Reflexión Individual

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 nuestro proyecto requería un equilibrio entre escalabilidad, robustez y flexibilidad. A diferencia de alternativas como MySQL o SQLite, PostgreSQL ofrece soporte avanzado para transacciones complejas, tipos de datos personalizados (como JSON) y una integridad referencial más estricta, lo cual era crucial para gestionar pedidos personalizados con múltiples relaciones (usuarios, plantillas, materiales). Además, al ser open-source, evitamos costos de licencia sin sacrificar capacidades empresariales.

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

El diseño se basó en tres pilares:

- Normalización (hasta 3FN): Para eliminar redundancias (ej: separar material y plantilla_prenda en tablas independientes).
- Reglas de negocio: Cada tabla reflejaba una funcionalidad clave (ej: historial_estado para rastrear pedidos).
- Claridad: Nombres descriptivos (ej: pedido_personalizado en lugar de orden) y documentación en el script SQL.

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

Las entidades críticas fueron:

a) pedido_personalizado: El núcleo del sistema, donde se vinculan usuarios, plantillas y materiales.

b) usuario: Diferenciaba roles (cliente, diseñador, administrador) con permisos implícitos.

c) plantilla_prenda: Permitía reutilizar diseños base para múltiples pedidos, evitando duplicar datos.

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

- 1FN: Campos atómicos (ej: color en pedido_personalizado es un valor único).

- 2FN y 3FN: Eliminamos dependencias parciales/transitivas (ej: material_id solo depende de material, no de pedido).

Beneficio: Evitamos anomalías en inserciones/actualizaciones (ej: si un material cambiaba su nombre, no había que editarlo en múltiples pedidos).

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

Implementamos:

- Constraints: CHECK para validar roles o estados (estado_pedido IN ('pendiente', 'completado')).
- Valores por defecto: fecha_creacion TIMESTAMP DEFAULT CURRENT_TIMESTAMP para auditoría automática.
- Claves foráneas con acciones claras: ON DELETE CASCADE en perfil_medidas para borrar datos huérfanos.

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

Los ajustes fueron mínimos (ej: añadir índices para optimizar consultas frecuentes). Usamos ALTER TABLE para modificaciones y documentamos todo en Git.

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

Seleccionamos datos que:

- Cubrieran todos los escenarios (ej: pedidos en distintos estados).
- Validaran relaciones (ej: un pedido con material y plantilla existentes).
- Incluyeran casos límite (ej: usuario sin perfil de medidas)

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

Rol	Tarea
Compañero 1	Planteo de las reglas de negocio
Compañero 2	Modelo ER y diseño inicial.
Compañero 3	Script SQL de creación de tablas y ajustes.
Yo	Inserción de datos de prueba y apoyo en elección de DBMS.
Compañero 4	Repositorio Git y documentación.

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

Sí, cada uno aportó en el área que se asignó.
¿Qué cambiaría?:

- Más discusión grupal en decisiones clave (ej: elección de tipos de datos).
- Pruebas tempranas de rendimiento con datos masivos.