



**Implementación de estándares de almacenamiento y seguridad de Bioseñales en una  
historia clínica electrónica de código abierto.**

Daniel Pérez Viana

Trabajo de grado presentado para optar al título de Bioingeniero

Asesor

John Fredy Ochoa Gómez, Doctor (PhD) en Ingeniería Electrónica

Línea de investigación:

Neuroingeniería

Grupo de investigación:

Grupo de Neuropsicología y Conducta

Universidad de Antioquia

Facultad de Ingeniería

Bioingeniería

Medellín

2025

Cita	Pérez Viana [1]
Referencia Estilo IEEE (2020)	[1] D. Pérez Viana, "Implementación de estándares de almacenamiento y seguridad de Bioseñales en una historia clínica electrónica de código abierto", Trabajo de grado profesional, Bioingeniería, Universidad de Antioquia, Medellín, Antioquia, Colombia, 2025.



Centro de Documentación Ingeniería (CENDOI)

**Repositorio Institucional:** <http://bibliotecadigital.udea.edu.co>

Universidad de Antioquia - [www.udea.edu.co](http://www.udea.edu.co)

**Rector:** John Jairo Arboleda Céspedes.

**Decano/Director:** Julio César Saldarriaga Molina.

**Jefe departamento:** Jenny Kateryne Aristizábal Nieto.

El contenido de esta obra corresponde al derecho de expresión de los autores y no compromete el pensamiento institucional de la Universidad de Antioquia ni desata su responsabilidad frente a terceros. Los autores asumen la responsabilidad por los derechos de autor y conexos.

#### TABLA DE CONTENIDO

LISTA DE TABLAS.....	5
RESUMEN.....	9
ABSTRACT .....	10

I. INTRODUCCIÓN .....	11
II. OBJETIVOS .....	13
OBJETIVO GENERAL.....	13
OBJETIVOS ESPECÍFICOS.....	13
III. MARCO TEÓRICO.....	14
Historias Clínicas Electrónicas (HCE): Concepto y Relevancia .....	14
Software Libre en el Ámbito de la Salud.....	14
Adaptación de las HCE a Necesidades Clínicas Especializadas .....	15
Seguridad y Normatividad en el Almacenamiento de Información Clínica .....	16
IV. METODOLOGÍA .....	18
Implementación de OpenMRS.....	18
Implementación de GNU Health .....	19
Implementación de OscarEMR.....	20
V. RESULTADOS Y DISCUSIÓN.....	27
VI. CONCLUSIONES .....	47
REFERENCIAS .....	48
ANEXOS.....	50
ANEXO I. Ficha técnica de documentación del software. .....	50
ANEXO II. Matriz comparativa de los softwares de HCE. ....	51
ANEXO III. Componente de barra lateral con ReactJS. ....	51
ANEXO IV. Componente de encabezado con ReactJS.....	54
ANEXO V. Componente de menú inicial con ReactJS.....	54
ANEXO VI. Componente de listado de pacientes con ReactJS. ....	55
ANEXO VII. Componente de creación de pacientes con ReactJS.....	57
ANEXO VIII. Componente de motivo de consulta con ReactJS. ....	61
ANEXO IX. Componente de antecedentes generales con ReactJS.....	64

ANEXO X. Componente de antecedentes personales con ReactJS. ....	67
ANEXO XI. Componente de antecedentes sexuales con ReactJS. ....	69
ANEXO XII. Componente de examen físico con ReactJS.....	73
ANEXO XIII. Componente de examen neurológico con ReactJS. ....	76
ANEXO XIV. Componente de carga de archivos con ReactJS.....	79
ANEXO XV. Creación de tabla patient_analysis. ....	80
ANEXO XVI. Tabla de documentación cualitativa de la adaptación del software seleccionado.....	81
ANEXO XVII. Métodos de carga y nombrado de archivos. ....	81

