ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO

Facultad de Informática y Electrónica

FACULTAD: INFORMÁTICA Y ELECTRONICA

CARRERA:

SOFTWARE

MATERIA:

APLICACIONES INFORMÁTICAS II

ESTUDIANTE:

JEANPIERRE DAVID AGUILAR ENCALADA – 7144

DOCENTE:

ING. JULIO ROBERTO SANTILLAN CASTILLO

NIVEL: OCTAVO SEMESTRE

PARALELO: "A"

SEPTIEMBRE 2025 – FEBRERO 2026

DISEÑO ARQUITECTÓNICO

Selección De La Arquitectura

1.1. Tipo de arquitectura adoptada

Se adopta un Monolito modular: una sola base de código/repositorio, organizada por dominios claramente separados en módulos. Cada módulo encapsula capas internas (interfaces, application, domain, infrastructure) y depende de contratos internos, no de detalles de framework. Esta organización permite un desarrollo y despliegue simples sin perder orden ni escalabilidad razonable.

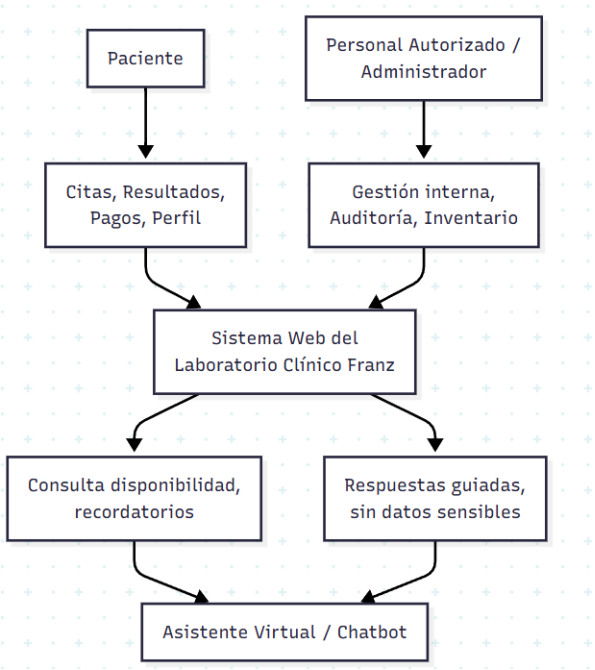
1.2. Capas lógicas dentro de cada módulo del backend

- interfaces/: controladores HTTP, DTOs y validación de entrada/salida.
- application/: casos de uso y servicios de aplicación
- domain/: entidades, agregados, value objects y reglas de negocio puras, sin dependencias de framework.
- infrastructure/: adaptadores técnicos (repositorios PostgreSQL por esquema: usuario, agenda, resultados, facturación, inventario, auditoría), almacenamiento S3 para PDFs, colas Redis/BullMQ, integraciones externas.

