

Sistema de Gestión Policlínico

Equipo 3

Diego Hernández Rodríguez C311 @eu_sou_diegueinho

Sammy Raul Sosa Justiz C312 @Sosa_Samy

Daniela De La Caridad Guerrero Álvarez C311 @DGA47

Rubén Martínez Rojas C311 @Nebur02

15 de diciembre de 2025



Índice

1. Diccionario de datos	3
2. Patrones de Diseño, Visualización y de Datos	5
3. Esquema de las clases definidas y su relacion	6
4. Arquitectura	7
5. Modelo de datos	8

1. Diccionario de datos

A continuacion el diccionario de datos que muestra los campos y las caracteristicas de estos campos de las entidades mas importantes del proyecto:

Personal de Salud

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Nombre	string	-	si	-
Codigo	string	6	si	2 primeras letras y los otros son numeros
Email	string	-	si	Debe decir @gmail al final
Rol	string	-	si	Debe ser alguno de los roles predefinidos
Activo	bool	-	si	-
Departamento id	string		no	-
Consultas	lista de consultas	-	no	-
Jefe de departamento	bool	-	no	-

Paciente

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Nombre	string	-	si	-
Id	number	-	si	-
Email	string	-	si	Debe decir @gmail al final
Fecha de nacimiento	fecha	-	si	-
Telefono	number	-	si	-
Historial Clinico	lista de consultas	-	no	-

Departamento

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Id	string	-	si	-
Nombre	string	-	si	-
Jefe de departamento id	string	-	si	-
Stock	lista de medicamentos	-	no	-
Consultas	lista de consultas	-	no	-
Remisiones	lista de remisiones	-	no	-
Historial de trabajadores	lista de trabajadores	-	si	-

Consulta

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Id	string	-	si	-
Doctor id	string	-	si	-
Departamento id	string	-	si	-
Estado	string	-	no	-
Diagnostico	string	-	no	-
Fecha de creacion	fecha	-	si	-
Prescripcion	string	-	no	-
Paciente id	number	-	no	-
Remision id	string	-	no	-
Fecha definida	fecha	-	si	-

Remision

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Id	string	-	si	-
Paciente id	number	-	si	-
Departamento destino	string	-	si	-
Fecha de creacion	fecha	-	si	-
Consulta	consulta	-	si	-
Departamento remisor id	string	-	no	-
Puesto medico id	string	-	no	-

Medicamento

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Id	string	-	si	-
Nombre	string	-	si	-
Codigo	string	-	si	-
Descripcion	string	-	si	-
Unidad	string	-	si	-
Prescripcion	string	-	no	-
Stock	stock	-	no	-

Stock

Nombre Campo	Tipo Dato	Longitud	Obligatorio	Restricciones
Id	string	-	si	-
Cantidad	number	-	si	-
Mínimo	number	-	no	-
Máximo	number	-	no	-
Medicamento	medicamento	-	no	-
Departamento id	string	-	si	-

2. Patrones de Diseño, Visualización y de Datos

2.1. Patrón Repository

Usado para abstraer la persistencia y aislar la base de datos del dominio. Cada módulo tiene su repositorio:

- consultations.repository.ts
- remissions.repository.ts
- patients.repository.ts

2.2. Patrón DTO (Data Transfer Object)

Para garantizar validación y contrato estricto entre frontend → backend. Ejemplo:

- CreatePatientDto
- CreateConsultationDto
- CreateRemissionDto

2.3. Patrón Dependency Injection

NestJS lo usa de forma nativa. Permite sustituir servicios, repositorios y mockear para testing.

2.4. Patrón Modularización

Toda funcionalidad está dividida por módulos independientes y escalables:

- consultations/
- remissions/
- patients/

2.5. Patrón Aggregates (DDD)

Aplicado especialmente en remissions → consultations para garantizar consistencia. Cada remisión actúa como un agregado que contiene varias consultas.

