



ESPECIALIZACION DE INGENIERIA DE SOFTWARE

PATRONES DE DISEÑO DE SOFTWARE

INTEGRASALUD: IDEAL PARA UNA SOLUCIÓN QUE UNIFICA ÁREAS

CLÍNICAS, ADMINISTRATIVAS Y LEGALES.



INTEGRASALUD

IDEAL PARA UNA SOLUCIÓN QUE UNIFICA ÁREAS CLÍNICAS, ADMINISTRATIVAS Y LEGALES

ROMARIO ALDAIR MARTINEZ

LUIS ALEJANDRO FUENTES CORZO

MIGUEL JOSE MENDOZA SAMPER

MANUEL CABRERA

Fecha: 20/06/2025

TABLA DE CONTENIDO

CONTENIDO

1. INTRODUCCIÓN	3
2. PLANTEAMIENTO DEL PROBLEMA	4
3. JUSTIFICACIÓN	5
4. OBJETIVO GENERAL.....	7
Objetivos Específicos.....	7
5. ANÁLISIS DEL ESCENARIO Y DEFINICIÓN DE REQUERIMIENTOS:.....	9
5.1. REQUERIMIENTOS FUNCIONALES.....	9
5.2. REQUERIMIENTOS NO FUNCIONALES	11
6. DISEÑO DE LA ARQUITECTURA DE LA SOLUCIÓN	14
6.1. Tipo de Arquitectura Seleccionada: Microservicios con enfoque modular.....	14
6.3. Aplicación de patrones de diseño	15
6.3.1. Patrones Creacionales	15
6.3.2Patrones Estructurales	16
6.3.3 Diagrama de clases	19
6.3.4. Diagrama de arquitecturas y componentes	19
7. IMPLEMENTACIÓN DE LA SOLUCIÓN.....	20
7.1. Stack Tecnológico Seleccionado	21
7.2. Integración con Patrones de Diseño – Spring Boot	22
7.3. Explicación detallada de los Patrones Utilizados	23
7.3.1 Patrones Creacionales	23
7.3.2. Patrones Estructurales	24
8. Estructura de Carpetas	25
9. LECCIONES APRENDIDAS	29
CONCLUSIÓN.....	31
11. BIBLIOGRAFIA	32

1.INTRODUCCIÓN

En Colombia, trabajar en una IPS, ya sea un consultorio, una clínica o un centro médico— significa enfrentarse todos los días a múltiples retos. No solo se trata de atender pacientes, también hay que organizar historias clínicas, cumplir con normativas, generar reportes, gestionar la farmacia, coordinar al personal, facturar, enviar información a las EPS... y todo eso, muchas veces, con herramientas limitadas o incluso con procesos manuales.

Mientras otros sectores ya han dado pasos firmes hacia la digitalización, muchas instituciones de salud siguen trabajando como hace décadas. El papel se acumula, los errores se repiten y los tiempos se alargan, afectando tanto al personal de salud como a los pacientes. Y aunque existen sistemas en el mercado, muchos son costosos, difíciles de usar o simplemente no se ajustan a la realidad de las IPS colombianas.

Frente a esta situación, nace la necesidad de una solución que no solo sea tecnológica, sino también humana, útil y localmente relevante. Una herramienta que entienda cómo funciona una IPS por dentro, que se adapte a su ritmo, que cumpla con la ley, y que al mismo tiempo, ayude a mejorar la calidad del servicio y facilite el trabajo del equipo médico.

Ese es el propósito de SaludConecta: una plataforma integral que reúne en un solo lugar todos los procesos clínicos y administrativos que una IPS necesita, desde la historia clínica electrónica y la telemedicina, hasta la facturación y los reportes regulatorios. Todo diseñado con

un enfoque claro: que la tecnología no sea una barrera, sino un puente para brindar una atención más organizada, segura y humana.

Este proyecto busca más que digitalizar procesos. Quiere acompañar a las IPS en su transformación, hacer más fácil lo complejo, y poner la tecnología al servicio de quienes cuidan nuestra salud todos los días.

2. PLANTEAMIENTO DEL PROBLEMA

En Colombia, muchas IPS, especialmente las pequeñas clínicas, consultorios o centros médicos ubicados fuera de las grandes ciudades, enfrentan serias dificultades para manejar su operación diaria. Aunque su propósito es brindar atención médica de calidad, la falta de herramientas digitales adecuadas termina afectando tanto a los profesionales como a los pacientes.