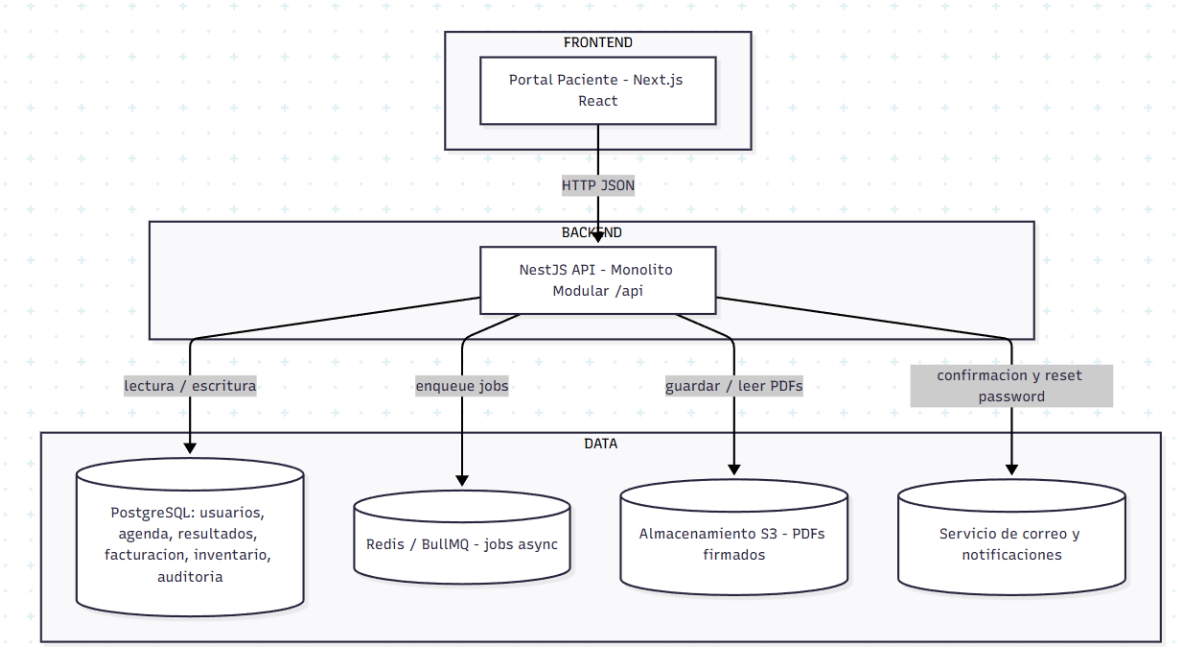
1.3. Relación frontend ↔ backend

El Portal Paciente (Next.js, App Router) consume endpoints REST expuestos por la API (NestJS). El Portal Paciente ofrece rutas principales: /dashboard, /citas, /resultados, /cotizaciones, /pagos, /perfil. La API valida identidad y rol en cada request, y registra eventos de auditoría para accesos y operaciones sensibles

Diagrama contextual



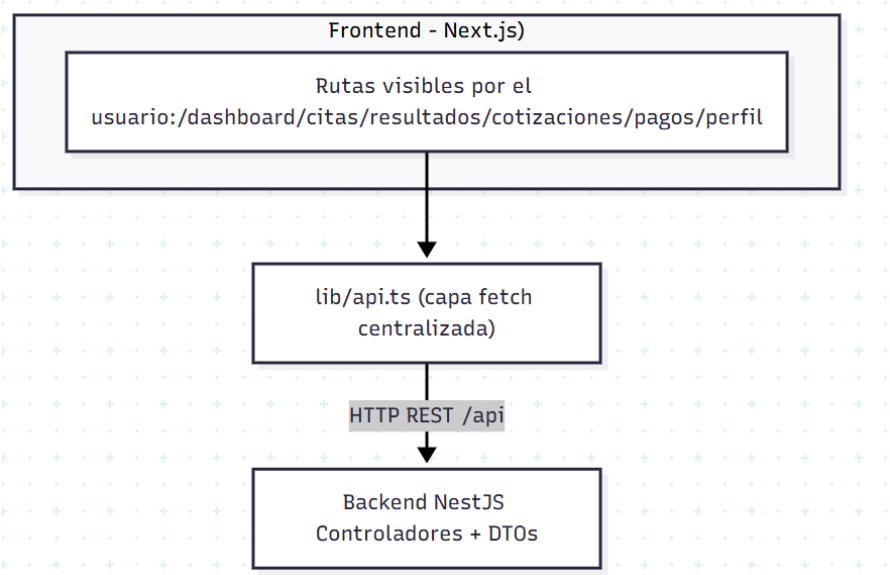
Arquitectura lógica: Frontend ↔ Backend ↔ Infra/Datos



Este diagrama muestra cómo el Portal Paciente (Next.js) consume la API REST del backend NestJS.

Dentro del backend, cada dominio (auth, citas, resultados, pagos, auditoría…) es como un módulo independiente, pero todos usan una capa de infraestructura compartida para acceder a Postgres, Redis/BullMQ, S3 y el servicio de correo.

