TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE

- Django y su integración a base de datos.
- Las bases de datos soportadas por Django.
- Soporte para bases NoSql.
- Querys SQL en Django.

DJANGO Y SU INTEGRACIÓN A BASE DE DATOS

La capa de base de datos de Django proporciona varias formas para aprovechar al máximo sus bases de datos. Django utiliza SQLite 3 para el almacenamiento de datos, pero permite cambiar a otros motores fácilmente.

Éste obtiene la estructura, acceso y control de los datos de una aplicación a través de su ORM (Object Relational Mapper), lo que significa que no importa qué motor de base de datos esté usando, el mismo código seguirá funcionando.

Todo se define dentro del archivo `settings.py` de nuestro proyecto dentro de la variable `DATABASES`. El archivo `settings.py` muestra las aplicaciones instaladas, y las configuraciones de las bases de datos:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': 'cu07',
        'USER': 'user07',
        'PASSWORD': 'user07_password',
        'HOST': '',
        'PORT': '',
    }
}
```

`DATABASES` es donde se indicará que definiremos una base de datos. Dentro de este objeto, tendremos el `default`, que contiene toda la configuración clave de la base de datos. En este apartado se define el motor, se agrega un usuario y contraseña en caso de ser necesario.

LAS BASES DE DATOS SOPORTADAS POR DJANGO

Django soporta oficialmente las siguientes bases de datos:

- **Postgresql:** Django es compatible con PostgreSQL 10, y superior. Se requiere psycopg2 2.8.4 o superior, aunque se recomienda la última versión.
- **Mysql:** Django es compatible con MySQL 5.7, y superior.
- **Oracle:** Django es compatible con las versiones 19c y posteriores de Oracle Database Server. Se requiere la versión 7.0, o superior, del controlador cx_Oracle Python.
- **SQLite:** Django es compatible con SQLite 3.9.0, y versiones posteriores.

SOPORTE PARA BASES NOSQL

En el caso de los manejadores de bases de datos NoSql, el más utilizado en el entorno Django es MongoDB.

El método para conectar a esta base de datos es a través de MongoEngine, un mapeador de objetos de documentos (similar al ORM, pero para bases de datos de documentos) que permite trabajar con MongoDB desde Python.

Los requisitos son: MongoDB superior a 3.4, y pymongo mayor a 3.4.

En este caso, en el archivo `settings.py` se comenta la sección BASES DE DATOS como se muestra en el siguiente fragmento de código, y se le agregan las líneas resaltadas.

```
import mongoengine
mongoengine.connect(db=DATABASE_NAME,host=DATABASE_HOST,
username=USERNAME, password=PASSWORD)
#DATABASES = {
# 'default': {
# 'ENGINE': 'djongo', #'django.db.backends.sqlite3',
# 'NAME': 'blogs', # DB name
# 'USER': 'root', # DB User name <optional>
# }
#}
```

QUERYS SQL EN DJANGO

Django ofrece dos formas de realizar consultas de SQL:

- Puede utilizar `Manager.raw()` para realizar consultas sin procesar, y devolver instancias de modelo. En este caso, `Manager. raw( raw_query, params=(), translations=None)`

  Este método toma una consulta SQL sin procesar, la ejecuta, y devuelve una instancia `django.db.models.query.RawQuerySet`. Esta instancia `RawQuerySet` se puede iterar como un `QuerySet` para proporcionar instancias de objetos.

- Puede prescindir de la capa de modelo por completo, y ejecutar SQL personalizado directamente. En estos casos, se accede a la base de datos por la capa del modelo para ejecutar las sentencias SQL. El objeto `django.db.connection` representa la conexión de base de datos predeterminada. Para usar la conexión de la base de datos, se debe llamar `connection.cursor()` para obtener un objeto de cursor. Luego, se ejecuta el SQL para devolver las filas resultantes con: `.cursor.execute(sql, [params])cursor.fetchone()cursor.fetchall()`.