1. ¿Cuál fue el aporte técnico de cada miembro del equipo?

Osman de León // Creación de CRUDS

Olivier Viau // Datos de prueba, solución de errores

Milton Polanco //Creación de CRUDS

Adrián Gonzáles //Planificador de base de datos

2. ¿Qué decisiones estructurales se tomaron en el modelo de datos y por qué?

- **Entidades fuertes**: Facultad → Departamento → Carrera (relación 1:N)
- **Entidades débiles**: Curso → Horario (dependencia identificativa)
- **Motivación**: Modelar fielmente la estructura universitaria real donde:
    - Cada facultad contiene departamentos
    - Las carreras pertenecen a una facultad
    - Los cursos existen sólo dentro de carreras

3. ¿Qué criterios siguieron para aplicar la normalización?

1. **1FN**: Eliminamos arrays (ej: lista de profesores en curso → tabla CourseAssignment)
2. **2FN**: Extrajimos atributos no-clave a tablas separadas (ej: datos de facultad de departamento)
3. **3FN**: Eliminamos dependencias transitivas (ej: decano en facultad, no en departamento)

4. ¿Cómo estructuraron los tipos personalizados y para qué los usaron?

Para los tipos de semestre utilizamos tipos personalizados para evitar que se puedan ingresar datos inválidos.

5. ¿Qué beneficios encontraron al usar vistas para el índice?

Es más rápido y es más seguro al no mostrar la lógica del código.

6. ¿Cómo se aseguraron de evitar duplicidad de datos?

Utilizando UNIQUE

7. ¿Qué reglas de negocio implementaron como restricciones y por qué?

No se pueden ingresar créditos negativos ni valores negativos en edad.

8. ¿Qué trigger resultó más útil en el sistema? Justifica.

El de préstamo de libros, cambia el estado automáticamente lo cual lo hace muy práctico.

9. ¿Cuáles fueron las validaciones más complejas y cómo las resolvieron?

El solapamiento de horarios fue complicado de resolver y se terminó realizando con un before insert.

10. ¿Qué compromisos hicieron entre diseño ideal y rendimiento?

Tuvimos que desnormalizar algunas tablas para tener más eficiencia.

Solo aplicamos triggers en lugares críticos.

Hicimos validaciones en la base de datos y el programa, la razón fue porque nos tronamos la base de datos con una mala validación y para evitar eso validamos 2 veces.