HospitalDB

# 

**JUAN DAVID SANTOYO JAIMES**

**S1**

**PEDRO FELIPE GÓMEZ BONILLA**

**CAMPUSLANDS**

**CAJASAN  
RUTA NODE.JS**

**FLORIDABLANCA**

**2025**

# 

# Introducción

El objetivo de este proyecto es diseñar y desarrollar un sistema de base de datos en MongoDB para la gestión integral de un sistema hospitalario. Este sistema permite administrar de forma eficiente hospitales, pacientes, médicos, tratamientos, medicamentos, visitas médicas, historiales clínicos, áreas especializadas y personal administrativo.

# 

# 

# Caso de Estudio

El caso de estudio se centra en un sistema hospitalario que necesita una base de datos para gestionar sus operaciones diarias. El objetivo es diseñar y desarrollar un sistema en MongoDB que gestione de manera eficiente hospitales, pacientes, médicos, tratamientos, medicamentos, visitas médicas, historiales clínicos, áreas especializadas y personal administrativo.

# Planificación

## Construcción del Modelo Conceptual

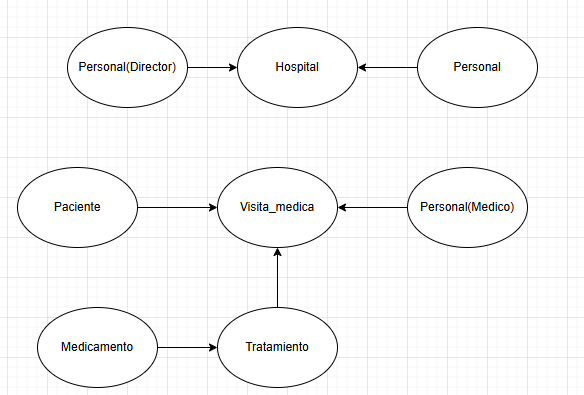
### **Descripción**

El modelo conceptual del sistema hospitalario es una representación de alto nivel de las entidades principales y sus interacciones. Su propósito es definir los componentes del sistema y las reglas del negocio de una manera que sea fácil de entender para todos los involucrados, sin entrar en detalles técnicos sobre la implementación en la base de datos.

Este modelo establece que existen entidades clave como Hospitales, Pacientes y Personal, que son gestionadas por el sistema. Un Hospital emplea a Personal y atiende a Pacientes. Los Pacientes tienen Visitas Médicas, que son atendidas por el Personal médico. A su vez, estas visitas resultan en Tratamientos, que pueden incluir varios Medicamentos.

La principal utilidad de este modelo es la de servir como un borrador del sistema, asegurando que se capturen todas las necesidades y relaciones esenciales antes de proceder con el diseño más detallado de la base de datos.

### Gráfica



### Descripción Técnica

El modelo conceptual se compone de las siguientes entidades, atributos y relaciones, que establecen la estructura de la información del sistema:

#### **Entidades y Atributos**

* **Hospital:** nombre, dirección, áreas especializadas (atributo multivaluado).
* **Personal:** número de identificación (clave), nombre, salario, especialidad (para médicos), rol, teléfono, correo electrónico.
* **Paciente:** historia clínica (clave), nombre, dirección, teléfono, correo electrónico, seguros médicos (atributo multivaluado).
* **Visita Médica:** fecha, hora, diagnóstico.
* **Tratamiento:** nombre, descripción, área médica relacionada, costo.
* **Medicamento:** nombre, fabricante, tipo, disponibilidad.

#### **Relaciones y Cardinalidad**

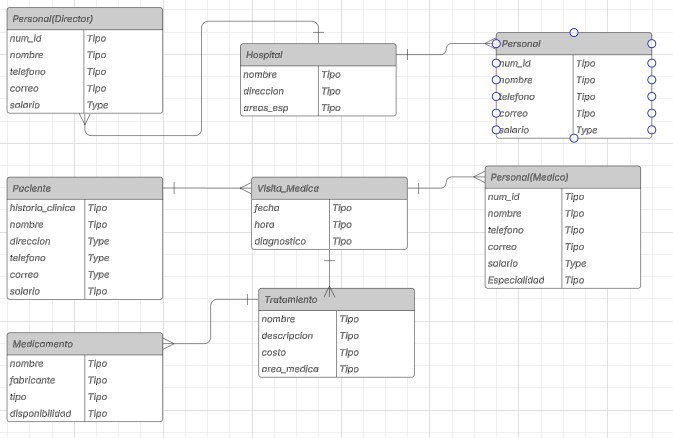
Las relaciones se definen por la forma en que las entidades se conectan, con la siguiente cardinalidad:

* **Hospital emplea Personal:**
  + Un Hospital emplea a muchos miembros del Personal (1:N).
  + Un Personal trabaja en un único Hospital (N:1).
* **Hospital es dirigido por Personal:**
  + Un Hospital es dirigido por un solo Director General (1:1).
  + Un Director General (parte del Personal) dirige un solo Hospital (1:1).
* **Paciente tiene Visita Médica:**
  + Un Paciente tiene muchas Visitas Médicas (1:N).
  + Una Visita Médica es de un único Paciente (N:1).
* **Personal atiende Visita Médica:**
  + Un Médico (parte del Personal) atiende muchas Visitas Médicas (1:N).
  + Una Visita Médica es atendida por un único Médico (N:1).
* **Tratamiento incluye Medicamento:**
  + Un Tratamiento incluye muchos Medicamentos (1:N).
  + Un Medicamento puede ser parte de muchos Tratamientos (N:1). Esta relación es en realidad de muchos a muchos (N:N).

## Construcción del Modelo Lógico

El modelo lógico es la representación detallada de la base de datos que se construirá en MongoDB, basada en las entidades y relaciones del modelo conceptual. En este modelo, las entidades se transforman en **colecciones** y sus atributos en **documentos anidados** o **campos**. Se establece cómo las relaciones entre las entidades se implementarán usando referencias (ObjectId) o embebiendo documentos para optimizar el rendimiento y la flexibilidad, aprovechando la naturaleza NoSQL de MongoDB. El objetivo principal es definir la estructura de los datos para una implementación eficiente y escalable, anticipando las consultas que se realizarán en el sistema.

### Gráfica



### Descripción Técnica

#### **Colecciones y Estructura**

* **Colección personal:**
  + **Propósito:** Almacena a todos los empleados con sus roles y detalles.
  + **Campos clave:** \_id, rol, nombre, especialidad (opcional), colegiatura\_numero (opcional).
* **Colección hospitales:**
  + **Propósito:** Contiene la información de los hospitales.
  + **Campos clave:** \_id, nombre, direccion.
  + **Referencias:** director\_general\_id (hace referencia a personal.\_id).
* **Colección pacientes:**
  + **Propósito:** Guarda los datos de los pacientes.
  + **Campos clave:** \_id, historia\_clinica\_numero, nombre.
  + **Referencias:** historial\_medico (un array de ObjectId que hacen referencia a visitas\_medicas.\_id).
* **Colección visitas\_medicas:**
  + **Propósito:** Registra las visitas de los pacientes.
  + **Campos clave:** \_id, fecha, diagnostico.
  + **Referencias:** paciente\_id (a pacientes.\_id), medico\_id (a personal.\_id).
* **Colección tratamientos:**
  + **Propósito:** Almacena los tratamientos disponibles.
  + **Campos clave:** \_id, nombre, descripcion, costo.
  + **Relación:** medicamentos\_asociados (puede ser un array de documentos embebidos para una consulta más rápida).
* **Colección medicamentos:**
  + **Propósito:** Gestiona el inventario de medicamentos.
  + **Campos clave:** \_id, nombre, fabricante, disponibilidad\_en\_inventario.