Hoy en día, todavía es común encontrar historias clínicas en papel, agendas médicas en Excel, y procesos de facturación hechos a mano. Esto no solo consume más tiempo y esfuerzo del personal de salud, sino que también genera errores, retrasos, pérdida de información y, en algunos casos, hasta sanciones por no cumplir con la normatividad exigida por el Ministerio de Salud o la DIAN.

Muchos médicos no pueden acceder fácilmente al historial completo de un paciente. Los reportes de RIPS se hacen con dificultad o con errores, y ofrecer telemedicina sigue siendo un reto, porque no hay una plataforma que lo permita de forma legal, segura y práctica. Todo esto ocurre

mientras crece la presión por modernizar el sistema y brindar una atención más rápida, segura y eficiente.

El problema no es la falta de intención por parte de las IPS. El verdadero problema es que no cuentan con una solución tecnológica completa, accesible y diseñada específicamente para su realidad. Las herramientas que existen suelen ser costosas, difíciles de usar o no cumplen con los requerimientos legales y clínicos de nuestro país.

Por eso nace la necesidad de crear una plataforma que realmente entienda a las IPS colombianas: que les permita organizar su trabajo, cuidar mejor a sus pacientes, cumplir con la ley, y dar un salto a la transformación digital sin tener que volverse expertos en tecnología.

3. JUSTIFICACIÓN

El acceso a una atención médica digna, eficiente y bien organizada es un derecho fundamental, pero para muchas IPS en Colombia, lograrlo es un reto diario. Aunque la tecnología ha avanzado, gran parte del sector salud aún depende de procesos manuales, herramientas dispersas y soluciones que no responden a su realidad operativa. Esto genera desgaste en los profesionales, pérdida de tiempo, errores en la atención y, lo más preocupante, pone en riesgo la seguridad y continuidad del cuidado del paciente.

Mientras tanto, las exigencias normativas crecen: reportes a EPS, facturación electrónica ante la DIAN, protección de datos personales, historias clínicas digitales, entre otros. Cumplir con todo esto no debería ser una carga extra, sino parte de un sistema que ayude, facilite y acompañe.

Aquí es donde cobra sentido SaludConecta: una plataforma pensada desde las verdaderas necesidades del día a día en una IPS. Este proyecto no busca solo digitalizar papeles, sino crear una solución integral que acompañe al médico, al paciente y al administrador en cada paso, desde la cita médica hasta el cierre contable del mes.

Además, SaludConecta nace con visión social. No es solo para grandes clínicas privadas; es una herramienta que puede llegar a regiones rurales, consultorios independientes y centros de salud comunitarios. Al ser modular, adaptable y fácil de usar, cualquier institución podrá empezar con lo básico y crecer con el tiempo.

En un país donde muchas veces la salud depende de la voluntad de personas que hacen más con menos, este proyecto busca ser un verdadero aliado, haciendo que la tecnología esté al servicio de la salud, y no al revés.

4. OBJETIVO GENERAL

Crear una plataforma digital que ayude a las IPS colombianas a organizar y mejorar todos sus procesos médicos y administrativos, desde la historia clínica hasta la facturación, de una manera fácil, segura y legal. La idea es que puedan enfocarse en lo más importante: cuidar la salud de sus pacientes, mientras el sistema se encarga del resto.

Objetivos Específicos

Escuchar las necesidades reales de las IPS y entender cómo trabajan, qué herramientas usan actualmente y dónde se generan los mayores dolores operativos.

Definir claramente qué funciones necesita el sistema, tanto en la parte médica (como la historia clínica y las recetas) como en la parte administrativa (como la facturación o los reportes para EPS).

Diseñar un sistema amigable, sencillo y adaptable, que se pueda usar desde cualquier dispositivo, sin necesidad de tener conocimientos técnicos avanzados.

Cumplir con todas las leyes y normas colombianas, como la protección de datos personales, la historia clínica electrónica y la facturación ante la DIAN.

Incluir herramientas modernas como la telemedicina, permitiendo que los médicos puedan atender virtualmente a sus pacientes, con respaldo legal y técnico.

Ofrecer indicadores e informes que ayuden a tomar decisiones, como saber cuántas citas se atienden por semana, qué medicamentos se usan más o qué servicios son más solicitados.

Construir una plataforma que pueda crecer con cada IPS, es decir, que funcione para una clínica pequeña, pero que también se adapte si la entidad abre más sedes o aumenta su volumen de atención.