## LISTA DE TABLAS

TABLA I COMPARATIVO ENTRE GRAFANA Y POWER BI.....	16
TABLA II RESULTADOS DE LA ENCUESTA DE PERCEPCIÓN.....	36

## LISTA DE FIGURAS

Fig.	1.	Icono de New Schema.....	20
Fig.	2.	Ventana emergente para crear un nuevo esquema.....	21
Fig.	3.	Tabla Portables creada dentro de MySQL.....	22
Fig.	4.	Botón de Inicio para crear un nuevo dashboard.....	23
Fig.	5.	Botón para seleccionar la fuente de datos.....	23
Fig.	6.	Vista previa del modelo de datos.....	24
Fig.	7.	Botones para crear nuevas medidas y columnas en el modelo de datos.....	25
Fig.	8.	Línea de código en DAX.....	25
Fig.	9.	Nueva tabla despivoteada.....	26
Fig.	10.	Página #1 del Dashboard.....	28
Fig.	11.	Página #2 del Dashboard.....	29
Fig.	12.	Sección de objetos visuales en Power BI.....	30
		Fig. 13. Marketplace de objetos visuales.....	31
Fig.	14.	Gráfico de Cajas y Bigotes.....	32
Fig.	15.	Editor de la visualización de cajas y bigotes.....	32

Fig.	16.	Sección	de	filtros.		
.....	.....	.....	.....	34		
Fig.	17.	Botón	para	publicar	el	dashboard.
.....	.....	.....	.....	.....	.....	34

## SIGLAS, ACRÓNIMOS Y ABREVIATURAS

**HCE** Historia Clínica Electrónica

**EEG** Electronecefalografía

**DB** Base de datos

## RESUMEN

El almacenamiento seguro y estructurado de datos clínicos sensibles, como las bioseñales derivadas de estudios electrofisiológicos (particularmente la electroencefalografía, EEG), representa un reto técnico y normativo en el desarrollo de sistemas de información en salud. En este contexto, el presente proyecto tiene como objetivo principal analizar la viabilidad del uso de sistemas de Historia Clínica Electrónica (HCE) de código abierto para gestionar información neurofisiológica, evaluando su capacidad de adaptación a requerimientos específicos y su conformidad con los estándares internacionales de seguridad y protección de datos personales.

A partir de una revisión detallada de plataformas de HCE de libre acceso, se establece un marco comparativo que considera criterios como interoperabilidad, escalabilidad, modularidad, arquitectura técnica y capacidad de extensión hacia dominios clínicos especializados. Particular énfasis se da al tratamiento de bioseñales, las cuales presentan demandas adicionales en términos de almacenamiento estructurado, trazabilidad y confidencialidad.

El enfoque del proyecto contempla no solo la evaluación conceptual de estas plataformas, sino también el análisis de su sostenibilidad tecnológica, su comunidad de soporte y su potencial para ser implementadas y adaptadas en entornos clínicos reales o simulados. Con base en esta revisión, se busca identificar qué soluciones de software libre ofrecen una arquitectura robusta, segura y extensible para incorporar información neurofisiológica, permitiendo su uso en contextos clínicos especializados como el diagnóstico y seguimiento de patologías neurológicas.

***Palabras clave — HCE, seguridad, almacenamiento, bioseñales, EEG, normatividad, software.***

## ABSTRACT

Secure and structured storage of sensitive clinical data—such as biosignals obtained from electrophysiological studies, particularly electroencephalography (EEG)—poses significant technical and regulatory challenges for health-information systems. This project's primary objective is to assess the feasibility of open-source Electronic Health Record (EHR) platforms for managing neurophysiological information, evaluating their adaptability to specialized requirements and their compliance with international standards on data security and privacy.

An in-depth review of freely available EHR platforms establishes a comparative framework based on criteria including interoperability, scalability, modularity, technical architecture, and extensibility to specialized clinical domains. Special emphasis is placed on biosignal handling, which demands additional guarantees of structured storage, traceability, and confidentiality.

The project encompasses not only a conceptual evaluation of these platforms but also an analysis of their technological sustainability, community support, and potential deployment in real or simulated clinical environments. Drawing on this review, the study seeks to identify which open-source solutions provide a robust, secure, and extensible architecture for integrating neurophysiological data, thereby supporting specialized clinical applications such as the diagnosis and monitoring of neurological disorders.