#### **Relaciones (implementación en MongoDB)**

* **Referencias 1:1 (ObjectId):**
  + hospitales a personal (Director General): Se usa un único ObjectId en el documento del hospital (director\_general\_id) para apuntar al director correspondiente en la colección personal.
* **Referencias 1:N (Array de ObjectId):**
  + pacientes a visitas\_medicas: El documento del paciente contiene un array de ObjectIds (historial\_medico), cada uno apuntando a un documento de visita médica.
* **Embeber documentos (Array de documentos):**
  + visitas\_medicas a tratamientos: En lugar de una referencia, la visita médica podría contener un array de objetos con el nombre y el costo de los tratamientos aplicados, para evitar una consulta adicional al buscar los detalles de una visita.
* **Relaciones N:N:**
  + tratamientos a medicamentos: Esta relación se maneja típicamente embebiendo documentos o referencias en un array dentro de la colección tratamientos, como un array de objetos que describen el medicamento y la dosis.

## Normalización del Modelo Lógico

La normalización es el proceso de estructurar la base de datos para reducir la redundancia de datos y mejorar la integridad. Aunque MongoDB es una base de datos NoSQL que permite la desnormalización para optimizar el rendimiento, los principios de la normalización se aplican para diseñar documentos lógicos y eficientes.

### **1. Primera Forma Normal (1FN)**

La Primera Forma Normal establece que:

1. Cada campo de un documento debe contener un valor atómico (no puede ser un grupo repetido de valores).
2. Cada documento debe tener una clave primaria que lo identifique de forma única.

**Aplicación a tu proyecto:** Tu modelo ya cumple con esta forma normal en el contexto de MongoDB. Aunque tienes campos como areas\_especializadas y seguros\_medicos que son arrays, en una base de datos de documentos esto se considera un valor atómico. Cada documento en colecciones como personal o pacientes se identifica de manera única por su \_id, que funciona como la clave primaria.

**Ejemplo de cumplimiento:** En la colección pacientes, cada campo (historia\_clinica\_numero, nombre, etc.) contiene un valor único o un array que se trata como una unidad, y cada paciente tiene un \_id único.

### **2. Segunda Forma Normal (2FN)**

La Segunda Forma Normal se aplica a las tablas con claves primarias compuestas y establece que:

* Todos los atributos no clave deben depender completamente de la clave primaria.

**Aplicación a tu proyecto:** Dado que en MongoDB cada documento tiene una única clave primaria (\_id), la mayoría de tus colecciones ya cumplen con la 2FN. Por ejemplo, en el documento del personal, todos los campos (nombre, salario, rol) dependen directamente del \_id de ese documento y no de alguna otra parte de la clave.

**Ejemplo de cumplimiento:** La colección tratamientos con los campos nombre, descripcion y costo depende totalmente de su \_id. No hay datos del medicamento en esta colección, ya que esa información está en su propia colección.

### **3. Tercera Forma Normal (3FN)**

La Tercera Forma Normal establece que:

* No debe haber dependencias transitivas entre los atributos no clave. Es decir, los atributos no clave no deben depender de otros atributos no clave.

**Aplicación a tu proyecto:** Esta es la forma normal más relevante para tu modelo, ya que te ayuda a decidir qué información debe estar en un documento y qué debe estar en otra colección (relacionada por \_id). En tu modelo, has evitado las dependencias transitivas al separar las entidades:

* La información del personal (médicos, directores) está en la colección personal y no se repite en visitas\_medicas ni en hospitales.
* La información de los pacientes está en su propia colección y no en las visitas\_medicas.

En su lugar, utilizas referencias (ObjectId) para conectar las colecciones, como en el caso de visitas\_medicas.paciente\_id o visitas\_medicas.medico\_id.

## Construcción del Modelo Físico

El modelo físico es la implementación práctica del modelo lógico. En él, cada colección se define con su nombre, sus campos y las reglas de validación de esquemas ($jsonSchema) para asegurar que los datos insertados cumplan con la estructura esperada. Las relaciones entre las colecciones, que en el modelo lógico eran abstractas, se materializan aquí mediante el uso de referencias ObjectId. Esta sección incluye el código ejecutable para crear la base de datos y poblarse con los datos iniciales, lo que sirve como la base del proyecto final.

### Código

Aquí se presenta el código completo del modelo físico, que se divide en dos archivos principales: ddl.js para la definición de la estructura y dml.js para la inserción de datos.

**ddl.js:**

// Archivo: ddl.js

// Script para la creación de la base de datos, colecciones y validaciones de esquemas

use HospitalDB;

// Colección para el personal del hospital

db.createCollection("personal", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["tipo\_personal", "rol", "nombre"],

properties: {

tipo\_personal: { bsonType: "string" },

rol: { bsonType: "string" },

nombre: { bsonType: "string" },

colegiatura\_numero: { bsonType: "string" },

especialidad: { bsonType: "string" }

}

}

}

});

// Colección para los hospitales

db.createCollection("hospitales", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["nombre", "direccion", "director\_general\_id"],

properties: {

nombre: { bsonType: "string" },

direccion: { bsonType: "string" },

areas\_especializadas: { bsonType: "array", items: { bsonType: "string" } },

director\_general\_id: { bsonType: "objectId" }

}

}

}

});

// Colección para los pacientes

db.createCollection("pacientes", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["historia\_clinica\_numero", "nombre"],

properties: {

historia\_clinica\_numero: { bsonType: "string" },

nombre: { bsonType: "string" },

historial\_medico: { bsonType: "array", items: { bsonType: "objectId" } }

}

}

}

});

// Colección para las visitas médicas

db.createCollection("visitas\_medicas", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["fecha", "paciente\_id", "medico\_id", "diagnostico"],

properties: {

fecha: { bsonType: "date" },

paciente\_id: { bsonType: "objectId" },

medico\_id: { bsonType: "objectId" },

diagnostico: { bsonType: "string" }

}

}

}

});

// Colección para los tratamientos

db.createCollection("tratamientos", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["nombre", "area\_medica\_relacionada", "costo"],

properties: {

nombre: { bsonType: "string" },

area\_medica\_relacionada: { bsonType: "string" },

costo: { bsonType: "double" }

}

}

}

});

