Fase 1 — Diseño Conceptual (ER)

Qué hice: construí el diagrama ER con cuatro entidades principales (Autor, Libro, Estudiante, Préstamo), definiendo atributos, claves primarias y foráneas, y las cardinalidades (Autor 1:N Libro; Libro 1:N Préstamo; Estudiante 1:N Préstamo).

Qué aprendí:

- A identificar entidades y relaciones reales del dominio (quién se relaciona con quién y por qué).
- A representar reglas del negocio visualmente (p. ej., un préstamo activo es un vínculo 1:1 temporal entre un libro y un estudiante).
- La importancia de nombrar bien atributos y claves desde el inicio, porque eso evita confusiones en etapas posteriores.

Fase 2 — Normalización (hasta 3FN)

Transformé el ER en tablas y apliqué 1FN, 2FN y 3FN. Definí las dependencias funcionales y dejé ISBN como único en Libro. Mantuvimos Nombre_Autor y Nacionalidad en la tabla Autor para evitar duplicación.

Qué aprendí:

- 1FN nada de listas en una sola columna (evita “varios libros en un mismo préstamo”).
- 2FN: eliminar dependencias parciales de claves compuestas; usar una PK simple en Préstamo (ID_Prestamo) estabiliza el diseño.
- 3FN: evitar dependencias transitivas; p. ej., no guardar Nombre_Autor en Libro, solo ID_Autor.
- Las DF guían el diseño: ISBN → ID_Libro asegura unicidad; ID_Autor → {Nombre, Nacionalidad} evita redundancias.

Fase 3 — Implementación en C++

Qué aprendí:

- A traducir el diseño relacional a C++: las FKs se vuelven IDs que verifico antes de crear registros.
- A implementar reglas de integridad sin SGBD:
    - ISBN único, IDs duplicados bloqueados.
    - FKs: no crear Libro si no existe Autor; no crear Préstamo si no existen Libro/Estudiante.
    - Disponibilidad: impedir dos préstamos activos del mismo libro; al devolver, marcarlo disponible.

- Restricciones de borrado (no borrar Autor con libros; no borrar Préstamo activo; y, según regla, no borrar Libro con historial).