***Keywords — EHR, security, storage, biosignals, EEG, regulations, software.***

## I. INTRODUCCIÓN

El manejo seguro y eficiente de la información clínica es un pilar fundamental en los sistemas de salud contemporáneos. La creciente digitalización de los datos médicos ha impulsado el desarrollo e implementación de Historias Clínicas Electrónicas (HCE), las cuales permiten un acceso ágil, centralizado y estructurado a la información del paciente, mejorando los procesos clínicos, administrativos y de investigación [1]. No obstante, este avance también plantea desafíos importantes en cuanto a la protección de datos sensibles, interoperabilidad, escalabilidad y adaptabilidad de los sistemas utilizados.

En este contexto, los sistemas de HCE de código abierto han cobrado relevancia por ofrecer soluciones accesibles, auditables y modificables que pueden adaptarse a distintas necesidades institucionales, técnicas y regionales [2]. Frente a la necesidad de contar con una HCE orientada al análisis neurológico, este proyecto comenzó con la evaluación comparativa de tres plataformas libres: OpenMRS, GNU Health y OscarEMR. El criterio inicial fue identificar cuál de estas herramientas ofrecía mejores condiciones para la personalización de formularios clínicos, así como facilidad de implementación y mantenimiento.

Durante el proceso de prueba e instalación, se presentaron dificultades técnicas con OscarEMR, particularmente debido a problemas en la configuración de los repositorios disponibles, lo que impidió avanzar con su implementación. Por otro lado, si bien tanto OpenMRS como GNU Health fueron instalables y funcionales en sus versiones libres, ambos presentaron complejidades importantes al momento de modificar el código fuente, lo que limitó la posibilidad de personalizar directamente sus interfaces o estructuras internas.

Finalmente, se optó por desarrollar una solución propia basada en la base de datos de OpenMRS, dado que esta plataforma demostró ser la más robusta, ampliamente documentada y con mayor soporte de la comunidad. Además, su arquitectura modular y la posibilidad de implementación mediante contenedores Docker facilitaron su integración en el entorno de desarrollo. Sin embargo, debido a las restricciones de sus esquemas internos y la rigidez de algunas estructuras de datos, se decidió construir una nueva capa de formularios personalizados para el

registro detallado del historial clínico neurológico del paciente, incluyendo antecedentes personales y familiares, examen físico general y evaluación neurológica.

El sistema resultante consiste en una interfaz frontend con múltiples formularios diseñados específicamente para la recolección de datos clínicos orientados a estudios neurológicos. Esta capa interactúa con un backend adaptado para gestionar la comunicación con la base de datos original de OpenMRS, incorporando una nueva tabla estructurada para cumplir con los requerimientos específicos del proyecto, manteniendo al mismo tiempo la relación con las tablas base de pacientes y registros clínicos.

Este enfoque busca responder a una necesidad concreta del ámbito clínico-académico, al desarrollar una herramienta funcional que permita almacenar información sensible bajo criterios de seguridad y flexibilidad, empleando tecnologías de libre acceso que favorecen la transparencia, la reutilización y la soberanía tecnológica [3].

## II. OBJETIVOS

### OBJETIVO GENERAL

Implementar una plataforma de historia clínica electrónica (HCE) libre, basada en OpenMRS, optimizada para el manejo de bioseñales, cumpliendo con los estándares de seguridad y normativas de protección de datos en Colombia.

### OBJETIVOS ESPECÍFICOS

Analizar diferentes sistemas de HCE libres evaluando su vigencia tecnológica, capacidad de extensión para almacenar bioseñales y su nivel de seguridad en la protección de datos del paciente.

Implementar la HCE más adecuada para la gestión de bioseñales, con enfoque en neurociencia aplicada (epilepsia, trastornos del sueño o enfermedades neurodegenerativas), asegurando el cumplimiento de las normativas colombianas de protección de datos.