// Colección para los medicamentos

db.createCollection("medicamentos", {

validator: {

$jsonSchema: {

bsonType: "object",

required: ["nombre", "fabricante", "disponibilidad\_en\_inventario"],

properties: {

nombre: { bsonType: "string" },

fabricante: { bsonType: "string" },

tipo: { bsonType: "string" },

disponibilidad\_en\_inventario: { bsonType: "int" }

}

}

}

});

[**dmls.js**](http://dmls.js)**:**

**// Archivo: dml.js**

**// Script para la inserción de datos de ejemplo**

**use HospitalDB;**

**// Inserta el personal**

**db.personal.insertMany([**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3100"),**

**"tipo\_personal": "001",**

**"rol": "Director General",**

**"nombre": "Dr. Juan Perez",**

**"telefono": "555-123-4567",**

**"correo\_electronico": "juan.perez@hospital.com",**

**"salario": 150000,**

**"fecha\_contratacion": ISODate("2020-01-15T00:00:00Z")**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3101"),**

**"tipo\_personal": "002",**

**"rol": "Médico Especialista",**

**"nombre": "Dra. Ana Gomez",**

**"especialidad": "Cardiología",**

**"colegiatura\_numero": "MED-12345",**

**"telefono": "555-987-6543",**

**"correo\_electronico": "ana.gomez@hospital.com",**

**"salario": 90000,**

**"fecha\_contratacion": ISODate("2021-03-20T00:00:00Z")**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3102"),**

**"tipo\_personal": "002",**

**"rol": "Médico Especialista",**

**"nombre": "Dr. Luis Fernandez",**

**"especialidad": "Neurología",**

**"colegiatura\_numero": "MED-54321",**

**"telefono": "555-111-2222",**

**"correo\_electronico": "luis.fernandez@hospital.com",**

**"salario": 95000,**

**"fecha\_contratacion": ISODate("2022-05-10T00:00:00Z")**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3103"),**

**"tipo\_personal": "003",**

**"rol": "Enfermero/a",**

**"nombre": "Maria Lopez",**

**"telefono": "555-333-4444",**

**"correo\_electronico": "maria.lopez@hospital.com",**

**"salario": 45000,**

**"fecha\_contratacion": ISODate("2021-08-01T00:00:00Z")**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3104"),**

**"tipo\_personal": "005",**

**"rol": "Personal de Mantenimiento",**

**"nombre": "Carlos Sanchez",**

**"telefono": "555-555-6666",**

**"correo\_electronico": "carlos.sanchez@hospital.com",**

**"salario": 30000,**

**"fecha\_contratacion": ISODate("2023-02-15T00:00:00Z")**

**}**

**]);**

**// Inserta un hospital**

**db.hospitales.insertOne({**

**"\_id": ObjectId("669ff8e79435b62b168e3110"),**

**"nombre": "Hospital Central de la Ciudad",**

**"direccion": "Calle Falsa 123, Ciudad",**

**"areas\_especializadas": ["Cardiología", "Neurología", "Pediatría"],**

**"personal\_ids": [**

**ObjectId("669ff8e79435b62b168e3100"), // Director General**

**ObjectId("669ff8e79435b62b168e3101"), // Dra. Ana Gomez**

**ObjectId("669ff8e79435b62b168e3102") // Dr. Luis Fernandez**

**],**

**"director\_general\_id": ObjectId("669ff8e79435b62b168e3100")**

**});**

**// Inserta los tratamientos**

**db.tratamientos.insertMany([**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3140"),**

**"nombre": "Terapia de reemplazo hormonal",**

**"descripcion": "Tratamiento para regular hormonas.",**

**"area\_medica\_relacionada": "Neurología",**

**"costo": 500**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3141"),**

**"nombre": "Angioplastia coronaria",**

**"descripcion": "Procedimiento para abrir arterias bloqueadas.",**

**"area\_medica\_relacionada": "Cardiología",**

**"costo": 5000**

**}**

**]);**

**// Inserta los medicamentos**

**db.medicamentos.insertMany([**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3150"),**

**"nombre": "Aspirina",**

**"fabricante": "Bayer",**

**"tipo": "Analgésico",**

**"disponibilidad\_en\_inventario": 1000**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3151"),**

**"nombre": "Propranolol",**

**"fabricante": "AstraZeneca",**

**"tipo": "Betabloqueante",**

**"disponibilidad\_en\_inventario": 500**

**}**

**]);**

**// Inserta los pacientes**

**db.pacientes.insertMany([**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3120"),**

**"historia\_clinica\_numero": "HC-001",**

**"nombre": "Sofia Diaz",**

**"direccion": "Avenida Siempre Viva 742",**

**"telefono": "555-777-8888",**

**"correo\_electronico": "sofia.diaz@email.com",**

**"seguros\_medicos": ["Seguros SA"],**

**"historial\_medico": []**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3121"),**

**"historia\_clinica\_numero": "HC-002",**

**"nombre": "Ricardo Morales",**

**"direccion": "Calle de los Olivos 50",**

**"telefono": "555-999-0000",**

**"correo\_electronico": "ricardo.morales@email.com",**

**"seguros\_medicos": ["Salud Vital"],**

**"historial\_medico": []**

**}**

**]);**

**// Inserta las visitas médicas**

**db.visitas\_medicas.insertMany([**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3130"),**

**"fecha": ISODate("2024-07-20T10:00:00Z"),**

**"paciente\_id": ObjectId("669ff8e79435b62b168e3120"), // Sofia Diaz**

**"medico\_id": ObjectId("669ff8e79435b62b168e3101"), // Dra. Ana Gomez**

**"diagnostico": "Insuficiencia cardiaca leve",**

**"tratamientos\_aplicados": [**

**{"nombre": "Angioplastia coronaria", "costo": 5000}**

**]**

**},**

**{**

**"\_id": ObjectId("669ff8e79435b62b168e3131"),**

**"fecha": ISODate("2024-07-22T14:30:00Z"),**

**"paciente\_id": ObjectId("669ff8e79435b62b168e3121"), // Ricardo Morales**

**"medico\_id": ObjectId("669ff8e79435b62b168e3102"), // Dr. Luis Fernandez**

**"diagnostico": "Migraña crónica",**

**"tratamientos\_aplicados": [**

**{"nombre": "Terapia de reemplazo hormonal", "costo": 500}**

**]**

**}**

**]);**

### Descripción Técnica

El modelo físico se ha construido siguiendo una estructura de documentos con referencias para manejar las relaciones entre colecciones:

