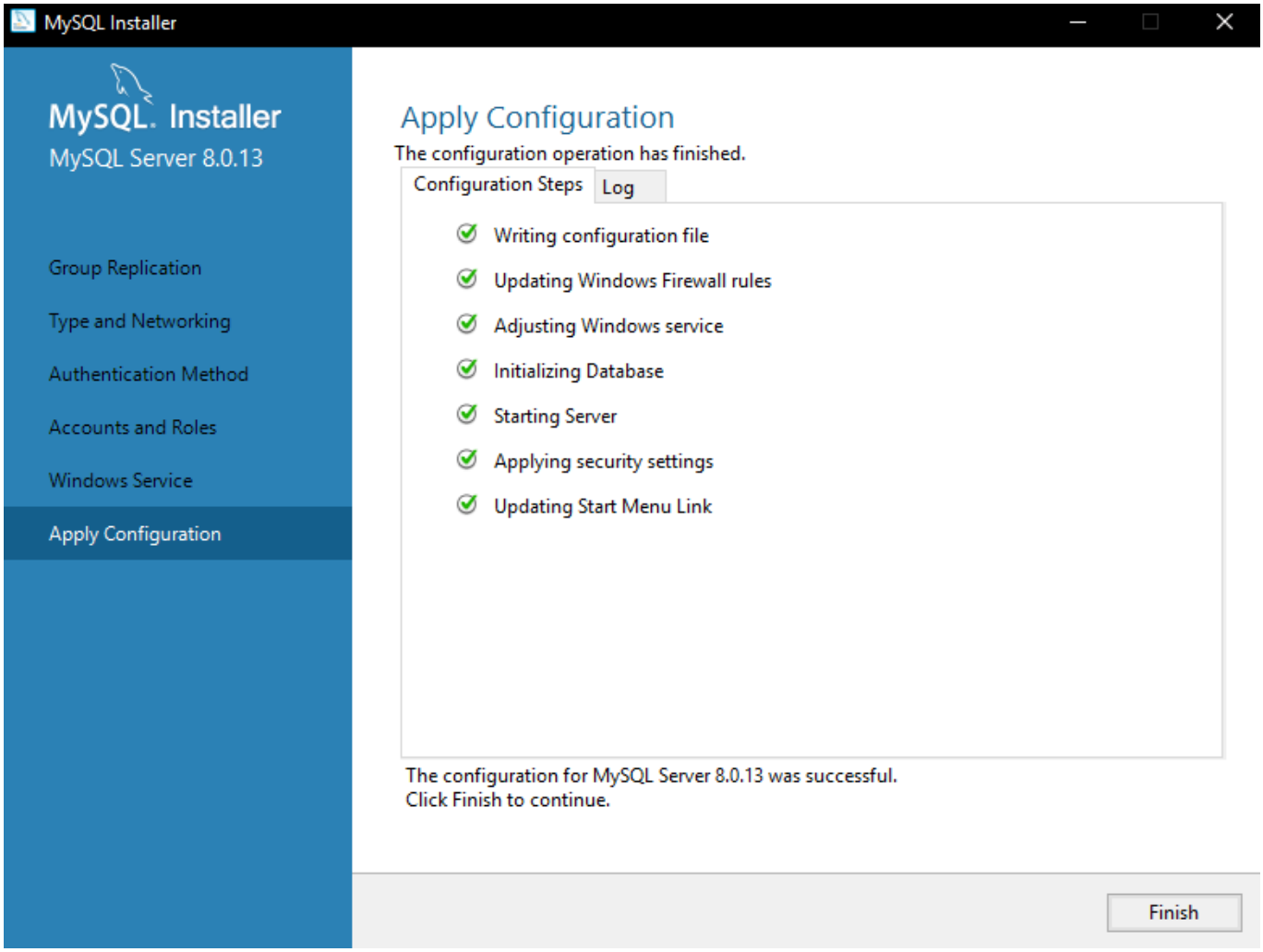


Cuando seleccionemos los productos y demos clic en *Next*, los productos se comenzarán a instalar y luego nos pedirá algunos parámetros de configuración para los productos.

