JUSTIFICACION DEL DISEÑO DE LA BASE DE DATOS - SISTEMA DE GESTION DE

El diseño de la base ed datos que desarrollé para el sistema de biblioteca se basa en identificar primero
cuáles son las entidades principales del problema real y cómo se relacionan entre sí. La idea fue
organizar todo de manera que la información esté ordenada, sin repetirse y sea fácil de consultar o
modificar desde la aplicación en Python.

1. Entidades fuertes y por qué las elegí

Las entidades principales del sistema son:
Usuarios: representan a los socios de la biblioteca. Es fundamental saber quién puede pedir libros y
quién paga las cuotas.
Libros: contienen toda la información de cada libro que se presta.
Préstamos: esta tabla une usuarios con libros y registra las fechas y las devoluciones.
Pagos: guarda los pagos mensuales de cada socio.Cuotas: permite registrar el valor de la cuota de cada
mes.
Estas entidades se consideran "fuertes" porque existen por sí mismas (excepto préstamos y pagos,
que necesitan un usuario).

2. Relaciones y cardinalidades

Un usuario puede tener muchos préstamos, pero un préstamo pertenece solo a un usuario relación 1 a
N.
Un libro puede estar en muchos préstamos, pero cada préstamo solo usa un libro relación 1 a N.
Un usuario puede tener muchos pagos, pero cada pago pertenece a un solo usuario relación 1 a N.

3. Normalización aplicada

Apliqué las tres formas normales:
1NF: todos los datos están atómicos, sin listas ni valores repetidos dentro de la misma celda.
2NF: cada atributo depende completamente de la clave primaria (por ejemplo, en préstamos no se
repiten datos del usuario ni del libro).
3NF: eliminé dependencias que no correspondían, evitando que una tabla tenga datos que pertenecen a
otra (por ejemplo, el nombre del usuario no está en préstamos, solo su ID).
Esto permite que no haya redundancia y evita incoherencias.

4. Integridad y restricciones

Usé distintas restricciones para asegurar que los datos siempre sean válidos:
Llaves primarias en todas las tablas.
Llaves foráneas para relacionar préstamos y pagos con usuarios, y préstamos con libros.NOT NULL para
valores obligatorios.
UNIQUE en emails de usuarios y en (mes, año) de cuotas, porque no puede haber duplicados.

5. Cascadas (ON DELETE y ON UPDATE)

Agregué ON DELETE CASCADE en préstamos y pagos para que, si se elimina un usuario (solo si el
sistema lo permite), se borren automáticamente los registros relacionados. Lo mismo para los
préstamos cuando se borra un libro. Esto mantiene la base limpia y evita errores de referencias.

6. Índices

Creé índices en las columnas más consultadas:prestamos (id_usuario) prestamos (id_libro) pagos(id_
usuario, mes, anio) libros (titulo)
Esto mejora la velocidad de búsquedas, sobre todo cuando la base crezca.

7. Justificación del uso de Stored Procedures y Funciones

Para operaciones más complejas, como registrar un préstamo o devolver un libro, usé:
Stored Procedures que hacen transacciones completas e incluyen validaciones.Una función para calcular
multas, lo que evita repetir la lógica en Python.
Esto hace que parte de la lógica quede en la base, lo cual es más seguro.

8. Por qué el modelo es adecuado

El diseño refleja exactamente cómo funciona una biblioteca real:
Usuarios que piden libros.
Libros que tienen stock.
Préstamos que registran todo lo que pasó.
Pagos mensuales.
Cuotas que pueden cambiar cada mes.
Y al mismo tiempo mantiene la base ordenada, rápida y sin datos duplicados.