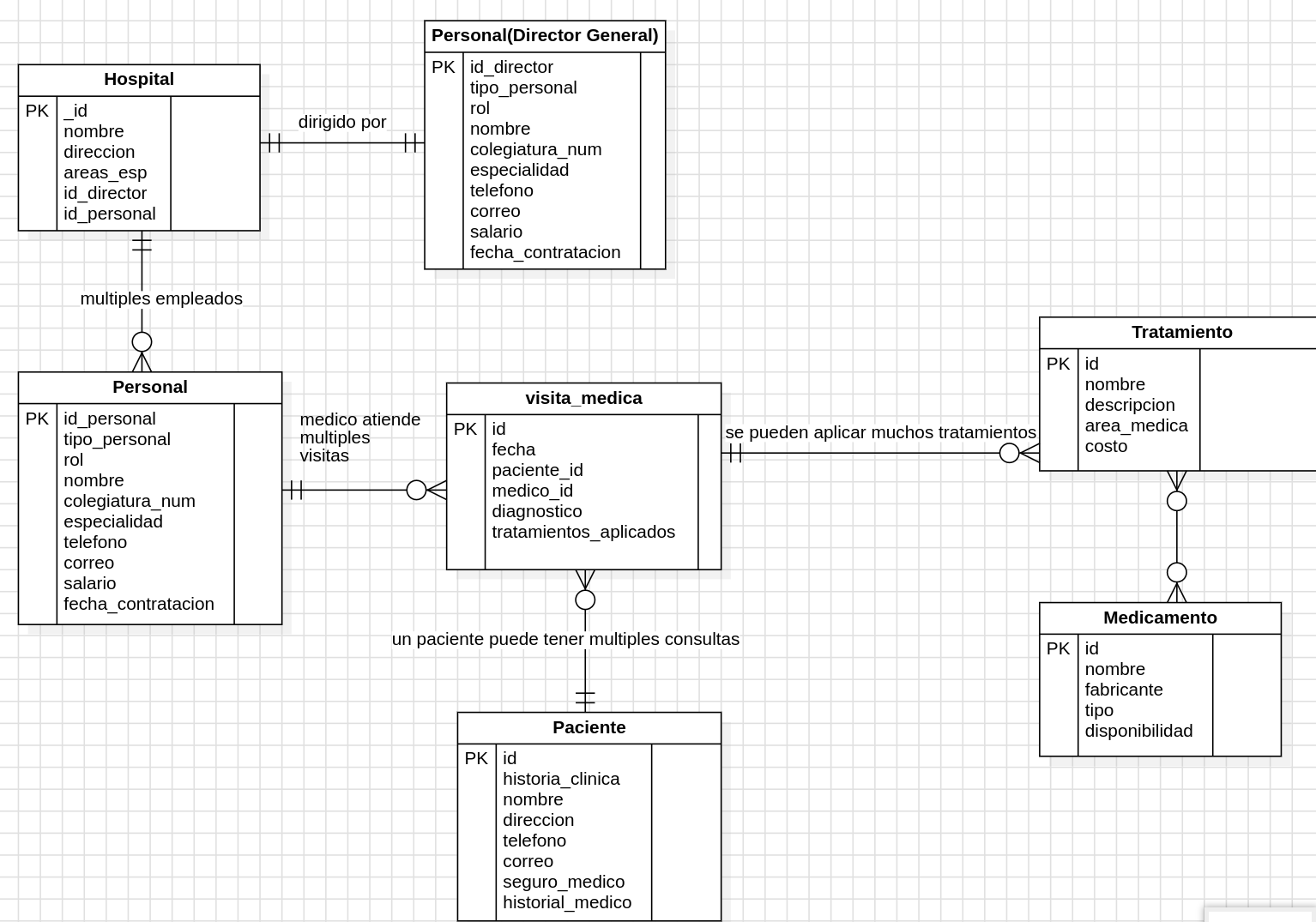
* **Creación de Colecciones:** El script ddl.js utiliza el método db.createCollection() para crear cada una de las colecciones. Se implementa la validación de esquemas ($jsonSchema) para asegurar que todos los documentos insertados tengan los campos requeridos y los tipos de datos correctos (string, objectId, date, etc.).
* **Inserción de Datos:** El script dml.js utiliza insertMany() e insertOne() para poblar las colecciones con datos de ejemplo. Se han asignado ObjectIds estáticos para mantener la coherencia y las relaciones entre los documentos.
* **Relaciones:** Las relaciones se gestionan a través de ObjectIds. Por ejemplo, en el documento del hospital se usa director\_general\_id para referenciar al director en la colección personal. De manera similar, en los documentos de visitas\_medicas, se utilizan paciente\_id y medico\_id para conectar las visitas con sus respectivos pacientes y médicos.
* **Datos Anidados:** Se ha optado por la desnormalización en la colección visitas\_medicas, donde se anidan los detalles de los tratamientos\_aplicados. Esto se hace para mejorar el rendimiento de las consultas, ya que se puede obtener toda la información de una visita en una sola lectura sin necesidad de un join.

## Diagrama E-R

El Diagrama Entidad-Relación (E-R) del Sistema Hospitalario representa la estructura lógica de la base de datos, mostrando las entidades principales, sus atributos y las relaciones que existen entre ellas. Este diagrama sirve como base para la implementación en MongoDB, donde las entidades se convierten en colecciones y las relaciones se manejan mediante referencias ObjectId o documentos embebidos.

El diagrama incluye seis entidades principales: Hospital, Personal, Paciente, Visita Médica, Tratamiento y Medicamento. Cada entidad tiene atributos específicos que describen sus características, y las relaciones entre ellas definen cómo interactúan los diferentes componentes del sistema hospitalario.

#### Gráfica



#### Descripción Técnica

### **Entidades y Atributos**

#### **1. HOSPITAL**

* **\_id (ObjectId):** Clave primaria única que identifica cada hospital
* **nombre (String):** Denominación oficial del hospital
* **direccion (String):** Ubicación física del hospital
* **areas\_especializadas (Array[String]):** Lista de especialidades médicas disponibles
* **director\_general\_id (ObjectId):** Referencia al personal que dirige el hospital

#### **2. PERSONAL**

* **\_id (ObjectId):** Clave primaria única para cada empleado
* **tipo\_personal (String):** Código que identifica el tipo de empleado (001-005)
* **rol (String):** Descripción del puesto (Director General, Médico Especialista, etc.)
* **nombre (String):** Nombre completo del empleado
* **colegiatura\_numero (String):** Número de registro profesional (opcional, solo médicos)
* **especialidad (String):** Área de especialización médica (opcional, solo médicos)
* **telefono (String):** Número de contacto
* **correo\_electronico (String):** Dirección de correo electrónico
* **salario (Number):** Remuneración mensual
* **fecha\_contratacion (Date):** Fecha de ingreso al hospital

#### **3. PACIENTE**

* **\_id (ObjectId):** Clave primaria única para cada paciente
* **historia\_clinica\_numero (String):** Código único de identificación médica
* **nombre (String):** Nombre completo del paciente
* **direccion (String):** Domicilio del paciente
* **telefono (String):** Número de contacto
* **correo\_electronico (String):** Dirección de correo electrónico
* **seguros\_medicos (Array[String]):** Lista de seguros médicos activos
* **historial\_medico (Array[ObjectId]):** Referencias a las visitas médicas del paciente

#### **4. VISITA\_MEDICA**

* **\_id (ObjectId):** Clave primaria única para cada visita
* **fecha (Date):** Fecha y hora de la consulta médica
* **paciente\_id (ObjectId):** Referencia al paciente atendido
* **medico\_id (ObjectId):** Referencia al médico que realizó la consulta
* **diagnostico (String):** Resultado del examen médico
* **tratamientos\_aplicados (Array[Object]):** Lista de tratamientos realizados durante la visita

