Autor: Eduardo Escárez

- ¿Cuáles son las bases de datos soportadas por Django y en qué se diferencian?

Los sistemas de gestión de base de datos soportados son:

* Oracle Database: Es un gestor de base de datos desarrollado por Oracle y se diferencia por su versatilidad y escalabilidad para ser utilizado en empresas de cualquier tamaño, ofreciendo versiones adaptadas para cada tipo de cliente; una gran capacidad de procesamiento de datos, y alta disponibilidad y seguridad.
* SQLite: Es un gestor de base de datos de código libre de dominio público. Sus mayores diferencias son, funciona mediante un sistema de archivos locales, el servidor puede ser integrado en otras aplicaciones al ser compacto, no requiere de configuración inicial ni tener iniciar un servicio para utilizarlo.
* PostgreSQL: Es un gestor de base de datos de código libre. Sus mayores diferencias son, es un sistema de gestión que adhiere estrictamente los estándares SQL, es extensible sobre la marcha, y es escalable mediante partición de tablas en diversos servidores para mejor rendimiento.
* MySQL: Es un gestor de base de datos de código libre. Sus mayores ventajas son, la facilidad de uso respecto a otros gestores de base de datos, su popularidad que permite disponer de diversas herramientas para usarlo (MySQL Workbench, DBeaver, phpMyAdmin), ser considerado un gestor de base de datos rápido.
* MariaDB: Es un gestor de base de datos de código libre, siendo inicialmente un fork de MySQL. Sus mayores ventajas son, soporte de la mayoría de ecosistema de aplicaciones que funciona con MySQL, mejoras de rendimiento y características adicionales respecto de MySQL, licenciamiento único mediante la licencia GPL a diferencia del licenciamiento dual de MySQL.

- ¿Qué es una migración en Django y para qué se utiliza?

La migración es una herramienta de Django que permite al desarrollador crear un modelo de datos y que este sea convertido en una estructura SQL para ser ejecutada en una base de datos.

Se utiliza para garantizar el mantenimiento y consistencia de la base de datos, ya que como la migración es creada por Django en base al modelo provisto, es capaz de verificar que la estructura sea válida antes de ser ejecutada en la base de datos. También, como la realización de las tareas es ejecutada por Django sin necesidad de escribir manualmente código SQL, se evitan los errores al ejecutar esas tareas y que la base de datos tenga una estructura según los requerimientos de los modelos.

- ¿Cuál es la diferencia entre usar consultas SQL y consultas ORM en Django?

La diferencia es al usar consultas ORM las consultas son más legibles y fáciles de escribir, ya que son realizadas en Python con el conjunto de funciones que provee el ORM.

También son más seguras, ya que al abstraer las funciones SQL a otro tipo de funciones, se elimina el riesgo de ataques como por inyección SQL, que puede provocar filtraciones o pérdidas de datos.

- ¿Cómo se instalan los paquetes de base de datos en Django y cuál es su importancia?

La instalación de paquetes de base de datos se realiza mediante el gestor de paquetes pip y existen alternativas como:

* SQLite: Django ya incluye un paquete para gestión de esta base de datos
* PostgreSQL: Se dispone del paquete pyscopg2
* MySQL: Se dispone del paquete mysqlclient
* MariaDB: Se dispone del paquete mariadb
* Oracle: Se dispone del paquete django-oracle

Su importancia es que actúan como intermediarios entre Django y el gestor de base de datos, lo que permite no solo la adaptación de las consultas ORM al equivalente SQL según el gestor, sino adicionalmente usar las características adicionales que cada servidor provee.

- ¿Qué ventajas ofrece Django como ORM para la integración con una base de datos?

Ofrece como ventajas:

* Abstracción de la base datos: Uno puede utilizar la base de datos mediante las funciones incorporadas en el ORM en vez de utilizar SQL directamente.
* Independencia del gestor de base de datos: Uno puede usar los mismos modelos sin grandes cambios independiente de gestor de base de datos donde se realicen las operaciones como MySQL, PostgreSQL, SQLite, u otros.
* Consultas SQL optimizadas: El ORM permite generar las consultas SQL necesarias optimizando tanto las consultas en si como respecto del gestor de base de datos utilizado.