2.6. Patrones de Visualización (Frontend)

Basados en:

- Componentes reutilizables
- Servicios centralizados para llamadas HTTP
- Manejo de estado local y UI reactiva

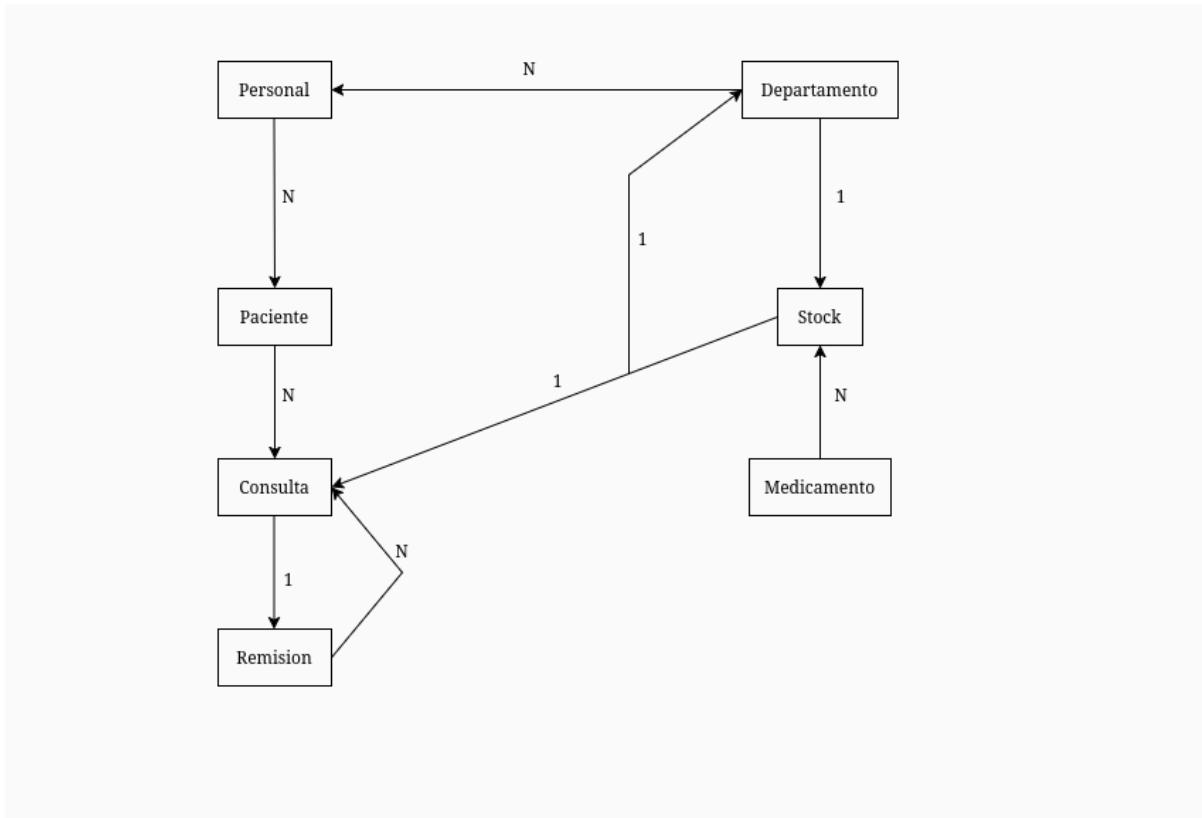
2.7. Patrones de Datos

Estructuración normalizada en PostgreSQL:

- Relaciones 1:N entre remisiones y consultas
- Integridad referencial garantizada con foreign keys
- Uso de UUID como identificadores únicos para escalabilidad

3. Esquema de las clases definidas y su relacion

Esquema en el cual se muestran las distintas clases definidas y la relacion entre ellas con sus respectivas cardinalidades:



4. Arquitectura

Nuestra solución utiliza una arquitectura Clean Architecture aplicada dentro del framework NestJS. Esta arquitectura separa de forma estricta las capas de dominio, aplicación e infraestructura para garantizar mantenibilidad, testabilidad y escalabilidad.