#### **5. TRATAMIENTO**

* **\_id (ObjectId):** Clave primaria única para cada tratamiento
* **nombre (String):** Denominación del procedimiento médico
* **descripcion (String):** Detalle del tratamiento
* **area\_medica\_relacionada (String):** Especialidad médica correspondiente
* **costo (Number):** Precio del tratamiento

#### **6. MEDICAMENTO**

* **\_id (ObjectId):** Clave primaria única para cada medicamento
* **nombre (String):** Denominación comercial o genérica
* **fabricante (String):** Empresa productora del medicamento
* **tipo (String):** Clasificación farmacológica
* **disponibilidad\_en\_inventario (Number):** Cantidad disponible en stock

### **Relaciones entre Entidades**

#### **1. Hospital es dirigido por Personal (1:1)**

* **Descripción:** Cada hospital tiene un único director general
* **Cardinalidad:** Un hospital tiene un director, un director dirige un hospital
* **Implementación:** Campo director\_general\_id en la colección hospitales
* **Restricciones:** El director debe tener rol "Director General"

#### **2. Hospital emplea Personal (1:N)**

* **Descripción:** Un hospital puede tener múltiples empleados
* **Cardinalidad:** Un hospital emplea muchos empleados, un empleado trabaja en un hospital
* **Implementación:** Campo personal\_ids (array) en la colección hospitales
* **Restricciones:** Cada empleado debe estar asignado a un hospital específico

#### **3. Personal atiende Visita Médica (1:N)**

* **Descripción:** Un médico puede atender múltiples visitas médicas
* **Cardinalidad:** Un médico atiende muchas visitas, una visita es atendida por un médico
* **Implementación:** Campo medico\_id en la colección visitas\_medicas
* **Restricciones:** Solo el personal con rol "Médico Especialista" puede atender visitas

#### **4. Paciente tiene Visita Médica (1:N)**

* **Descripción:** Un paciente puede tener múltiples consultas médicas a lo largo del tiempo
* **Cardinalidad:** Un paciente tiene muchas visitas, una visita pertenece a un paciente
* **Implementación:** Campo paciente\_id en visitas\_medicas y array historial\_medico en pacientes
* **Restricciones:** Cada visita debe estar asociada a un paciente válido

#### **5. Tratamiento incluye Medicamento (N:N)**

* **Descripción:** Un tratamiento puede requerir varios medicamentos, y un medicamento puede usarse en varios tratamientos
* **Cardinalidad:** Muchos tratamientos incluyen muchos medicamentos
* **Implementación:** Array de objetos embebidos en tratamientos o colección intermedia
* **Restricciones:** Los medicamentos deben estar disponibles en inventario

#### **6. Visita Médica aplica Tratamiento (1:N)**

* **Descripción:** Durante una visita médica se pueden aplicar uno o varios tratamientos
* **Cardinalidad:** Una visita puede incluir muchos tratamientos, un tratamiento puede aplicarse en muchas visitas
* **Implementación:** Array tratamientos\_aplicados embebido en visitas\_medicas
* **Restricciones:** Los tratamientos deben corresponder al área de especialidad del médico

### **Reglas de Integridad**

1. **Integridad de Entidad:** Cada entidad debe tener una clave primaria única (\_id)
2. **Integridad Referencial:** Las referencias ObjectId deben apuntar a documentos válidos
3. **Integridad de Dominio:** Los valores deben cumplir con las validaciones de esquema definidas
4. **Reglas de Negocio:**
   * Solo médicos especialistas pueden atender visitas médicas
   * Los directores generales deben estar asignados a un hospital específico
   * Los medicamentos deben tener disponibilidad positiva para ser incluidos en tratamientos
   * Las áreas especializadas del hospital deben coincidir con las especialidades del personal médico

**Consultas:**

// =================================================================

// 100 CONSULTAS MONGODB - SISTEMA HOSPITALARIO

// Base de datos: HospitalDB

// =================================================================

use HospitalDB;

// =================================================================

// SECCIÓN 1: CONSULTAS BÁSICAS DE PERSONAL (1-15)

// =================================================================

// 1. Obtener todo el personal del hospital

db.personal.find();

// 2. Obtener solo médicos especialistas

db.personal.find({ rol: "Médico Especialista" });

// 3. Obtener directores generales

db.personal.find({ rol: "Director General" });

// 4. Obtener personal de un tipo específico

db.personal.find({ tipo\_personal: "002" });

// 5. Obtener médicos con especialidad específica

db.personal.find({ especialidad: "Cardiología" });

// 6. Obtener personal con salario mayor a 50000

db.personal.find({ salario: { $gt: 50000 } });

// 7. Obtener personal contratado después de 2021

db.personal.find({ fecha\_contratacion: { $gte: ISODate("2021-01-01") } });

// 8. Contar total de empleados

db.personal.countDocuments();

// 9. Contar médicos por especialidad

db.personal.countDocuments({ especialidad: "Cardiología" });

// 10. Obtener enfermeros

db.personal.find({ rol: "Enfermero/a" });

// 11. Obtener personal administrativo

db.personal.find({ rol: "Personal Administrativo" });

// 12. Obtener personal de mantenimiento

db.personal.find({ rol: "Personal de Mantenimiento" });

// 13. Buscar personal por nombre (expresión regular)

db.personal.find({ nombre: { $regex: "Dr.", $options: "i" } });

// 14. Obtener personal con colegiatura

db.personal.find({ colegiatura\_numero: { $exists: true } });

// 15. Obtener personal ordenado por salario descendente

db.personal.find().sort({ salario: -1 });

// =================================================================

// SECCIÓN 2: CONSULTAS BÁSICAS DE HOSPITALES (16-25)

// =================================================================

// 16. Obtener todos los hospitales

db.hospitales.find();

// 17. Obtener hospital por nombre

db.hospitales.find({ nombre: "Hospital Central de la Ciudad" });

// 18. Obtener hospitales con área especializada específica

db.hospitales.find({ areas\_especializadas: "Cardiología" });

// 19. Obtener hospitales con múltiples especialidades

db.hospitales.find({ areas\_especializadas: { $size: { $gte: 3 } } });

// 20. Obtener director general de un hospital específico

db.hospitales.find({}, { director\_general\_id: 1, nombre: 1 });

// 21. Contar total de hospitales

db.hospitales.countDocuments();

// 22. Obtener hospitales que tengan "Neurología"

db.hospitales.find({ areas\_especializadas: { $in: ["Neurología"] } });

// 23. Obtener hospitales ordenados por nombre

db.hospitales.find().sort({ nombre: 1 });

// 24. Obtener solo nombres y direcciones de hospitales

db.hospitales.find({}, { nombre: 1, direccion: 1 });

// 25. Buscar hospitales por ubicación

db.hospitales.find({ direccion: { $regex: "Ciudad", $options: "i" } });

// =================================================================

// SECCIÓN 3: CONSULTAS BÁSICAS DE PACIENTES (26-40)

// =================================================================

// 26. Obtener todos los pacientes

db.pacientes.find();