Vamos a dejar tal cual se encuentran en un principio las configuraciones y sólo presionaremos *Next* en *Group Replication* y *Type and Networking*. Para cuando lleguemos a una sección de autenticación, nos va a solicitar una contraseña; usted podrá poner la

que guste. **Es importante saber la contraseña que se va a insertar porque será utilizada más adelante.**

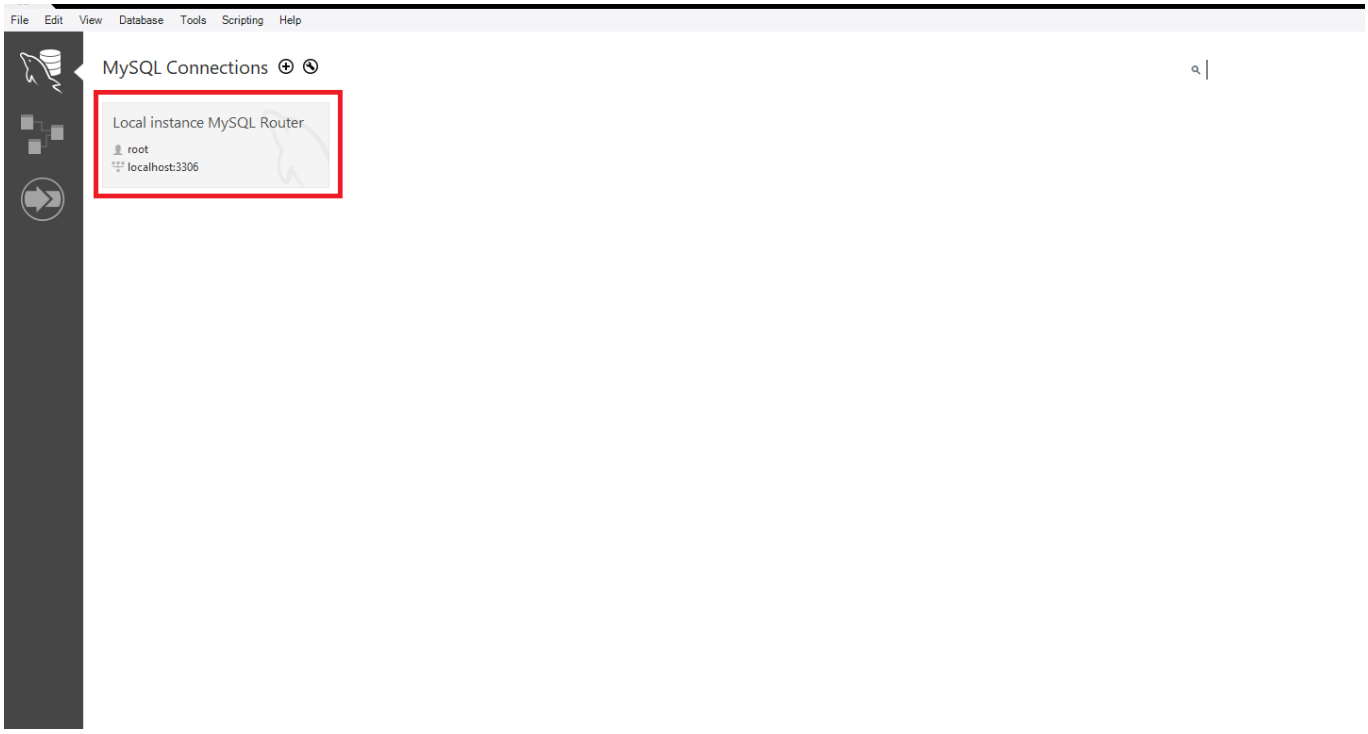
Después de poner la contraseña, avanzaremos por la configuración, dejando las opciones predeterminadas y finalmente daremos clic en *Execute*. Si al final tienes una pantalla como la que se muestra a continuación, significa que ya todo quedó correctamente instalado y configurado:



