**- ¿Qué es PostgreSql y cuáles son sus ventajas en comparación con otras bases de datos relacionales?**

PostgreSQL es un sistema de gestión de bases de datos relacionales de código abierto que ofrece un enfoque robusto, escalable y confiable para el almacenamiento y manipulación de datos. Sus ventajas en comparación con otras bases de datos relacionales incluyen:

* Funcionalidad avanzada: PostgreSQL proporciona una amplia gama de características avanzadas, como soporte para tipos de datos complejos, consultas complejas, funciones almacenadas, disparadores, vistas y más. Esto permite una mayor flexibilidad y capacidad para manejar escenarios de datos complejos.
* Rendimiento y escalabilidad: PostgreSQL está diseñado para ofrecer un buen rendimiento incluso en cargas de trabajo intensivas. Ofrece opciones de configuración y optimización para adaptarse a diferentes requisitos de rendimiento y escala.
* Confiabilidad y durabilidad: PostgreSQL se destaca por su capacidad de recuperación en caso de fallas, gracias a su registro de transacciones y soporte para transacciones ACID (atómicas, consistentes, aisladas y duraderas). También ofrece mecanismos de respaldo y recuperación robustos.
* Comunidad activa y soporte: PostgreSQL cuenta con una comunidad de usuarios y desarrolladores activa y dedicada, lo que garantiza una amplia documentación, actualizaciones regulares de versiones y un soporte sólido.

**- ¿Cuál es la diferencia entre una llave primaria simple y una llave primaria compuesta en Django?**

Una llave primaria simple en Django es una columna única en una tabla que se utiliza para identificar de manera exclusiva cada registro en esa tabla. Por otro lado, una llave primaria compuesta es una combinación de dos o más columnas que se utilizan conjuntamente para identificar de manera exclusiva cada registro en una tabla. Mientras que una llave primaria simple utiliza una única columna para la identificación, una llave primaria compuesta utiliza múltiples columnas.

**- ¿Cuál es el propósito de las operaciones CRUD en el desarrollo de aplicaciones web y cómo se**

**implementan en Django?**

Las operaciones CRUD son una parte fundamental del desarrollo de aplicaciones web y se refieren a las acciones básicas realizadas sobre los datos: Crear (Create), Leer (Read), Actualizar (Update) y Borrar (Delete). Estas operaciones permiten a las aplicaciones interactuar con una base de datos y gestionar la información.

En Django, estas operaciones CRUD se implementan utilizando el ORM (Object-Relational Mapping) incorporado. El ORM de Django mapea los modelos de la aplicación a tablas en la base de datos y proporciona métodos y funciones para realizar operaciones CRUD de manera sencilla. Por ejemplo, para crear un nuevo registro en la base de datos, se puede crear una instancia de un modelo y llamar al método **save()**. Para leer registros, se pueden utilizar métodos como **filter()** o **get()**. Para actualizar registros, se pueden modificar los atributos del modelo y llamar a **save()**. Para eliminar registros, se puede llamar al método **delete()**.

**- ¿Qué herramientas o componentes adicionales de Django utilizarías para mejorar el rendimiento y la**

**seguridad de una aplicación web?**

* Caché: Django ofrece soporte para diferentes sistemas de caché, como Memcached o Redis, que pueden ayudar a mejorar el rendimiento al almacenar en memoria resultados de consultas costosas o datos procesados.
* CDN (Content Delivery Network): Utilizar un CDN para servir archivos estáticos, como imágenes o hojas de estilo, puede mejorar el rendimiento al distribuir el contenido en servidores ubicados estratégicamente en diferentes regiones geográficas.
* Django Middleware: Se pueden implementar middlewares personalizados para agregar capas adicionales de seguridad, como protección contra ataques CSRF (Cross-Site Request Forgery) o XSS (Cross-Site Scripting).
* Uso de índices y optimización de consultas: Optimizar las consultas de la base de datos mediante la creación de índices adecuados en las columnas utilizadas con frecuencia en las consultas.
* Medidas de seguridad adicionales: Implementar autenticación y autorización adecuadas, utilizar protección contra inyección de SQL y asegurarse de que los formularios estén validados correctamente para prevenir ataques de seguridad comunes.
* Uso de herramientas de monitoreo y rendimiento: Utilizar herramientas como Django Debug Toolbar o Django Silk para realizar un seguimiento del rendimiento de la aplicación, identificar cuellos de botella y realizar ajustes necesarios.

<https://github.com/MochitoSpooky/m7-individual-2.git>