// 27. Obtener paciente por historia clínica

db.pacientes.find({ historia\_clinica\_numero: "HC-001" });

// 28. Obtener pacientes por nombre

db.pacientes.find({ nombre: { $regex: "Sofia", $options: "i" } });

// 29. Contar total de pacientes

db.pacientes.countDocuments();

// 30. Obtener pacientes con historial médico

db.pacientes.find({ historial\_medico: { $exists: true, $ne: [] } });

// 31. Obtener pacientes sin historial médico

db.pacientes.find({ $or: [{ historial\_medico: { $exists: false } }, { historial\_medico: [] }] });

// 32. Obtener pacientes con seguros específicos

db.pacientes.find({ seguros\_medicos: "Seguros SA" });

// 33. Obtener pacientes con múltiples seguros

db.pacientes.find({ seguros\_medicos: { $size: { $gte: 2 } } });

// 34. Buscar pacientes por dirección

db.pacientes.find({ direccion: { $regex: "Avenida", $options: "i" } });

// 35. Obtener pacientes ordenados por nombre

db.pacientes.find().sort({ nombre: 1 });

// 36. Obtener solo nombres e historias clínicas

db.pacientes.find({}, { nombre: 1, historia\_clinica\_numero: 1 });

// 37. Obtener pacientes con correo electrónico

db.pacientes.find({ correo\_electronico: { $exists: true } });

// 38. Obtener pacientes por teléfono

db.pacientes.find({ telefono: { $regex: "^555" } });

// 39. Contar pacientes por seguro médico

db.pacientes.countDocuments({ seguros\_medicos: "Salud Vital" });

// 40. Obtener pacientes con historia clínica específica

db.pacientes.find({ historia\_clinica\_numero: { $in: ["HC-001", "HC-002"] } });

// =================================================================

// SECCIÓN 4: CONSULTAS BÁSICAS DE VISITAS MÉDICAS (41-55)

// =================================================================

// 41. Obtener todas las visitas médicas

db.visitas\_medicas.find();

// 42. Obtener visitas de un paciente específico

db.visitas\_medicas.find({ paciente\_id: ObjectId("669ff8e79435b62b168e3120") });

// 43. Obtener visitas de un médico específico

db.visitas\_medicas.find({ medico\_id: ObjectId("669ff8e79435b62b168e3101") });

// 44. Obtener visitas por fecha específica

db.visitas\_medicas.find({ fecha: ISODate("2024-07-20T10:00:00Z") });

// 45. Obtener visitas en un rango de fechas

db.visitas\_medicas.find({

fecha: {

$gte: ISODate("2024-07-01T00:00:00Z"),

$lte: ISODate("2024-07-31T23:59:59Z")

}

});

// 46. Obtener visitas por diagnóstico

db.visitas\_medicas.find({ diagnostico: { $regex: "cardiaca", $options: "i" } });

// 47. Contar total de visitas médicas

db.visitas\_medicas.countDocuments();

// 48. Obtener visitas ordenadas por fecha

db.visitas\_medicas.find().sort({ fecha: -1 });

// 49. Obtener visitas del día actual

db.visitas\_medicas.find({

fecha: {

$gte: new Date(new Date().setHours(0, 0, 0, 0)),

$lt: new Date(new Date().setHours(23, 59, 59, 999))

}

});

// 50. Obtener visitas con tratamientos aplicados

db.visitas\_medicas.find({ tratamientos\_aplicados: { $exists: true, $ne: [] } });

// 51. Contar visitas por médico

db.visitas\_medicas.countDocuments({ medico\_id: ObjectId("669ff8e79435b62b168e3101") });

// 52. Obtener diagnósticos únicos

db.visitas\_medicas.distinct("diagnostico");

// 53. Obtener visitas con costo alto en tratamientos

db.visitas\_medicas.find({ "tratamientos\_aplicados.costo": { $gte: 1000 } });

// 54. Obtener visitas de la última semana

db.visitas\_medicas.find({

fecha: { $gte: new Date(Date.now() - 7 \* 24 \* 60 \* 60 \* 1000) }

});

// 55. Obtener visitas con tratamiento específico

db.visitas\_medicas.find({ "tratamientos\_aplicados.nombre": "Angioplastia coronaria" });

// =================================================================

// SECCIÓN 5: CONSULTAS BÁSICAS DE TRATAMIENTOS (56-65)

// =================================================================

// 56. Obtener todos los tratamientos

db.tratamientos.find();

// 57. Obtener tratamientos por área médica

db.tratamientos.find({ area\_medica\_relacionada: "Cardiología" });

// 58. Obtener tratamientos por rango de costo

db.tratamientos.find({ costo: { $gte: 1000, $lte: 5000 } });

// 59. Obtener tratamientos más costosos

db.tratamientos.find().sort({ costo: -1 }).limit(5);

// 60. Obtener tratamientos más económicos

db.tratamientos.find().sort({ costo: 1 }).limit(5);

// 61. Contar tratamientos por área médica

db.tratamientos.countDocuments({ area\_medica\_relacionada: "Neurología" });

// 62. Obtener áreas médicas únicas

db.tratamientos.distinct("area\_medica\_relacionada");

// 63. Buscar tratamientos por nombre

db.tratamientos.find({ nombre: { $regex: "terapia", $options: "i" } });

// 64. Obtener tratamientos con descripción

db.tratamientos.find({ descripcion: { $exists: true } });

// 65. Obtener costo promedio de tratamientos

db.tratamientos.aggregate([{ $group: { \_id: null, costo\_promedio: { $avg: "$costo" } } }]);

// =================================================================

// SECCIÓN 6: CONSULTAS BÁSICAS DE MEDICAMENTOS (66-75)

// =================================================================

// 66. Obtener todos los medicamentos

db.medicamentos.find();

// 67. Obtener medicamentos por fabricante

db.medicamentos.find({ fabricante: "Bayer" });

// 68. Obtener medicamentos por tipo

db.medicamentos.find({ tipo: "Analgésico" });

// 69. Obtener medicamentos con stock bajo

db.medicamentos.find({ disponibilidad\_en\_inventario: { $lt: 100 } });

// 70. Obtener medicamentos con stock alto

db.medicamentos.find({ disponibilidad\_en\_inventario: { $gte: 500 } });

// 71. Obtener medicamentos sin stock

db.medicamentos.find({ disponibilidad\_en\_inventario: 0 });

// 72. Contar medicamentos por fabricante

db.medicamentos.countDocuments({ fabricante: "AstraZeneca" });

// 73. Obtener fabricantes únicos

db.medicamentos.distinct("fabricante");

// 74. Obtener tipos de medicamentos únicos

db.medicamentos.distinct("tipo");

// 75. Buscar medicamentos por nombre

db.medicamentos.find({ nombre: { $regex: "Aspirina", $options: "i" } });

// =================================================================

// SECCIÓN 7: CONSULTAS CON AGREGACIONES AVANZADAS (76-100)

// =================================================================

// 76. Obtener información completa de visitas con datos de paciente y médico

db.visitas\_medicas.aggregate([

{

$lookup: {

from: "pacientes",

localField: "paciente\_id",

foreignField: "\_id",

as: "paciente\_info"