Validar y realizar pruebas de campo del sistema implementado, evaluando su funcionalidad, seguridad y efectividad en entornos simulados para el manejo de bioseñales.

### III. MARCO TEÓRICO

#### **Historias Clínicas Electrónicas (HCE): Concepto y Relevancia**

La Historia Clínica Electrónica (HCE) es un sistema computacional que permite el almacenamiento estructurado, longitudinal y seguro de la información médica de los pacientes. Incluye datos administrativos, antecedentes, diagnósticos, resultados de exámenes, tratamientos y seguimientos realizados por los profesionales de la salud [4]. Su implementación responde a la necesidad de mejorar la continuidad de la atención, reducir errores médicos, agilizar procesos asistenciales y facilitar la interoperabilidad entre instituciones de salud.

La Organización Mundial de la Salud ha destacado que las HCE representan un componente esencial dentro de las estrategias de salud digital, ya que contribuyen no solo a mejorar la eficiencia en la prestación del servicio, sino también a fortalecer la vigilancia epidemiológica y la toma de decisiones basadas en evidencia [5].

En la práctica clínica, una HCE efectiva debe permitir el registro detallado y ordenado de la evolución del paciente, con capacidad de integrar resultados de laboratorio, imágenes, y en algunos casos, bioseñales fisiológicos. Esto resulta particularmente útil en áreas como la neurología, donde el análisis histórico y detallado del paciente es crítico para el diagnóstico y seguimiento de enfermedades como la epilepsia, los trastornos neurodegenerativos o las alteraciones del sueño [6].

#### **Software Libre en el Ámbito de la Salud**

El software libre o de código abierto se caracteriza por permitir a los usuarios ejecutar, estudiar, modificar y redistribuir el programa, lo que ofrece ventajas clave en sectores como la salud, donde los requerimientos institucionales, normativos y de infraestructura varían ampliamente entre contextos. En contraste con las soluciones propietarias, los sistemas de salud basados en software libre permiten una mayor transparencia, menor dependencia tecnológica, y la posibilidad de adaptar el sistema a necesidades clínicas o investigativas particulares [7].

En el ámbito de las HCE, existen varias soluciones de código abierto ampliamente utilizadas, entre las cuales se destacan:

- OpenMRS: Sistema modular y extensible, diseñado especialmente para contextos de bajos recursos, pero con capacidad de adaptarse a entornos clínicos complejos. Está respaldado por una comunidad internacional activa y una arquitectura orientada a servicios, lo que facilita su integración y despliegue [8].
- OscarEMR: Software originado en Canadá, centrado en atención ambulatoria y diseñado para ser utilizado por médicos de familia. Aunque es funcional y maduro, su instalación y mantenimiento pueden requerir configuraciones complejas, lo que limita su adaptabilidad en contextos externos.
- GNU Health: Sistema que integra funcionalidades clínicas, administrativas y de salud pública. Está orientado a la atención primaria con un enfoque de medicina social, e incluye módulos para gestión de pacientes, historia clínica y salud comunitaria [9].

Estas plataformas representan distintas filosofías de diseño y prioridades clínicas. Sin embargo, todas comparten el principio de permitir una personalización profunda del sistema, algo fundamental cuando se requiere construir estructuras de datos especializadas, como las necesarias para análisis neurológicos detallados.

### **Adaptación de las HCE a Necesidades Clínicas Especializadas**

Si bien las HCE están concebidas para ser herramientas genéricas en la gestión clínica, su verdadera eficacia radica en su capacidad de adaptarse a las necesidades específicas de cada especialidad médica. En el caso de la neurología, la historia clínica debe permitir el registro detallado de aspectos como:

- Antecedentes neurológicos personales y familiares.
- Exámenes físicos enfocados en la evaluación neurológica.
- Resultados de estudios clínicos y paraclínicos como electroencefalogramas (EEG), resonancias magnéticas, y potenciales evocados.
- Descripción precisa de síntomas, evolución temporal y factores desencadenantes.