5. ANÁLISIS DEL ESCENARIO Y DEFINICIÓN DE REQUERIMIENTOS:

5.1. REQUERIMIENTOS FUNCIONALES

ID	Requerimiento Funcional	Descripción
RF1	Historia clínica electrónica	Registro detallado de antecedentes, diagnósticos, evoluciones y tratamientos.
RF2	Agenda y citas	Agendamiento por médico, especialidad, sede y horario, con recordatorios.
RF3	Registro de pacientes	Ingreso de datos básicos, contacto, EPS, historial, etc.
RF4	Autorización de servicios	Registrar y validar servicios con número de autorización.
RF5	Facturación electrónica	Generación automática de factura conforme a DIAN.
RF6	Emisión de RIPS	Exportación de archivos RIPS por servicio, sede y periodo.
RF7	Inventario de medicamentos	Control de stock, vencimientos, y salidas por fórmula.
RF8	Prescripción médica	

		Recetas electrónicas con firma del médico y código QR.
RF9	Gestión de pagos y cartera	Control de pagos, cuentas por cobrar y cartera vencida.
RF10	Carga y resultados de laboratorio	Registro manual o automático de exámenes y análisis.
RF11	Imágenes diagnósticas	Enlace a estudios de rayos X, ecografías, etc., con visor opcional.
RF12	Telemedicina integrada	Consulta por videollamada, con historia y prescripción en línea.
RF13	Módulo de enfermería	Registro de signos vitales, cuidados, administración de medicamentos.
RF14	Referencia y contrarreferencia	Gestión de pacientes derivados a otras entidades.
RF15	Reportes estadísticos	Generación de gráficos y KPIs por mes, sede o especialidad.

RF16	Panel de administración	Control de usuarios, configuraciones, parámetros del sistema.
RF17	Gestión de usuarios por rol	Permisos diferenciados: médico, auxiliar, administrativo, etc.
RF18	Módulo de consentimiento informado	Firma digital de autorizaciones clínicas o legales.
RF19	Atención domiciliaria	Registro de visitas domiciliarias y controles en campo.
RF20	Registro de incapacidades y certificados	Emisión y firma de incapacidades, certificados médicos y constancias.

5.2. REQUERIMIENTOS NO FUNCIONALES

ID	Requerimiento No Funcional	Descripción
RNF1	Seguridad	Cifrado de datos, roles y doble autenticación.

RNF2	Alta disponibilidad	Funcionar 24/7 con respaldo y tolerancia a fallos.
RNF3	Interfaz intuitiva	Diseño limpio, con navegación simple y accesible.
RNF4	Escalabilidad	Soportar desde una sede hasta redes médicas con múltiples usuarios.
RNF5	Portabilidad	Acceso desde PC, tablet o celular sin pérdida de funcionalidad
RNF6	Interoperabilidad	Compatible con EPS, FOSYGA, ADRES, HL7, FHIR.
RNF7	Cumplimiento legal	Acorde con leyes colombianas (Ley 1581, Ley 527, Resol. 1995).
RNF8	Auditoría de accesos	Registro de entradas, modificaciones y firmas.
RNF9	Actualizaciones automáticas	Nuevas versiones sin afectar la operación en curso.

RNF10	Tiempo de respuesta <2s	Rapidez para abrir historiales, cargar módulos, etc.
RNF11	Múltiples idiomas	Posibilidad de configurar idioma (español, inglés, etc.).
RNF12	Trazabilidad	Rastro completo de acciones realizadas por cada usuario.
RNF13	Soporte técnico integrado	Acceso a chat o ticket desde la misma aplicación.
RNF14	Backups automáticos	Copias de seguridad frecuentes para recuperación ante fallos.
RNF15	Módulo offline básico	Acceso parcial a historia y citas en zonas sin internet.
RNF16	Integración con correo/SMS	Envío automático de recordatorios o reportes.
RNF17	Restricción geográfica	Bloqueo de accesos sospechosos desde ubicaciones no permitidas.
RNF18	Compatibilidad con firma digital	Certificados de firma digital legalmente válidos en Colombia.

RNF19	Modularidad	Activar o desactivar módulos según necesidad de cada IPS.
RNF20	Tiempo de inactividad programado <1% mensual	Mantenimiento sin afectar la operación crítica.

6. DISEÑO DE LA ARQUITECTURA DE LA SOLUCIÓN

6.1. Tipo de Arquitectura Seleccionada: Microservicios con enfoque modular

Justificación de la arquitectura seleccionada:

Se elige una arquitectura basada en microservicios por las siguientes razones:

Permite escalabilidad independiente de cada módulo: historia clínica, citas, farmacia, etc.

Facilita el despliegue continuo y actualizaciones sin afectar todo el sistema.