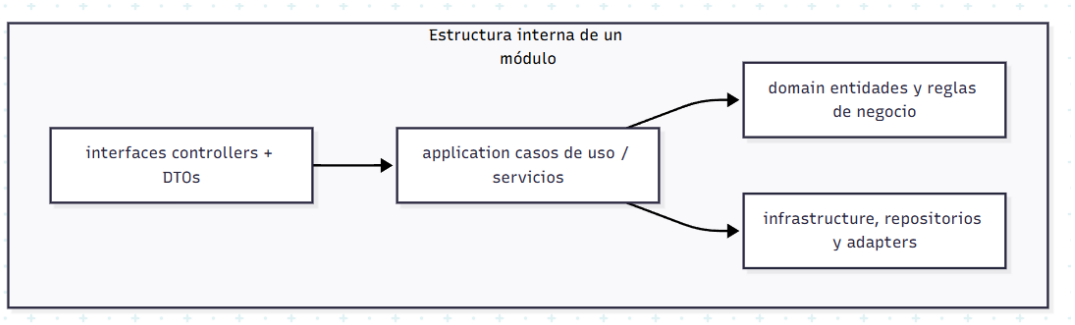
Frontend → capa fetch → API backend



Cada ruta no llama directamente a la base de datos. En su lugar, cada vista usa una capa de fetch (lib/api.ts) que centraliza las llamadas HTTP hacia la API NestJS.

.

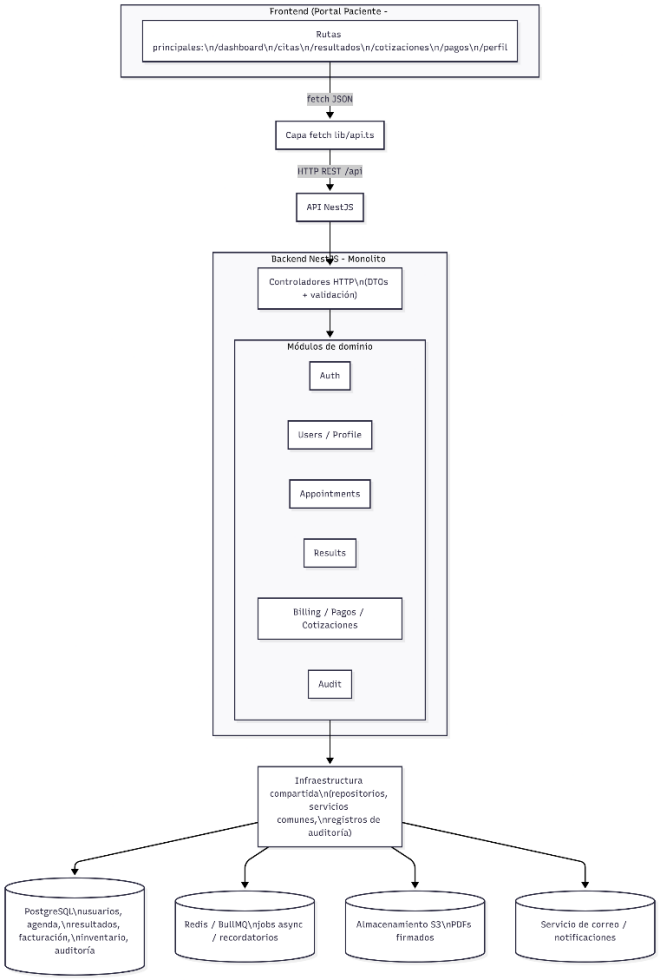
Capas internas de un modulo



Cada módulo del backend sigue la misma estructura interna.

- interfaces expone controladores HTTP y valida la entrada/salida.
- application contiene casos de uso y servicios orquestadores
- domain define las reglas de negocio puras y entidades.
- infrastructure implementa acceso técnico (repositorios Postgres, S3, Redis, correo) sin interrumpir la lógica de negocio.

Vista consolidada de la arquitectura lógica y dominios internos



El Portal Paciente (Next.js) expone rutas principales al usuario.

Esas rutas hablan con la capa fetch centralizada, que se comunica vía HTTP REST con la API NestJS.

Dentro del backend, cada dominio (Auth, Users/Profile, Appointments, Results, Billing/Pagos/Cotizaciones, Audit) esta como módulo separado con validación y controladores propios.