Estas características requieren estructuras de datos que no siempre están contempladas en las HCE generales, por lo cual la posibilidad de modificar, extender o personalizar los formularios

y tablas de datos es crucial. Los sistemas de código abierto permiten este tipo de adaptaciones, aunque en algunos casos esto puede implicar un nivel de complejidad alto en términos técnicos y de desarrollo.

Además, cuando se busca integrar este tipo de información en una HCE, es necesario considerar no solo el diseño del frontend (formularios y experiencia de usuario), sino también la estructura de la base de datos, el modelo relacional y los mecanismos de validación de datos. Esto implica una comprensión profunda de los esquemas internos del sistema y una arquitectura flexible que permita extender su funcionalidad sin comprometer la integridad de los datos clínicos existentes.

### **Seguridad y Normatividad en el Almacenamiento de Información Clínica**

El manejo de información en salud está regulado por estrictas normativas de protección de datos debido a la sensibilidad y confidencialidad inherente a este tipo de información. En el ámbito internacional, estándares como la ISO/IEC 27001 establecen directrices para la gestión segura de la información, incluyendo control de accesos, trazabilidad, cifrado y políticas de recuperación ante desastres [10].

En Colombia, la Ley 1581 de 2012 establece el marco de protección de datos personales, y se complementa con los lineamientos técnicos del Ministerio de Salud en temas como la Historia Clínica Electrónica Interoperable (HCEI). Estos lineamientos definen aspectos técnicos y semánticos que deben cumplirse para asegurar la integridad, disponibilidad y confidencialidad de los datos clínicos [11].

En proyectos que involucran el diseño o adaptación de una HCE, se vuelve indispensable garantizar:

- Confidencialidad: Solo personal autorizado debe acceder a la información.
- Integridad: Los datos deben mantenerse completos, precisos y no manipulables.
- Disponibilidad: La información debe estar accesible cuando sea requerida para la atención.
- Trazabilidad: Todo acceso, modificación o consulta debe quedar registrado.

El desarrollo de soluciones informáticas en salud debe contemplar estos principios desde su concepción, asegurando que la arquitectura técnica permita auditar procesos, restringir accesos y proteger los datos tanto en tránsito como en reposo. El software libre, al permitir acceso total al código, brinda una ventaja adicional al facilitar auditorías externas de seguridad y cumplimiento normativo.

## IV. METODOLOGÍA

Este proyecto se desarrolló bajo un enfoque cualitativo, el cual se fundamenta en la recopilación, análisis e interpretación de información a partir de fuentes documentales, recursos técnicos y experiencias prácticas de implementación. Se analizó la viabilidad del uso de software libre de historia clínica electrónica (HCE) para el almacenamiento y gestión de registros médicos y bioseñales, con énfasis en la plataforma OpenMRS, desde una perspectiva técnica, normativa y funcional.

### Implementación de OpenMRS

La implementación de OpenMRS se llevó a cabo utilizando el repositorio oficial de la distribución de referencia disponible en GitHub: <https://github.com/openmrs/openmrs-distro-referenceapplication>.

Se optó por la versión basada en Docker, que facilita el despliegue del sistema completo mediante contenedores. El procedimiento seguido fue el siguiente:

1. Clonación del repositorio:

```
"git clone https://github.com/openmrs/openmrs-distro-referenceapplication.git"
"cd openmrs-distro-referenceapplication"
```

2. Configuración de variables:

Antes de la ejecución, se revisó el archivo docker-compose.yml para verificar los servicios definidos: servidor OpenMRS, base de datos MySQL, y entorno de inicialización. No fue necesario modificar variables para pruebas básicas.

3. Construcción y despliegue de los contenedores:

Se ejecutó el comando: “*docker-compose up -d*”

Este comando sirvió para levantar los contenedores y ejecutar automáticamente los scripts de configuración inicial.

4. Verificación del entorno:

Se utilizó: “*docker-compose ps*” para confirmar que los servicios estuvieran corriendo correctamente.

5. Acceso a la aplicación:

La aplicación quedó disponible en: <http://localhost:8080/openmrs>

6. Credenciales predeterminadas:

Al finalizar la instalación, se completó la configuración vía interfaz web y se estableció un usuario administrador.