}

},

{

$lookup: {

from: "personal",

localField: "medico\_id",

foreignField: "\_id",

as: "medico\_info"

}

},

{

$unwind: "$paciente\_info"

},

{

$unwind: "$medico\_info"

}

]);

// 77. Obtener estadísticas de visitas por médico

db.visitas\_medicas.aggregate([

{

$group: {

\_id: "$medico\_id",

total\_visitas: { $sum: 1 },

diagnosticos\_unicos: { $addToSet: "$diagnostico" }

}

},

{

$lookup: {

from: "personal",

localField: "\_id",

foreignField: "\_id",

as: "medico\_info"

}

},

{

$unwind: "$medico\_info"

},

{

$project: {

medico\_nombre: "$medico\_info.nombre",

total\_visitas: 1,

cantidad\_diagnosticos\_unicos: { $size: "$diagnosticos\_unicos" }

}

}

]);

// 78. Obtener pacientes con su historial completo de visitas

db.pacientes.aggregate([

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "paciente\_id",

as: "visitas"

}

},

{

$project: {

nombre: 1,

historia\_clinica\_numero: 1,

total\_visitas: { $size: "$visitas" },

ultima\_visita: { $max: "$visitas.fecha" }

}

}

]);

// 79. Obtener hospitales con información de su director

db.hospitales.aggregate([

{

$lookup: {

from: "personal",

localField: "director\_general\_id",

foreignField: "\_id",

as: "director\_info"

}

},

{

$unwind: "$director\_info"

},

{

$project: {

nombre: 1,

direccion: 1,

areas\_especializadas: 1,

director\_nombre: "$director\_info.nombre",

director\_salario: "$director\_info.salario"

}

}

]);

// 80. Obtener estadísticas de medicamentos por fabricante

db.medicamentos.aggregate([

{

$group: {

\_id: "$fabricante",

total\_medicamentos: { $sum: 1 },

stock\_total: { $sum: "$disponibilidad\_en\_inventario" },

tipos\_medicamentos: { $addToSet: "$tipo" }

}

},

{

$project: {

fabricante: "$\_id",

total\_medicamentos: 1,

stock\_total: 1,

variedad\_tipos: { $size: "$tipos\_medicamentos" }

}

}

]);

// 81. Obtener diagnósticos más frecuentes

db.visitas\_medicas.aggregate([

{

$group: {

\_id: "$diagnostico",

frecuencia: { $sum: 1 }

}

},

{

$sort: { frecuencia: -1 }

},

{

$limit: 10

}

]);

// 82. Obtener costos totales por área médica

db.tratamientos.aggregate([

{

$group: {

\_id: "$area\_medica\_relacionada",

costo\_total: { $sum: "$costo" },

cantidad\_tratamientos: { $sum: 1 },

costo\_promedio: { $avg: "$costo" }

}

}

]);

// 83. Obtener personal médico con sus especialidades y cantidad de visitas

db.personal.aggregate([

{

$match: { rol: "Médico Especialista" }

},

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "medico\_id",

as: "visitas\_realizadas"

}

},

{

$project: {

nombre: 1,

especialidad: 1,

total\_visitas: { $size: "$visitas\_realizadas" }

}

}

]);

// 84. Obtener resumen de actividad hospitalaria por mes

db.visitas\_medicas.aggregate([

{

$group: {

\_id: {

year: { $year: "$fecha" },

month: { $month: "$fecha" }

},

total\_visitas: { $sum: 1 },

pacientes\_unicos: { $addToSet: "$paciente\_id" }

}

},

{

$project: {

año: "$\_id.year",

mes: "$\_id.month",

total\_visitas: 1,

pacientes\_unicos: { $size: "$pacientes\_unicos" }

}

},

{

$sort: { "\_id.year": -1, "\_id.month": -1 }

}

]);

// 85. Obtener medicamentos críticos (stock bajo) con información detallada

db.medicamentos.aggregate([

{

$match: { disponibilidad\_en\_inventario: { $lt: 100 } }

},

{

$project: {

nombre: 1,

fabricante: 1,

tipo: 1,

stock\_actual: "$disponibilidad\_en\_inventario",

estado\_critico: {

$cond: {

if: { $lt: ["$disponibilidad\_en\_inventario", 50] },

then: "Crítico",

else: "Bajo"

}

}

}

},

{

$sort: { disponibilidad\_en\_inventario: 1 }

}

]);

// 86. Obtener análisis de productividad del personal médico

db.personal.aggregate([

{

$match: { rol: "Médico Especialista" }

},

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "medico\_id",

as: "visitas"

}

},

{

$unwind: {

path: "$visitas",

preserveNullAndEmptyArrays: true

}

},

{

$group: {

\_id: "$\_id",

nombre: { $first: "$nombre" },

especialidad: { $first: "$especialidad" },

salario: { $first: "$salario" },

total\_visitas: { $sum: 1 },

ingresos\_generados: {

$sum: {

$sum: "$visitas.tratamientos\_aplicados.costo"

}

}

}

},

{

$project: {

nombre: 1,

especialidad: 1,

salario: 1,

total\_visitas: 1,

ingresos\_generados: 1,

ratio\_productividad: {

$divide: ["$ingresos\_generados", "$salario"]

}

}

}

]);

// 87. Obtener pacientes con mayor número de visitas y sus diagnósticos

db.pacientes.aggregate([

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "paciente\_id",

as: "visitas"

}

},

{

$project: {

nombre: 1,

historia\_clinica\_numero: 1,

total\_visitas: { $size: "$visitas" },

diagnosticos: "$visitas.diagnostico"

}

},

{

$match: { total\_visitas: { $gte: 1 } }

},

{

$sort: { total\_visitas: -1 }

}

]);

// 88. Obtener análisis de costos por tratamiento y frecuencia de uso

db.visitas\_medicas.aggregate([

{

$unwind: "$tratamientos\_aplicados"

},

{

$group: {

\_id: "$tratamientos\_aplicados.nombre",

frecuencia\_uso: { $sum: 1 },

costo\_promedio: { $avg: "$tratamientos\_aplicados.costo" },

costo\_total\_generado: { $sum: "$tratamientos\_aplicados.costo" }

}

},

{

$sort: { frecuencia\_uso: -1 }

}

]);

// 89. Obtener especialidades médicas con mayor demanda

db.personal.aggregate([

{

$match: {

rol: "Médico Especialista",

especialidad: { $exists: true }

}

},

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "medico\_id",

as: "visitas"

}

},

{

$group: {

\_id: "$especialidad",

medicos\_disponibles: { $sum: 1 },

total\_visitas: { $sum: { $size: "$visitas" } },

promedio\_visitas\_por\_medico: {

$avg: { $size: "$visitas" }

}

}

},

{

$sort: { total\_visitas: -1 }

}

]);

// 90. Obtener pacientes con seguros médicos y su actividad

db.pacientes.aggregate([

{

$match: { seguros\_medicos: { $exists: true, $ne: [] } }

},

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "paciente\_id",

as: "visitas"