Brinda resiliencia: un fallo en un servicio (por ejemplo, facturación) no detiene la plataforma completa.

Es ideal para entornos multi-sede o IPS en crecimiento.

Permite el uso de diferentes tecnologías para cada módulo si se desea (por ejemplo, Python para IA en diagnósticos, Node.js para backend rápido, etc.).

Cada módulo se comunica a través de APIs REST o gRPC, y está orquestado mediante una API Gateway central.

6.2. Componentes principales del sistema

Los microservicios de la plataforma estarían organizados de la siguiente forma:

- Módulo de autenticación y gestión de usuarios
- Módulo de historia clínica electrónica (HCE)
- Módulo de agendamiento de citas
- Módulo de farmacia e inventario
- Módulo de laboratorio e imágenes
- Módulo de facturación y RIPS
- Módulo de reportes e indicadores
- Módulo de telemedicina
- Módulo de notificaciones (correo/SMS)
- Frontend Web/Móvil y API Gateway
- Base de datos distribuida y replicada por servicio

6.3. Aplicación de patrones de diseño

6.3.1. Patrones Creacionales

Patrón	Aplicación	Justificación técnica

Factory Method	Creación de formularios médicos personalizados según la especialidad.	Permite instanciar formularios dinámicos con campos específicos.
Abstract Factory	Generador de componentes de interfaz para cada rol (médico, auxiliar, admin).	Encapsula el conjunto de elementos gráficos según el tipo de usuario.
Singleton	Manejador global de configuración del sistema (parámetros del sistema, tokens, etc).	Asegura que exista una única instancia accesible globalmente.
Builder	Construcción paso a paso de una historia clínica (física, antecedentes, órdenes).	Mejora la legibilidad y control de estructuras complejas

6.3.2 Patrones Estructurales

Patrón	Aplicación	Justificación técnica
		17
Adapter	Adaptación de formatos de datos externos (EPS, RIPS, HL7).	Permite integrar fuentes heterogéneas sin modificar el código base.
Bridge	Separar la lógica de notificación (correo/SMS) del canal de salida.	Permite cambiar proveedores sin alterar la lógica central.
Composite	Formularios médicos con subcomponentes reutilizables (antecedentes, signos vitales, etc).	Modelo jerárquico reutilizable y flexible.
Decorator	Añadir funcionalidades extra a un historial (firma, validación, exportación).	Mejora progresiva sin herencia rígida.
Proxy	Validación de acceso a historias clínicas sensibles	Controla accesos y delega lógica según permisos.

--	--	--

La decisión de implementar todos los patrones de diseño creacionales y estructurales no se tomó únicamente por cumplir un requisito académico, sino como una oportunidad para aprender, practicar y demostrar cómo cada uno puede aportar valor real en un sistema de información complejo como INTEGRASALUD.

En el contexto de un sistema médico con múltiples módulos (usuarios, citas, historia clínica, farmacia, notificaciones, etc.), es común enfrentar una gran variedad de problemas de diseño, acoplamiento, reutilización, mantenimiento y extensibilidad. Aplicar distintos patrones permitió:

Simular escenarios reales del desarrollo profesional, donde se busca un software limpio, robusto y escalable.