Todos los módulos reutilizan una infraestructura compartida (repositorios PostgreSQL, colas Redis/BullMQ para recordatorios/notificaciones, almacenamiento S3 para PDFs firmados y servicio de correo).

Estructura De Carpetas Propuesta

3.1 Backend

Estructura general del backend

Carpeta / archivo	Descripción	Responsable
software/backend/src/main.ts	Bootstrap NestJS, configuración inicial (pipes globales, CORS, prefijo /api).	Infraestructura
software/backend/src/app.module.ts	Módulo raíz que agrega módulos de dominio y common.	Arquitectura
software/backend/src/common/	Guards, decoradores, validadores, utilidades de cifrado (crypto), correo, auditoría.	Seguridad / Infraestructura
software/backend/src/modules/	Contenedor de los módulos de negocio.	Arquitectura

Módulo Auth

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/auth/	Módulo de autenticación y sesiones.	Seguridad
software/backend/src/modules/auth/auth.controller.ts + software/backend/src/modules/auth/dtos/	Controladores HTTP, DTOs y validación de entrada/salida.	Presentación
software/backend/src/modules/auth/auth.service.ts	Casos de uso: login, refresh / revoke de sesión, reset y cambio de contraseña.	Orquestación
software/backend/src/modules/auth/domain/	Entidades / valores (UsuarioAutenticado, Tokens), reglas puras de autenticación.	Negocio
software/backend/src/modules/auth/repos/ + software/backend/src/modules/auth/services/jwt.service.ts	Repositorios Postgres (usuarios / sesiones activas), emisión y validación JWT; integra infra/db.ts.	Infraestructura

Módulo Profile (UserProfile)

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/profile/	Módulo de perfil (datos del paciente / personal).	Negocio
software/backend/src/modules/profile/profile.controller.ts + software/backend/src/modules/profile/dtos/	Controladores HTTP y DTOs del perfil de usuario.	Presentación
software/backend/src/modules/profile/profile.service.ts	Casos de uso: actualizar datos personales, gestionar consentimientos, cambio de contraseña.	Orquestación
software/backend/src/modules/profile/domain/	Entidades y validaciones de dominio relacionadas al perfil.	Negocio
software/backend/src/modules/profile/infrastructure/	Repositorios Postgres (esquema usuario).	Infraestructura

Módulo Appointments

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/appointments/	Agenda y gestión de citas.	Negocio
software/backend/src/modules/appointments/appointments.controller.ts	Controladores HTTP y DTOs para citas.	Presentación
software/backend/src/modules/appointments/appointments.service.ts	Casos de uso: crear / cancelar cita, consultar disponibilidad.	Orquestación
software/backend/src/modules/appointments/domain/	Entidades Cita / Horario y reglas de disponibilidad.	Negocio

software/backend/src/modules/appointments/infrastructure/	Repositorios Postgres (esquema agenda), integración con colas de recordatorios.	Infraestructura

Módulo Results

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/results/	Resultados clínicos y descargas de PDF firmados.	Negocio
software/backend/src/modules/results/results.controller.ts	Controladores HTTP y DTOs para consulta y descarga de resultados.	Presentación
software/backend/src/modules/results/results.service.ts	Casos de uso: listar/ver resultados, generar URL firmada del PDF, registrar auditoría de acceso.	Orquestación
software/backend/src/modules/results/domain/	Entidades Resultado / Estudio y reglas de acceso.	Negocio
software/backend/src/modules/results/infrastructure/	Repositorios Postgres (esquema resultados), acceso a almacenamiento tipo S3 para PDFs firmados.	Infraestructura

Módulo Billing (Payments)

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/billing/	Facturación y pagos (cotizaciones, órdenes de pago, estados).	Negocio
software/backend/src/modules/billing/billing.controller.ts	Controladores HTTP y DTOs de cotizaciones / pagos.	Presentación

software/backend/src/modules/billing/billing.service.ts	Casos de uso: generar cotización, iniciar pago, confirmar pago.	Orquestación
software/backend/src/modules/billing/domain/	Entidades Orden / Cotización / Transacción.	Negocio
software/backend/src/modules/billing/infrastructure/	Repositorios Postgres (esquema facturacion), integraciones con pasarela de pago y colas.	Infraestructura