7. Detención del sistema:

Para detener todos los servicios: “*docker-compose down*”

Este proceso permitió levantar un entorno funcional y estable de OpenMRS, listo para pruebas y desarrollo de extensiones específicas, incluyendo la exploración de la base de datos y la personalización de formularios.

## Implementación de GNU Health

La implementación de GNU Health se realizó utilizando el repositorio oficial de contenedores disponible en GitLab: <https://gitlab.com/gnu-health/docker/gnu-health-docker-compose.git>

Los pasos ejecutados para desplegar el sistema fueron los siguientes:

1. Clonación del repositorio:

```
“git clone https://gitlab.com/gnu-health/docker/gnu-health-docker-compose.git”
“cd gnu-health-docker-compose”
```

2. Revisión del archivo docker-compose.yml:

Se verificaron los servicios definidos: servidor GNU Health, base de datos PostgreSQL y cliente gráfico.

3. Despliegue de contenedores:

Se ejecutó el siguiente comando: “*docker-compose up -d*”

4. Validación del estado del entorno:

Para verificar el estado de los servicios se utilizó: “*docker-compose ps*”

5. Acceso al sistema:

a. Servidor web en: <http://localhost:8000>

b. Cliente gráfico: se puede ejecutar de manera local o en contenedor, accediendo al servidor a través de la IP del host o localhost.

6. Credenciales predeterminadas:

a. Usuario: *admin*

b. Contraseña: *gnusolidario*

## 7. Detención

del

entorno:

Para detener los contenedores: “*docker-compose down*”

Este proceso permitió desplegar GNU Health en un entorno de pruebas, aunque el uso completo del sistema requiere configurar adecuadamente el cliente gráfico (Tryton) para acceder a la interfaz clínica.

## Implementación de OscarEMR

Para la implementación de OscarEMR se utilizó la información contenida en la página web oficial del software <https://oscargalaxy.org>, aquí se describen algunos requerimientos iniciales para poder instalar y utilizar la aplicación.

- **Sistema Operativo:** Ubuntu 22.04 LTS (64 bits), instalado de forma nativa o en una máquina virtual.
- **Conocimientos Básicos:** Familiaridad con la línea de comandos de Linux y privilegios de superusuario.
- **Hardware Recomendado:**
  - Procesador Intel Xeon o i7.
  - Mínimo 16 GB de RAM.
  - Almacenamiento suficiente para la base de datos y respaldos.

Para entornos de producción, se recomienda configurar Ubuntu con cifrado de disco completo para proteger los datos en caso de pérdida o robo del equipo.

Los ajustes adicionales de configuración del software incluían instalaciones de dependencias como una versión mínima de Java 8, librerías de BD de MariaDB, además de los ajustes y creaciones de las BD y tablas específicas, la descarga de los servicios de OscarEMR y consideraciones adicionales en temas de seguridad. Debido a la cantidad de requisitos y procesos de configuración e instalación, se decidió no continuar la implementación sino enfocarla en un análisis más superficial basado en la documentación existente.

## **Creación de análisis técnico, comparativo y documentación del proceso.**

El siguiente paso, posterior a la fase de implementación técnica de los sistemas HCE seleccionados, corresponde al proceso sistemático de análisis cualitativo, orientado a la construcción de los insumos documentales esperados (fichas técnicas, matrices comparativas y documentación de adaptación). Esta etapa se enfocará en la aplicación de criterios de evaluación previamente definidos, mediante técnicas de análisis documental y estudio comparativo.

A partir de la experiencia práctica de despliegue de los tres sistemas (OpenMRS, GNU Health y OSCAR EMR), se procedió a elaborar fichas de análisis técnico para cada uno. Este proceso contempló una lectura comprensiva de la documentación oficial de cada software, el examen del código fuente disponible en los repositorios implementados y la exploración de sus funcionalidades directamente en los entornos desplegados.