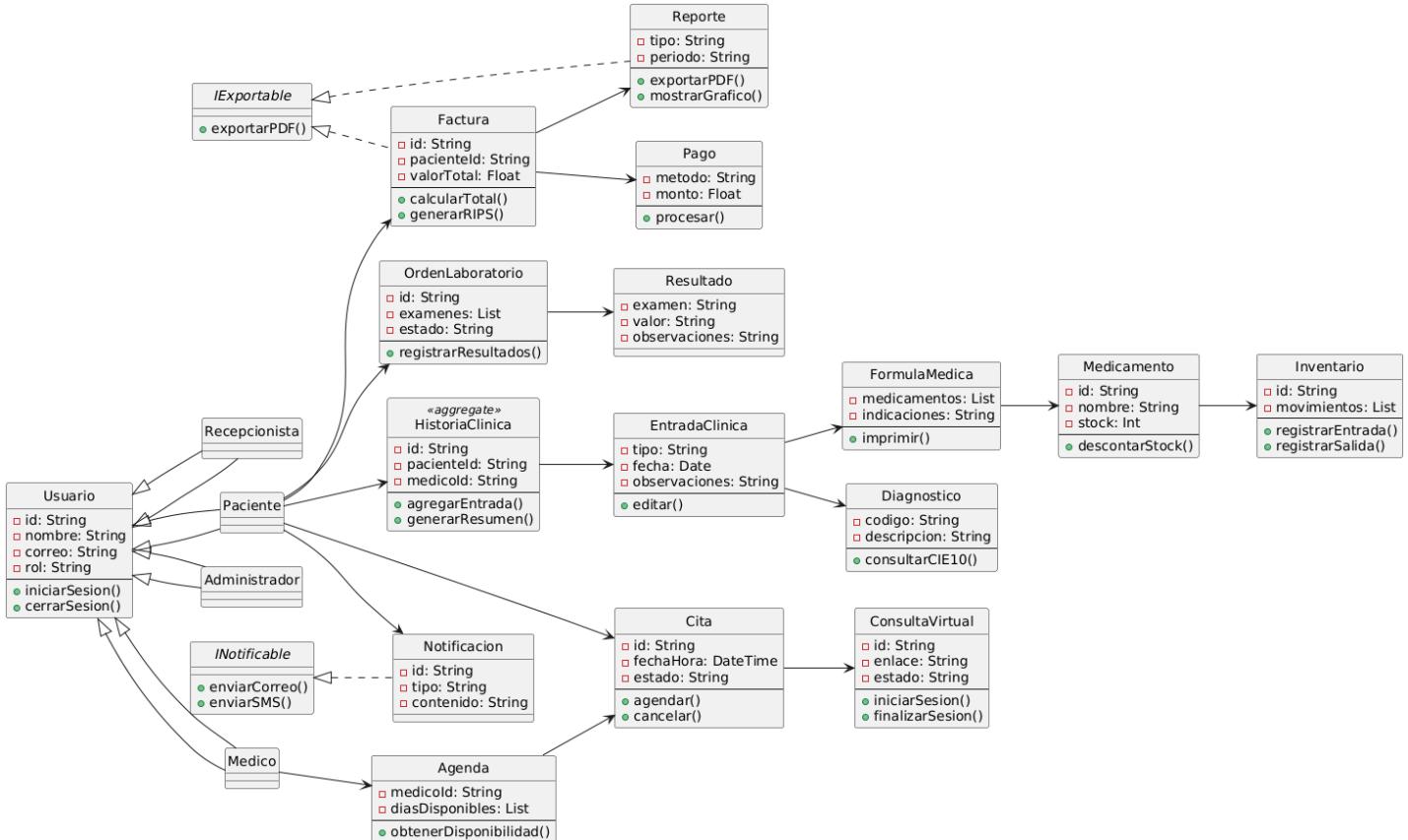
Elegir el patrón correcto para cada necesidad concreta, como controlar el acceso a datos sensibles , permitir la expansión de formularios clínicos o simplificar el envío de notificaciones

Mostrar dominio de buenas prácticas de ingeniería de software, lo que fortalece el proyecto tanto técnica como académicamente.