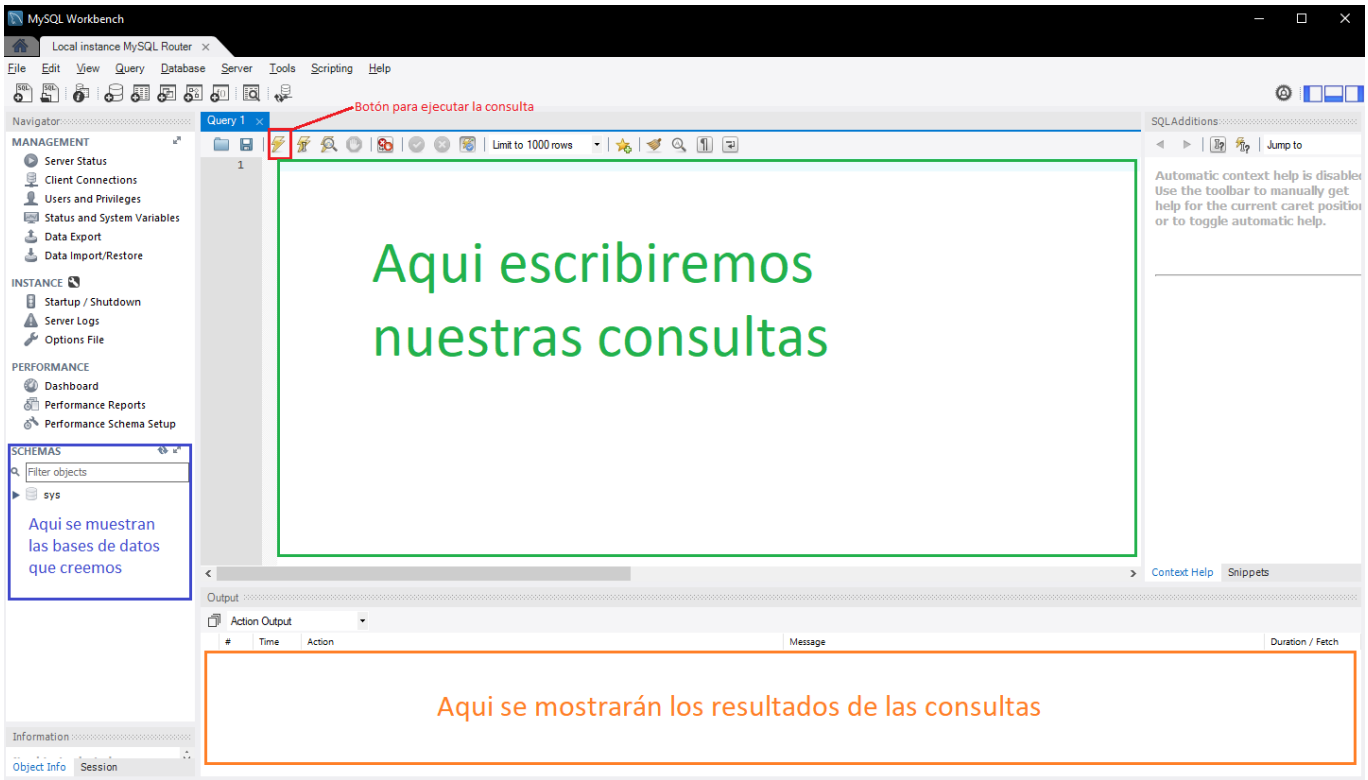
Paso 2.- Crear la Base de Datos y un usuario para acceder a ella

Ahora vamos a crear una base de datos que será la que se conecte con nuestra aplicación de Django. Para esto iniciaremos la aplicación de *Workbench* que instalamos en el paso 1.

Se nos abrirá una ventana como la de la imagen y daremos clic en *Local instance* para conectarnos a la base de datos con *Workbench*.



Cuando demos clic vamos a insertar la contraseña que ingresamos en el paso 1, y de eso tendremos un editor de texto para insertar nuestras consultas como el que se muestra abajo, en la imagen pondré los componentes que debemos conocer y ubicar para crear nuestra base de datos:



Ahora procederemos a escribir una consulta en el recuadro verde para crear la base de datos:

```
CREATE DATABASE nombreDB CHARACTER SET utf8mb4;
```

Donde *nombreDB* es el nombre que se le dará a la base de datos.

Al ejecutar esa consulta podremos actualizar el recuadro azul y veremos la nueva base de datos creada.