Las fichas técnicas (Anexo I) fueron construidas con el fin de documentar de manera detallada las características técnicas, normativas y funcionales de cada uno de los sistemas de historia clínica electrónica (HCE) libres seleccionados: OpenMRS, GNU Health y OSCAR EMR. Para su elaboración, se definieron campos orientados a registrar aspectos clave como la arquitectura del software, el cumplimiento de normativas de seguridad de datos (HIPAA, GDPR, Ley 1581 de 2012), el nivel de extensibilidad del sistema y su capacidad para integrar o almacenar bioseñales, en particular señales EEG. Estas fichas permitieron realizar un análisis individual y profundo de cada software desde una perspectiva técnica y normativa.

De forma complementaria, se desarrolló una plantilla de matriz comparativa (Anexo II) cuyo propósito fue sintetizar la información recopilada en las fichas y permitir una comparación transversal entre los tres sistemas revisados. Esta matriz fue estructurada en torno a criterios previamente definidos, tales como seguridad de los datos del paciente, compatibilidad con estándares internacionales, facilidad de integración de bioseñales, grado de modularidad y adecuación para aplicaciones en neurociencia clínica. La matriz facilitó la identificación de fortalezas y debilidades relativas de cada software, con miras a seleccionar la plataforma más apropiada para los fines del proyecto.

Una vez se analizó la aplicación del software con base a la información disponible de OpenMRS y con varios criterios considerados, se optó por tomar un acercamiento diferente, debido a la complejidad de la estructura y funcionalidad del software en cuestión, por lo que se determinó que era mucho más viable realizar una HCE mucho más enfocada en almacenar la información necesaria en formularios específicos, entre los cuales, se resaltan los datos relacionados con el motivo de consulta, los antecedentes tanto personales como no personales, patologías, almacenamiento de resultados de estudios, entre otros campos más [12].