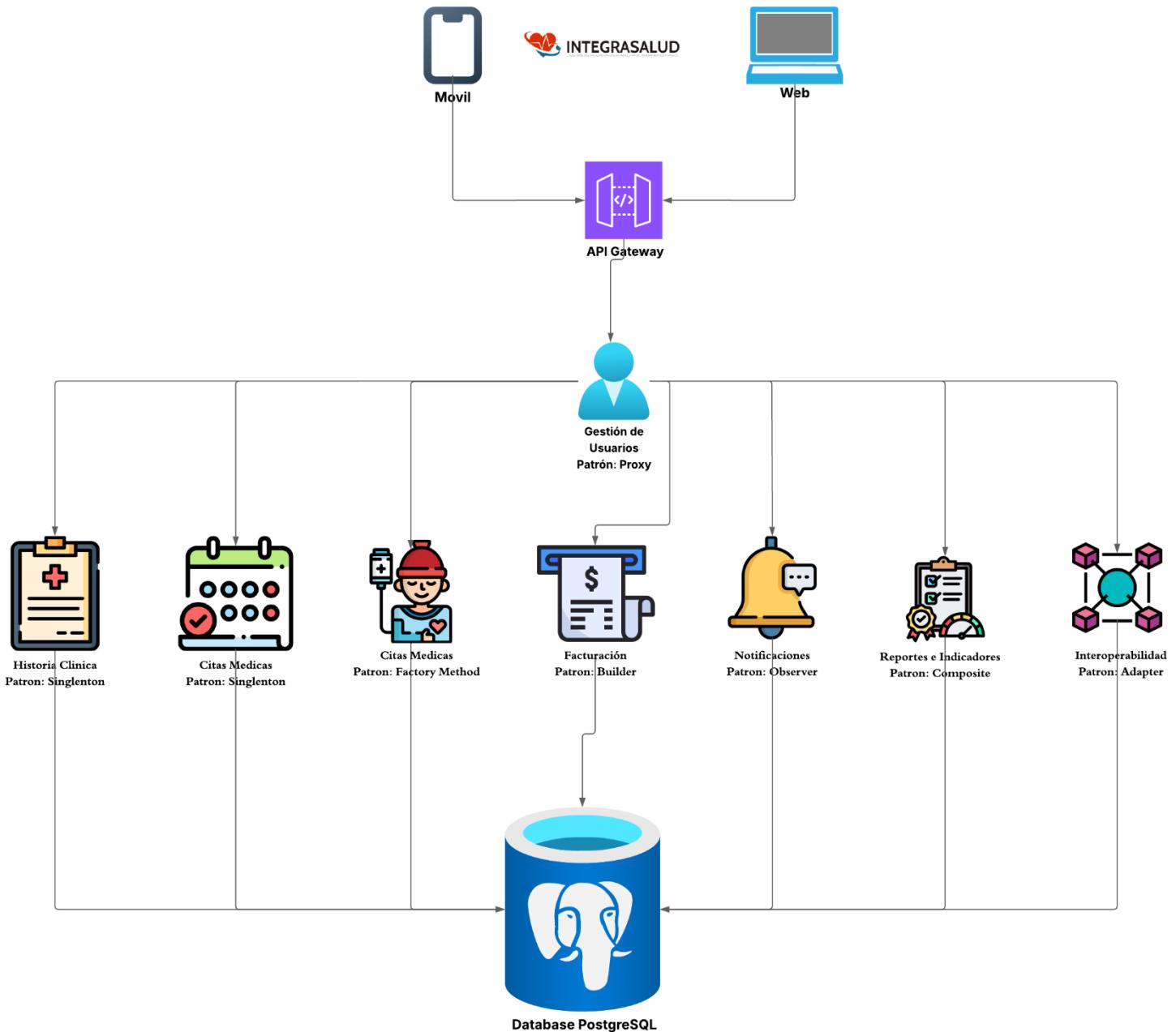
Reforzar el aprendizaje mediante el uso práctico de conceptos que muchas veces solo se estudian en la teoría.

Además, en un proyecto académico o de formación, tener la oportunidad de demostrar el uso adecuado de todos los patrones del curso no solo muestra conocimiento, sino que deja claro que se entienden sus ventajas, limitaciones y el contexto correcto para aplicarlos.

6.3.3 Diagrama de clases



6.3.4 Diagrama de Componentes



7.1. Stack Tecnológico Seleccionado

Componente	Herramienta /Tecnología	Justificación técnica
Lenguaje backend	Java 21	Potente, escalable, y compatible con patrones OO.
Framework backend	Spring Boot 3.x	Soporte completo para REST, seguridad, inyección de dependencias, y patrones de diseño
Frontend Web	React.js 18	Componentización, velocidad, comunidad activa.
Frontend Móvil	React Native + Expo	Código compartido multiplataforma Android/iOS.
Base de datos	PostgreSQL 15	Robusta, relacional, ideal para sistemas médicos.
ORM	Spring Data JPA (Hibernate)	Gestión eficiente de entidades con mapeo ORM.

Gestor de dependencias	Maven	Estándar en proyectos Java empresariales.
Contenedores	Docker	Portabilidad y despliegue reproducible.
Control de versiones	Git + GitHub	Flujo de trabajo colaborativo y CI/CD.

7.2. Integración con Patrones de Diseño – Spring Boot

Spring Boot permite aplicar los patrones creacionales y estructurales de manera natural gracias a sus características nativas:

Funcionalidad	Soporte del framework
Inyección de dependencias	@Autowired, @Component, @Service, @Repository
Gestión de ciclo de vida	Contenedor de beans y contexto de aplicación
Singleton automático	Por defecto, todos los beans son singleton
Configuración y modularidad	Archivos application.yml, perfiles, auto-configuration

7.3. Explicación detallada de los Patrones Utilizados

7.3.1 Patrones Creacionales

Patrón	Uso en el sistema	Integración con Spring
Singleton	Beans de servicios (ej. NotificacionService)	Spring lo gestiona por defecto
Factory Method	Creación de objetos de notificación (Correo, SMS) según tipo	Servicio con método crearNotificacion(tipo)
Abstract Factory	Generación de formularios dinámicos por especialidad médica	FormularioFactory.getFactory("Ginecología")
Builder	Construcción compleja de historial clínico paso a paso	Uso de clase HistoriaClinicaBuilder externa al controller