Módulo Inventory

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/inventory/	Inventario de insumos, reactivos, lotes, caducidad.	Negocio
software/backend/src/modules/inventory/interfaces/	Controladores y DTOs públicos del módulo de inventario.	Presentación
software/backend/src/modules/inventory/application/	Casos de uso: ajustes de stock, entradas/salidas, alertas de mínimos o caducidad.	Orquestación
software/backend/src/modules/inventory/domain/	Entidades Item / Lote / Stock y políticas de reposición / alerta.	Negocio
software/backend/src/modules/inventory/infrastructure/	Repositorios Postgres (esquema inventario) y colas de notificación.	Infraestructura

Módulo Audit

Carpeta / archivo	Descripción	Responsable
software/backend/src/modules/audit/	Auditoría append-only (registro de accesos, descargas, cambios críticos).	Seguridad
software/backend/src/modules/audit/audit.controller.ts	Endpoints de consulta de auditoría (solo para roles autorizados).	Presentación

software/backend/src/modules/audit/audit.service.ts	Registrar accesos / descargas / ediciones y exponer consultas auditables.	Orquestación
software/backend/src/modules/audit/domain/	Entidades EventoAuditoria y reglas de inmutabilidad (no se edita el histórico).	Negocio
software/backend/src/modules/audit/infrastructure/	Repositorios Postgres (esquema auditoria) y escritura asíncrona/colas.	Infraestructura

Módulo Chatbot

Carpeta / archivo	**Descripción**	**Responsable**
software/backend/src/modules/chatbot/	Lógica de interacción del asistente virtual / chatbot.	Negocio
software/backend/src/modules/chatbot/interfaces/	Controladores / DTOs para webhooks y mensajes del chatbot.	Presentación
software/backend/src/modules/chatbot/application/	Enrutamiento de intenciones, llamadas a otros módulos internos (sin exponer datos sensibles).	Orquestación
software/backend/src/modules/chatbot/domain/	Reglas conversacionales y políticas de privacidad / no divulgación de datos clínicos.	Negocio
software/backend/src/modules/chatbot/infrastructure/	Adaptadores externos (proveedor de IA / mensajería), colas de mensajes.	Infraestructura

3.2 Frontend

Carpeta / archivo	Descripción	Responsable
software/frontend/app/	Rutas del Portal Paciente (App Router).	Presentación
software/frontend/app/(portal)/dashboard/	Pantalla principal con KPIs y accesos rápidos.	Presentación
software/frontend/app/(portal)/citas/	Gestión de citas (listar, crear, cancelar, reprogramar).	Presentación
software/frontend/app/(portal)/resultados/	Listado/visualización de resultados clínicos y descarga de PDFs firmados.	Presentación
software/frontend/app/(portal)/cotizaciones/	Generación y consulta de cotizaciones de análisis.	Presentación
software/frontend/app/(portal)/pagos/	Flujo de pagos y estados (pendiente, pagado, vencido).	Presentación
software/frontend/app/(portal)/perfil/	Perfil del paciente: datos personales, contacto, consentimiento, cambio de contraseña.	Presentación
software/frontend/components/	UI reutilizable: navegación lateral/topbar, tarjetas/KPI, tablas, formularios, badges de estado, modales.	Presentación
software/frontend/lib/	Cliente HTTP centralizado (api.ts), manejo de sesión/estado, utilidades compartidas.	Integración
software/frontend/app/globals.css	Estilos globales base (tokens de color, espaciados, tipografía vía Tailwind/utilidades).	Diseño
software/frontend/app/design/tokens.ts	Tokens de diseño (paleta, espaciado, radios, etc.) declarados en TypeScript.	Diseño
software/frontend/hooks/	Hooks que encapsulan formularios (react-hook-form + Zod), estados locales, validaciones y helpers de UI.	Presentación

Checklist de verificación (Backend + Frontend)

