

# 1. Justificación del diseño y normalización

## 1.1 Diseño conceptual

Utilicé un modelo “Entidad-Relación” basado en las necesidades del hospital “San Juan”, para así modelar la información del sistema antes de ser implementada.

Dando como resultado diez (10) tablas que son: Pacientes, doctores, citas, diagnósticos, tratamientos, medicamentos, tratamientos\_medicamentos, habitaciones, camas y especialidades.

## 1.2 Normalización

La base de datos hospitalaria está normalizada para eliminar redundancias, evitar inconsistencias y optimizar el almacenamiento, siguiendo principalmente las tres primeras formas normales. Cada entidad representa un concepto único (como pacientes, doctores, especialidades, citas, diagnósticos, tratamientos y medicamentos) y sus atributos dependen directamente de su clave primaria, evitando duplicación de información. Las relaciones N:1 y N:M se gestionan mediante claves foráneas y tablas intermedias (por ejemplo, tratamientos\_medicamentos), lo que asegura integridad referencial y facilita actualizaciones, eliminaciones y consultas complejas sin riesgo de anomalías de inserción, actualización o borrado. Esto permite que el sistema sea eficiente, flexible y coherente, reflejando fielmente la operación real de un hospital.

## 1.3 Decisiones de diseño

Con el diseño del código SQL busqué garantizar integridad, consistencia y funcionalidad del sistema hospitalario. Se utilizan instrucciones `DROP TABLE IF EXISTS ... CASCADE` al inicio para eliminar tablas existentes y sus dependencias, evitando errores al recrearlas y asegurando un entorno limpio para la construcción de la base de datos (en resumen: para poder modificar la DB sobre la marcha).

Cada tabla representa una entidad clave (pacientes, doctores, especialidades, citas, diagnósticos, tratamientos, medicamentos, habitaciones y camas) y sus atributos reflejan la información necesaria, con tipos de datos adecuados y restricciones NOT NULL, UNIQUE y CHECK para prevenir valores inválidos. Las relaciones se implementan mediante claves foráneas con reglas de actualización y eliminación (ON UPDATE CASCADE, ON DELETE CASCADE/RESTRICT/SET NULL) para mantener coherencia entre tablas relacionadas. Los valores por defecto (CURRENT\_DATE, TRUE, FALSE, 0, etc.) facilitan la inserción de datos y aseguran consistencia inicial. Finalmente, la inclusión de índices y la separación de relaciones N:M (como tratamientos\_medicamentos) optimizan consultas, reducen redundancia y hacen que el diseño sea robusto, eficiente y alineado con la operación real del hospital.

## 2. Manual de instalación y uso

### 2.1 Requisitos (guía para Windows)

PostgreSQL 14–17

pgAdmin 4 (opcional, pero es el que más recomiendo)

Sistema operativo: Windows 10/11

### 2.2 Creación y ejecución de la DB (método pgAdmin 4).

1. Abre el administrador de bases de datos.
2. Conéctate al servidor (postgres).
3. Clic derecho en Databases – Create – Database.
4. Bautiza a la DB (ej: hospital\_db).
5. Guardar.
6. Seleccionar tu base de datos hospital\_db y abrir el menú Query Tool.
7. Arrastra creación\_db.sql a la ventana, seguido a ello presiona el botón de ejecutar

NOTA: importante verificar que diga “QUERY returned successfully”

1. Para insertar los datos abre nuevamente Query Tool
2. Carga el archivo datos\_prueba.sql
3. ejecuta

## 2.3 Uso

Para el uso solo agregué cinco (5) consultas específicas, en el propio archivo dejé un comentario que detalla cada una, pero de igual manera aquí está la lista de las solicitudes:

- 1) Pacientes con más de 3 citas en el último mes.
- 2) Doctores y su cantidad de citas por especialidad.
- 3) Medicamentos con stock menor al mínimo requerido.
- 4) Habitaciones disponibles por tipo.
- 5) Historial médico completo de un paciente específico.