7.3.2. Patrones Estructurales

Patrón	Uso en el sistema	Justificación
Adapter	Adaptar formatos de entrada (por ejemplo, carga HL7 de laboratorios externos)	Convierte input externo a DTO interno
Bridge	Separar lógica de envío de notificaciones y su implementación (Correo, SMS, WhatsApp)	Notificacion define la abstracción, CorreoService, SMSService la implementación
Composite	Formularios médicos con secciones reutilizables (antecedentes, signos vitales, etc.)	Composición de FormularioComponent con Seccion, Campo
Decorator	Agregar funcionalidades a una receta sin modificar su estructura original	RecetaBase + RecetaConValidacion, RecetaConFirma
Proxy	Control de acceso a historia clínica sensible	HistoriaClinicaProxy valida permisos antes de delegar al repositorio

8. Estructura de Carpetas

Integrasalud

```
|  
|   └── 🏷 auth          # Módulo de autenticación y seguridad  
|       |   └── controller      # Controladores REST para login/autenticación  
|       |       └── AuthController.java  
|       |   └── service        # Lógica de negocio de autenticación  
|       |       └── AuthService.java  
|       |   └── 🔒 security      # Configuración de seguridad y JWT  
|       |       └── JwtTokenProvider.java  
|       |       └── JwtFilter.java  
|       |       └── WebSecurityConfig.java  
|       └── 👤 model         # Entidad Usuario  
|           └── Usuario.java  
  
|  
|   └── 🚑 paciente        # Gestión de pacientes  
|       |   └── controller  
|       |   └── service  
|       |   └── repository  
|       |   └── model  
|       └── dto  
|  
|
```

```

└── ⚖ historiaClinica          # Historia Clínica Electrónica (HCE)
    ├── controller
    ├── service
    ├── repository
    ├── model
    ├── dto
    └── 📜 builder      # Patrón Builder para estructura de historia
        └── HistoriaClinicaBuilder.java

└── 🗓 citas           # Agendamiento de citas médicas
    ├── controller
    ├── service
    ├── repository
    ├── model
    ├── dto
    └── 🎯 strategy      # Patrón Strategy para reglas de agendamiento
        └──

└── 💊 farmacia        # Gestión de medicamentos e inventario
    ├── controller
    ├── service
    ├── repository
    ├── model
    ├── dto
    └── 👀 observer       # Patrón Observer para alertas de stock

```

```
| └── 🌱 composite          # Fórmulas compuestas tipo receta
|
|   └──
|
|     ├── 🧪 laboratorio      # Gestión de órdenes y resultados de laboratorio
|     |
|     ├── controller
|     |
|     ├── service
|     |
|     ├── repository
|     |
|     ├── model
|     |
|     └── dto
|
|   └──
|
|     ├── 💳 facturacion       # Facturación y generación de RIPS
|     |
|     ├── controller
|     |
|     ├── service
|     |
|     ├── repository
|     |
|     ├── model
|     |
|     └── dto
|
|     └── 🏢 builder           # Constructor de estructuras complejas de facturas
|
|   └── 📈 reportes            # Indicadores y exportación de reportes
|     |
|     ├── controller
|     |
|     ├── service
|     |
|     ├── generator           # Generación de PDF, Excel, etc.
|     |
|     ├── model
|     |
|     └── dto
|
|   └──
```

```

├── 📹 telemedicina          # Videollamadas y sesiones virtuales
|   ├── controller
|   ├── service
|   ├── model
|   ├── ⚙ config             # Configuración WebRTC o Zoom/Meet integration
|   └── dto
└── 📡 notificaciones        # Sistema de notificaciones multicanal
    ├── service
    ├── model
    ├── 🏢 factory             # Patrón Factory para canales de notificación
    ├── 🌐 bridge               # Bridge para conectar lógica y medio (correo, SMS)
    └── 📁 common                # Recursos compartidos globalmente
        ├── 🚨 exception           # Manejo de excepciones personalizadas
        ├── 📄 constants            # Constantes de configuración y texto
        ├── ⚙ config                # Configuración general de la app
        ├── 🎭 annotations          # Anotaciones personalizadas
        └── 🖊 utils                 # Utilidades generales
    └── 🛡 application           # Punto de arranque de Spring Boot
        └── IntegrasaludApplication.java

```

9. LECCIONES APRENDIDAS

Dividir el sistema por módulos o dominios funcionales fue una gran decisión. Pensar el software por áreas como pacientes, citas, farmacia, laboratorio, etc., no solo ordena el trabajo, sino que permite que diferentes personas puedan trabajar en paralelo sin enredarse. Además, facilita mucho encontrar errores o hacer cambios sin afectar todo el sistema.