Ítem a verificar	**Ruta esperada (ajustada al repo)**	**¿Cumple? (Sí/No/N/A)**	**Observación breve**
Existe app.module.ts en la API	software/backend/src/app.module.ts	Sí	Importa auth, users, appointments, results, billing, audit, profile.
Existe main.ts en la API	software/backend/src/main.ts	Sí	Configura CORS, prefijo /api, ValidationPipe.
Existe carpeta common/ con utilidades compartidas	software/backend/src/common/	Sí	Guards, decoradores, validators, crypto, mail.
Módulos de negocio dentro de modules/	software/backend/src/modules/	Sí	auth, profile, appointments, results, billing, audit, users.
Para cada módulo existen interfaces / application / domain / infrastructure	software/backend/src/modules/*/...	No	Patrón parcial: controller+dtos, service, repos; algunas capas no están todavía separadas explícitamente.
Los controladores HTTP **NO** acceden directo a BD	software/backend/src/modules//.controller.ts	Sí	Controladores llaman servicios; acceso a BD va vía services/repos.
Lógica de negocio en domain/ sin depender de Nest	software/backend/src/modules/*/domain/	No	domain/ no está definido en todos los módulos; parte de la lógica vive aún en services.
Infraestructura técnica en infrastructure/	software/backend/src/modules/*/infrastructure/	No	Repositorios están en repos/ y infra/db.ts; integración

(repos, S3, colas)			S3/colas aún en desarrollo.
Portal Paciente con rutas principales	software/frontend/app/(portal)/...	Sí	Existen dashboard, citas, resultados, cotizaciones, pagos, perfil.
Componentes reutilizables de UI	software/frontend/components/	Sí	Varias subcarpetas (ui, tables, profile, etc.).
Cliente HTTP y helpers de sesión / auth	software/frontend/lib/	Sí	api.ts, stores de sesión/toast.
Tokens de estilo global	software/frontend/app/globals.css y software/frontend/app/design/tokens.ts	Sí	No hay carpeta styles/ separada; tokens están centralizados ahí.
Accesibilidad (aria-label / texto en acciones sensibles)	software/frontend/components/	N/A	.
Auditoría: módulo y política operativa	software/backend/src/modules/audit/	Sí	Servicio registra eventos en esquema/tabla de auditoría append-only.

Checklist de rutas (Frontend – Portal Paciente)

Ruta	**¿Existe?**
software/frontend/app/(portal)/dashboard/page.tsx	Sí
software/frontend/app/(portal)/citas/page.tsx	Sí
software/frontend/app/(portal)/resultados/page.tsx	Sí
software/frontend/app/(portal)/cotizaciones/page.tsx	Sí
software/frontend/app/(portal)/pagos/page.tsx	Sí
software/frontend/app/(portal)/perfil/page.tsx	Sí

Justificación De Decisiones Técnicas

- La solución se construye con una arquitectura monolítica modular porque mantiene un orden interno. Un único despliegue reduce la complejidad operativa (infraestructura, monitoreo, versiones), mientras que la separación en módulos lógicos mantiene las responsabilidades definidas.
- El código vive en un único repositorio (monorepo) que agrupa frontend, backend y recursos de soporte. Con esto todo el sistema evoluciona de forma coordinada: cuando se cambia una regla de negocio o una validación, se actualiza en un solo lugar y se valida de extremo a extremo.
- La arquitectura interna sigue una separación en capas para proteger la lógica de negocio. Las reglas funcionales importantes no dependen directamente de frameworks ni de detalles de base de datos. Eso hace que el comportamiento crítico se pueda probar de forma aislada, sea más fácil de auditar y más fácil de modificar sin romper todo el sistema.
- Las integraciones técnicas como base de datos, almacenamiento de archivos firmados, cola de trabajos asíncronos o envío de notificaciones están encapsuladas detrás de interfaces. Esta decisión busca estabilidad a largo plazo: el sistema puede crecer en volumen (más citas, más resultados) sin tener que rediseñar la lógica central cada vez que se ajusta la infraestructura.
- Finalmente, la arquitectura prioriza seguridad y trazabilidad desde el diseño. El acceso a información sensible pasa por autenticación fuerte y control de permisos, y toda acción importante queda registrada. Esto no es solo un requisito técnico: es una medida explícita para proteger datos clínicos, respaldar auditorías internas y generar confianza en el uso del sistema