Para poder acceder a ella, será necesario crear un usuario con contraseña que tenga los privilegios necesarios. Para ello ejecutaremos la siguiente consulta:

```
CREATE USER nombreusuario@localhost IDENTIFIED BY 'pass';

GRANT ALL PRIVILEGES ON nombreDB.* TO nombreusuario@localhost;
FLUSH PRIVILEGES;
```

Dónde *nombreusuario* es el nombre de usuario que se creará con su respectiva contraseña que en este caso es *pass*.

Se recomienda poner una contraseña más segura para evitar problemas de seguridad

Paso 3.- Instalar el cliente de MySQL para Python

Ahora que ya instalamos y creamos una base de datos de MySQL de manera local, sigue instalar un cliente de para poder conectarnos con ella mediante Django que trabaja con Python.

Para esto basta con ejecutar el siguiente comando en el shell de Windows:

```
...\> pip install pymysql
```

Cabe aclarar que es recomendable ejecutar este comando dentro de un ambiente virtual dónde se haya instalado Django.

Paso 4.- Importar el cliente en nuestra aplicación de Django

Ahora que ya tenemos todo instalado y configurado, vamos a importar el cliente de *pymysql* a la aplicación de django.

Para efectos de presentación, yo creé un proyecto llamado MyProject que contiene una aplicación app_tutorial. La estructura de los archivos es la siguiente:

```
MyProject/
    manage.py
    MyProject/
        __init__.py
        settings.py
        urls.py
        wsgi.py
    app_tutorial/
        migrations/
        __init__.py
        admin.py
        apps.py
        models.py
        tests.py
        views.py
```

Para importar el cliente en todo el proyecto, lo que haremos será abrir el documento *MyProject/MyProject/__init__.py* y escribir los siguiente:

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

Paso 5.- Configurar la conexión de la aplicación con la base de datos

Ahora que somos capaces de conectarnos con una base de datos, vamos a configurar los parámetros para conectarnos con la base de datos que creamos en el *Workbench* de MySQL.

Para hacer esto abriremos el documento *MyProject/MyProject/settings.py* y buscaremos el siguiente fragmento:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Y lo vamos a modificar para que tenga la información de la base de datos que queremos conectar. Al final debería verse como lo que se muestra abajo:

```
DATABASES = {
        'default': {
            'ENGINE': 'django.db.backends.mysql',
            'NAME': 'nombreDB',
            'USER': 'nombreusuario',
            'PASSWORD': 'pass',
            'HOST': 'localhost',
            'PORT': '3306',
        }
    }
```

Paso 6.- Migrar los modelos a la base de datos

Ya tenemos todo configurado y listo para conectarse, ahora sólo queda realizar las migraciones para que los modelos de la aplicación de Django estén en la nueva base de datos.

Para mostrar que todo funciona de manera correcta, en el proyecto creé una pequeña aplicación que se llama *app_tutorial,* la cual en el archivo *MyProject/app_tutorial/models.py* está el siguiente modelo:

```
from django.db import models

# Create your models here.
class Tabla_test(models.Model):
    columna_uno = models.CharField(max_length=200)
    columna_dos = models.CharField(max_length=200)
```

```
    columna_tres = models.CharField(max_length=200)
    columna_cuatro = models.CharField(max_length=200)
```

Ahora realizaremos las migraciones, insertando en el prompt de windows el siguiente comando:

```
...\MyProject\> python manage.py makemigrations
```

Seguido de este:

```
...\MyProject\> python manage.py migrate
```

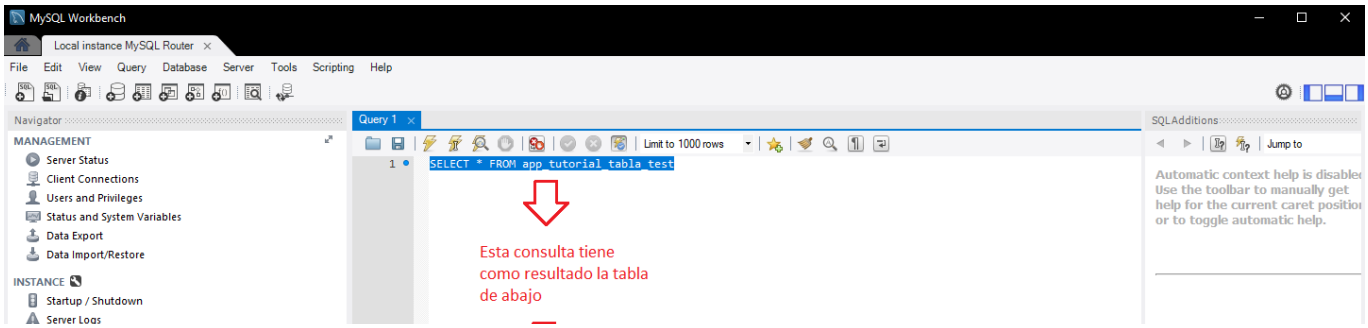
Paso 7.- Comprobar que todo funcionó correctamente