}

},

{

$unwind: "$seguros\_medicos"

},

{

$group: {

\_id: "$seguros\_medicos",

pacientes\_asegurados: { $sum: 1 },

total\_visitas: { $sum: { $size: "$visitas" } }

}

}

]);

// 91. Obtener análisis temporal de diagnósticos por mes

db.visitas\_medicas.aggregate([

{

$group: {

\_id: {

año: { $year: "$fecha" },

mes: { $month: "$fecha" },

diagnostico: "$diagnostico"

},

casos: { $sum: 1 }

}

},

{

$sort: { "\_id.año": -1, "\_id.mes": -1, "casos": -1 }

}

]);

// 92. Obtener médicos con mayor generación de ingresos

db.visitas\_medicas.aggregate([

{

$unwind: "$tratamientos\_aplicados"

},

{

$group: {

\_id: "$medico\_id",

ingresos\_totales: { $sum: "$tratamientos\_aplicados.costo" },

tratamientos\_realizados: { $sum: 1 }

}

},

{

$lookup: {

from: "personal",

localField: "\_id",

foreignField: "\_id",

as: "medico\_info"

}

},

{

$unwind: "$medico\_info"

},

{

$project: {

medico\_nombre: "$medico\_info.nombre",

especialidad: "$medico\_info.especialidad",

ingresos\_totales: 1,

tratamientos\_realizados: 1,

ingreso\_promedio\_por\_tratamiento: {

$divide: ["$ingresos\_totales", "$tratamientos\_realizados"]

}

}

},

{

$sort: { ingresos\_totales: -1 }

}

]);

// 93. Obtener inventario de medicamentos con alertas de stock

db.medicamentos.aggregate([

{

$project: {

nombre: 1,

fabricante: 1,

tipo: 1,

stock\_actual: "$disponibilidad\_en\_inventario",

estado\_stock: {

$switch: {

branches: [

{ case: { $eq: ["$disponibilidad\_en\_inventario", 0] }, then: "Sin Stock" },

{ case: { $lt: ["$disponibilidad\_en\_inventario", 50] }, then: "Stock Crítico" },

{ case: { $lt: ["$disponibilidad\_en\_inventario", 200] }, then: "Stock Bajo" },

{ case: { $gte: ["$disponibilidad\_en\_inventario", 200] }, then: "Stock Normal" }

],

default: "Estado Desconocido"

}

}

}

},

{

$sort: { disponibilidad\_en\_inventario: 1 }

}

]);

// 94. Obtener patrones de diagnósticos por especialidad médica

db.visitas\_medicas.aggregate([

{

$lookup: {

from: "personal",

localField: "medico\_id",

foreignField: "\_id",

as: "medico\_info"

}

},

{

$unwind: "$medico\_info"

},

{

$group: {

\_id: {

especialidad: "$medico\_info.especialidad",

diagnostico: "$diagnostico"

},

frecuencia: { $sum: 1 }

}

},

{

$sort: { "\_id.especialidad": 1, "frecuencia": -1 }

}

]);

// 95. Obtener análisis de eficiencia hospitalaria

db.hospitales.aggregate([

{

$lookup: {

from: "personal",

localField: "director\_general\_id",

foreignField: "\_id",

as: "director"

}

},

{

$lookup: {

from: "visitas\_medicas",

let: { hospital\_id: "$\_id" },

pipeline: [

{

$lookup: {

from: "personal",

localField: "medico\_id",

foreignField: "\_id",

as: "medico"

}

}

],

as: "todas\_visitas"

}

},

{

$project: {

nombre: 1,

areas\_especializadas: 1,

cantidad\_especialidades: { $size: "$areas\_especializadas" },

director: { $arrayElemAt: ["$director.nombre", 0] },

total\_visitas\_estimadas: { $size: "$todas\_visitas" }

}

}

]);

// 96. Obtener resumen financiero por área médica

db.tratamientos.aggregate([

{

$group: {

\_id: "$area\_medica\_relacionada",

tratamientos\_disponibles: { $sum: 1 },

costo\_minimo: { $min: "$costo" },

costo\_maximo: { $max: "$costo" },

costo\_promedio: { $avg: "$costo" },

inversion\_total: { $sum: "$costo" }

}

},

{

$sort: { inversion\_total: -1 }

}

]);

// 97. Obtener pacientes con tratamientos más costosos

db.visitas\_medicas.aggregate([

{

$unwind: "$tratamientos\_aplicados"

},

{

$lookup: {

from: "pacientes",

localField: "paciente\_id",

foreignField: "\_id",

as: "paciente\_info"

}

},

{

$unwind: "$paciente\_info"

},

{

$group: {

\_id: "$paciente\_id",

nombre\_paciente: { $first: "$paciente\_info.nombre" },

historia\_clinica: { $first: "$paciente\_info.historia\_clinica\_numero" },

costo\_total\_tratamientos: { $sum: "$tratamientos\_aplicados.costo" },

cantidad\_tratamientos: { $sum: 1 }

}

},

{

$sort: { costo\_total\_tratamientos: -1 }

},

{

$limit: 10

}

]);

// 98. Obtener estadísticas de personal por tipo y salarios

db.personal.aggregate([

{

$group: {

\_id: "$tipo\_personal",

rol\_representativo: { $first: "$rol" },

cantidad\_empleados: { $sum: 1 },

salario\_promedio: { $avg: "$salario" },

salario\_total: { $sum: "$salario" },

salario\_minimo: { $min: "$salario" },

salario\_maximo: { $max: "$salario" }

}

},

{

$sort: { \_id: 1 }

}

]);

// 99. Obtener análisis de seguros médicos más utilizados

db.pacientes.aggregate([

{

$unwind: "$seguros\_medicos"

},

{

$lookup: {

from: "visitas\_medicas",

localField: "\_id",

foreignField: "paciente\_id",

as: "visitas\_paciente"

}

},

{

$group: {

\_id: "$seguros\_medicos",

pacientes\_afiliados: { $sum: 1 },

total\_visitas\_cubiertas: { $sum: { $size: "$visitas\_paciente" } }

}

},

{

$project: {

seguro\_medico: "$\_id",

pacientes\_afiliados: 1,

total\_visitas\_cubiertas: 1,

promedio\_visitas\_por\_paciente: {

$divide: ["$total\_visitas\_cubiertas", "$pacientes\_afiliados"]

}

}

},

{

$sort: { pacientes\_afiliados: -1 }

}

]);

// 100. Obtener reporte completo de actividad hospitalaria

db.hospitales.aggregate([

{

$lookup: {

from: "personal",

let: { hospital\_id: "$\_id" },

pipeline: [

{ $match: { $expr: { $eq: ["$hospital\_id", "$$hospital\_id"] } } },

{

$group: {

\_id: "$rol",

cantidad: { $sum: 1 }

}

}

],

as: "personal\_por\_rol"

}

},

{

$lookup: {

from: "personal",

localField: "director\_general\_id",

foreignField: "\_id",

as: "director\_info"

}