1. ¿Por qué eligieron este sistema de gestión de bases de datos (DBMS)? ¿Qué ventajas y desventajas tiene en comparación con otros?

Elegimos PostgreSQL porque es un sistema robusto, de código abierto y altamente compatible con estándares SQL. Algunas ventajas que nos hicieron decidirnos fueron:

* Soporte para ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo que garantiza la integridad de los datos.
* Compatibilidad con JSON y datos espaciales (útil si en el futuro queremos integrar datos más complejos).
* Alto rendimiento con consultas SQL complejasgracias a su optimización de índices y transacciones.
* Escalabilidad y seguridad, con soporte para múltiples conexiones y permisos avanzados.

Sin embargo, también identificamos algunas desventajas en comparación con otras opciones como MySQL o MongoDB:

* Curva de aprendizaje más alta para quienes no están acostumbrados a su sintaxis y configuraciones.
* Uso de memoria y recursos mayor que MySQL en ciertos escenarios.
* Configuración inicial más compleja, especialmente en entornos de producción.

Aun así, elegimos PostgreSQL porque ofrece mayor flexibilidad y potencia para gestionar relaciones entre tablas y datos estructurados.

2. ¿Qué estándares o criterios usaron para diseñar su base de datos?

Para diseñar nuestra base de datos seguimos estos criterios:

* Normalización hasta la 3FN para evitar redundancias y mejorar la integridad de los datos.
* Uso de claves primarias y foráneas para garantizar relaciones entre tablas bien definidas.
* Restricciones `CHECK`, `NOT NULL` y `DEFAULT`\*\* para evitar errores en los datos ingresados.
* Índices en columnas clave (`correo\_electronico`, `estado\_pedido`, etc.) para optimizar consultas.
* ON DELETE CASCADE / SET NULL en algunas relaciones para manejar la eliminación de datos dependientes.

También nos basamos en la metodología entidad-relación y el modelo relacional para asegurarnos de que el diseño se ajustara a las necesidades del proyecto.

3. ¿Cuáles son las entidades más importantes del modelo y por qué?

Las entidades más importantes en nuestra base de datos son:

* `usuario`: Es la entidad principal porque representa a los clientes y diseñadores.
* `pedido\_personalizado`: Centraliza la información sobre cada solicitud de ropa personalizada.
* `plantilla\_prenda` y `material`: Son clave porque definen los productos disponibles.
* `transaccion\_pago`: Es fundamental para gestionar los pagos y facturación.

Estas entidades reflejan el núcleo del negocio: crear ropa personalizada basada en medidas y preferencias de los clientes.

4. ¿Cómo aplicaron las técnicas de normalización en su diseño? ¿Qué problemas evitaron gracias a esto?

Normalizamos la base de datos hasta la Tercera Forma Normal (3FN), lo que nos permitió:

* Eliminar redundancias (por ejemplo, separando `usuario` y `perfil\_medidas`).
* Evitar anomalías en actualización e inserción (no repetimos datos innecesarios).
* Mantener la consistencia en los datos al estructurarlos correctamente.

Si no hubiéramos normalizado, podríamos haber tenido datos repetidos y dificultad para actualizar información sin errores.

5. ¿Cómo definieron restricciones y valores por defecto para garantizar la integridad de los datos?

Usamos varias restricciones en el diseño:

* PRIMARY KEY` y `FOREIGN KEY` para garantizar relaciones correctas entre tablas.
* NOT NULL` en columnas clave como `nombre`, `correo\_electronico`, `estado\_pedido`, etc.
* CHECK` para restringir valores permitidos (ejemplo: `estado\_pedido` solo puede ser 'pendiente', 'diseño', 'producción', etc.).
* DEFAULT` para valores automáticos, como ‘fecha\_creacion’ con ‘CURRENT\_TIMESTAMP’.
* UNIQUE en ‘correo\_electronico’ para evitar registros duplicados.

Esto ayuda a prevenir errores humanos y asegurar que los datos sean válidos desde el principio.

6. ¿Cómo abordaron los cambios en la estructura de la base de datos?

Durante el desarrollo hicimos varios ajustes en la estructura:

* Usamos ‘ALTER TABLE’ para agregar o modificar columnas cuando fue necesario.
* Probamos en un entorno de prueba antes de aplicar cambios en la base de datos principal.
* Documentamos las modificaciones para que el equipo estuviera al tanto de las actualizaciones.
* Consideramos migraciones en caso de cambios importantes.

Un cambio importante fue la adición de la tabla ‘historial\_estado’, que no estaba en el diseño inicial, pero nos permitió llevar un mejor seguimiento de los pedidos.

7. ¿Cómo seleccionaron los datos de prueba para garantizar que el diseño es funcional?

Para probar la base de datos, seguimos estos pasos:

* Creamos datos de prueba realistas, incluyendo usuarios, pedidos y transacciones.
* Probamos diferentes estados de pedidos para verificar que las restricciones funcionaban.
* Ejecutamos consultas para verificar integridad (por ejemplo, contar pedidos por usuario).
* Evaluamos índices con consultas grandes para ver mejoras en rendimiento.

Gracias a esto, pudimos detectar errores como valores incorrectos en ‘estado\_pedido’ antes de implementar el sistema final.

8. ¿Cuál fue tu contribución específica en el desarrollo del proyecto? ¿Cómo se organizó el trabajo en el equipo?

Mi contribución principal fue realizar el DDL según el diagrama entidad relación propuesto, este bajo las reglas de negocio y datos de prueba vistos previamente.

El equipo se organizó dividiendo tareas simples, mientras que las complejas fueron hechas en conjunto por si existían dudas, otro se encargó de realizar las reglas de negocio y otro compañero de la realización del diagrama entidad-relacion, así mismo otro compañero se encargó de hacer el versionamiento del código hecho por mí; otro compañero se encargó de dar datos de prueba.

¿Sientes que trabajaste equitativamente en comparación con tus compañeros? ¿Qué hubieras hecho diferente en este proyecto?

Sí, siento que asumí un poco más de trabajo en la parte de la base de datos. Sin embargo, esto me permitió aprender más sobre optimización y normalización.

Si pudiera cambiar algo:

* Mejoraría la documentación desde el inicio para evitar confusiones en el equipo.
* Usaría más pruebas automatizadas para verificar la integridad de los datos.
* Organizaría mejor el trabajo para que todos aportaran más en la parte de la base de datos.