4.1. Capa de Presentación (Interfaces / Controllers)

Los controladores reciben las solicitudes HTTP, validan DTOs y delegan toda la lógica hacia los servicios. Ejemplo:

- consultations.controller.ts
- remissions.controller.ts
- patients.controller.ts

4.2. Capa de Aplicación (Servicios / Casos de Uso)

Los servicios contienen la lógica de negocio específica del caso de uso. Orquestan entidades, repositorios y reglas. Ejemplo:

- consultations.service.ts gestiona creación, actualización y consulta.
- remissions.service.ts maneja remisiones y su relación con consultas.

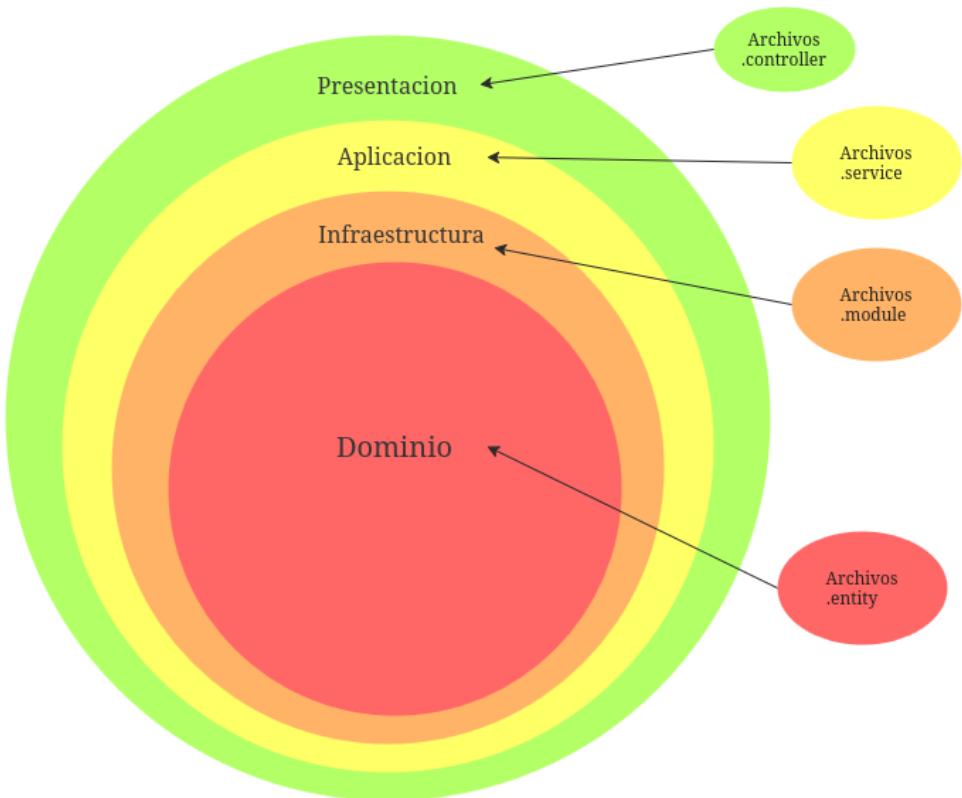
4.3. Capa de Dominio (Entidades)

Modela las reglas fundamentales del sistema:

- Consultation
- Remission
- Patient

4.4. Capa de Infraestructura (Repositorios y ORM)

Responsables de persistencia con TypeORM. Implementan interfaces de dominio. Esta separación permite reemplazar infraestructura sin afectar el dominio.



5. Modelo de datos

En esta sección se presenta el modelo entidad-relación que define la estructura de la base de datos del sistema. Este modelo representa las entidades principales, sus atributos y las relaciones entre ellas, garantizando la integridad y consistencia de los datos.