Aprender a usar los patrones de diseño en serio fue una experiencia muy valiosa. No se trató solo de cumplir un requisito académico, sino de ver cómo patrones como *Factory*, *Builder*, *Proxy* o *Composite* pueden resolver de forma elegante situaciones que en un sistema real se presentan todo el tiempo.

Entendimos que la interoperabilidad en salud no es tan simple como parece. Conectar el sistema con EPS, con estándares como HL7 o con servicios externos es un reto técnico y normativo. Pero también es una oportunidad para que el sistema esté preparado para escalar y ser competitivo en el mercado real.

Usar una base de datos única para todo tiene sus ventajas en versiones iniciales: es más fácil de mantener y entender. Pero a largo plazo, vimos que puede generar cuellos de botella o dependencia entre módulos. Por eso, es importante desde ya pensar en cómo dividirla lógicamente o escalarla de forma segura.

Por último, aprendimos que hacer buenos diseños antes de escribir código es una de las mejores decisiones que se pueden tomar. Diagramas de clases, de arquitectura, de componentes... todos ayudaron a entender mejor el sistema, anticipar errores y ahorrar tiempo en desarrollo.

CONCLUSIÓN

INTEGRASALUD nació con una idea clara: crear una solución que realmente entienda cómo funcionan las IPS en Colombia. No solo se trataba de digitalizar procesos, sino de hacerlo de forma ordenada, segura y útil para quienes trabajan día a día en la atención médica. Al final del desarrollo, podemos decir que sí es posible construir un sistema completo, modular y adaptable, capaz de apoyar tanto a un consultorio pequeño como a una red médica más grande.

Escoger herramientas como Spring Boot para el backend y React para el frontend fue una decisión acertada. Ambas tecnologías no solo son potentes y modernas, sino que permiten construir software con buena estructura, rendimiento y facilidad de mantenimiento. Además, se integran muy bien con prácticas como inyección de dependencias, control de acceso y arquitectura por capas.

Uno de los logros más importantes fue aplicar de forma concreta los patrones de diseño vistos en teoría. No quedaron solo en conceptos: se tradujeron en código real que resolvió problemas concretos, como cómo crear objetos de manera flexible, proteger accesos sensibles o generar estructuras complejas paso a paso.

Documentar todo —desde el análisis inicial hasta la implementación final— no fue una tarea menor, pero sí fundamental. Tener esa ruta clara nos ayudó a tomar mejores decisiones en cada etapa y a mantener la coherencia del sistema en todo momento.

11. BIBLIOGRAFIA

Diseño de Software y Patrones

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Freeman, E., Freeman, E., Bates, B., & Sierra, K. (2020). *Head First Design Patterns* (2nd ed.). O'Reilly Media.

Martin, R. C. (2017). *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall.

Tecnologías Utilizadas

Spring Team. (2023). *Spring Boot Documentation*. <https://docs.spring.io/spring-boot/>

React Documentation. (2024). *React – A JavaScript library for building user interfaces*. <https://react.dev>

PostgreSQL Global Development Group. (2024). *PostgreSQL 15 Documentation*.

<https://www.postgresql.org/docs/>

Normativas y Regulaciones en Salud - Colombia

Ministerio de Salud y Protección Social (Colombia). *Resolución 1995 de 1999*. Por la cual se establecen normas para el manejo de la historia clínica.

Congreso de Colombia. *Ley 1581 de 2012*. Protección de datos personales.

DIAN (Dirección de Impuestos y Aduanas Nacionales). *Lineamientos de facturación electrónica*. <https://www.dian.gov.co>

Estándares e Interoperabilidad en Salud

HL7 International. *HL7 FHIR – Fast Healthcare Interoperability Resources.*

<https://www.hl7.org/fhir/>

OpenHIE. (2023). *Open Health Information Exchange.* <https://ohie.org>

Herramientas de Desarrollo y Documentación

Docker Docs. (2024). *Docker for Developers.* <https://docs.docker.com/>

Complementaria (recomendado para profundizar)

Fowler, M. (2003). *Patterns of Enterprise Application Architecture.* Addison-Wesley.

Sommerville, I. (2020). *Software Engineering* (10th ed.). Pearson Education.