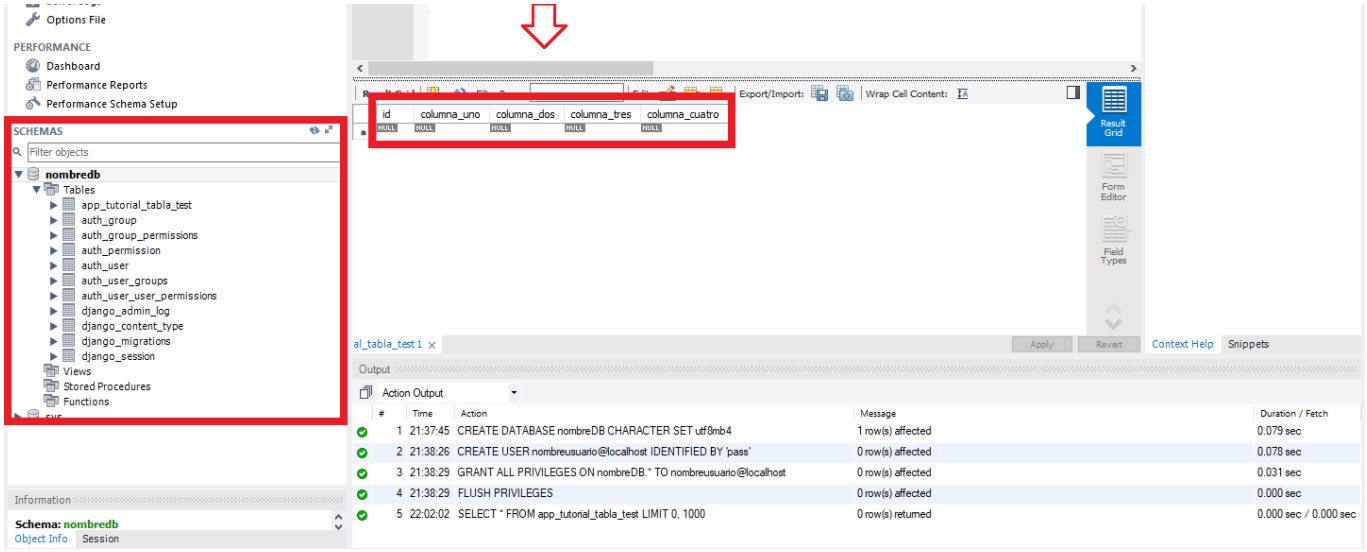
Si todos los pasos fueron seguidos correctamente como se mencionaron el artículo, los modelos de todas las aplicaciones que tienes en tu proyecto ya se debieron haber guardado en las base de datos que creamos.

Si quieres comprobar que se encuentran en donde deben estar, podemos abrir la aplicación de *Workbench* de MySQL. Y veremos en *schemas* que ahí se encuentran las tablas con los nombre de aplicación, seguido del nombre del modelo que creaste. Incluso si quieres verificar la integridad de las tablas, puedes ejecutar la siguiente consulta:

```
SELECT * FROM app_tutorial_tabla_test
```

El *Workbench* se vería como la imagen de abajo:





¡Y LISTO!

Ya tienes tu aplicación de Django conectada con tu base de datos MySQL. Recuerda que cada vez que hagas modificaciones en los archivos models.py realices las migraciones correspondientes para que las actualizaciones se vean reflejadas en tu base de datos. También recomiendo que sólo utilicen el Workbench para consultar la información de la base de datos. Si realizas cambios directamente desde el Workbench, puede que causes algún conflicto con tu aplicación.

Cualquier duda pueden hacérmela saber en los comentarios.

Django Database Python MySQL Base De Datos