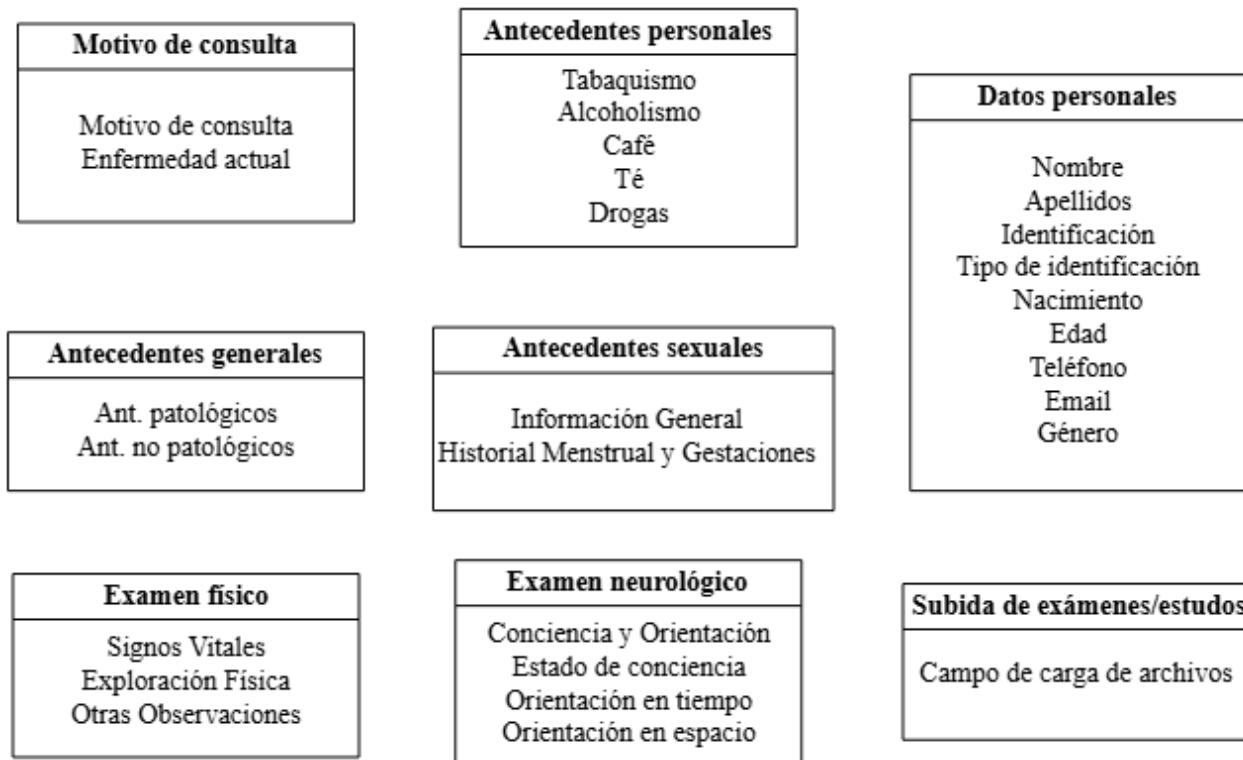


Figura 1. Mapa de los formularios necesarios con sus respectivos campos de información.

Ya sabiendo cuáles serían las categorías de los formularios, se inició el desarrollo de la interfaz gráfica que vería el profesional encargado de completarlos, para esto se utilizaron herramientas y frameworks FrontEnd en conjunto con JavaScript, como es el caso de ReactJS para la creación y manejo de componentes reutilizables, enrutamientos dentro de la página y la estilización visual con TailwindJS, que permite un diseño responsivo dependiendo de la resolución de pantalla desde la que se visite la plataforma.

Para no partir el desarrollo desde cero, se tomó como referencia un repositorio en Github que contenía una estructura de manejo de datos clínicos y pacientes, pero con algunas diferencias en cuanto a los frameworks utilizados, ya que este solamente sirvió como plantilla de ReactJS [13]. La adaptación inició con la modificación de los elementos estáticos como el caso de la barra lateral y la barra de encabezado (Anexos III y IV), para que incluyera botones de acceso rápido a las diferentes interfaces, como era el caso de la interfaz de inicio y los módulos de creación, edición y visualización de los pacientes.

El paso siguiente fue crear cada módulo, partiendo del menú inicial (Anexo V) desde donde se creó el redireccionamiento hacia la visualización de los pacientes (Anexo VI), la cual permite filtrar la información y buscar los registros en la BD para validar/editar los datos ingresados, y hacia la vista de creación de pacientes (Anexo VII), que es donde finalmente se ven los formularios planteados en la figura 1 de una forma dinámica, es decir, se cargan a medida que el usuario selecciona el componente deseado, lo cual permite una mayor agilidad en los tiempos de carga de la plataforma y de la información.

Finalmente, en el apartado visual se completaron los componentes correspondientes a los formularios de la figura 1, los cuales se pueden observar en los Anexos VIII – XIV y para el caso de la edición de pacientes, se utilizaron de forma recursiva los formularios de creación, con la condición de traer la información ya existente en la BD y la modificación sobre la misma.

Aprovechando las herramientas que brinda el repositorio de OpenMRS, se tomó la BD ya existente, con su arquitectura definida para la creación, modificación y manejo de los pacientes y se creó una tabla que permitiera el almacenamiento de la información suministrada en los formularios creados. Para esto, se utilizó la interfaz de MySQL Workbench con la cual se estableció una conexión directa para visualizar la información contenida en la BD implementada en los contenedores Docker y usando los puertos y credenciales por defecto que vienen configurados para esta, que son:

- Usuario/contraseña: *openmrs*
- Nombre de la BD: *openmrs*
- IP de conexión/puerto: *localhost:3306*

patient_analysis
patient_id (INT)
patient_data (TEXT)
consultation (TEXT)
general_records (TEXT)
personal_records (TEXT)
relational_records (TEXT)
physical_exam (TEXT)
neurological_exam (TEXT)
file_records (TEXT)
created_at (TIMESTAMP)

Figura 2. Estructura y campos de la BD.

Una vez establecida la conexión, se utilizó un “*Create\_table*” (Anexo XV) para generar la tabla necesaria, la cual debía cumplir con el requisito mínimo de tener una columna para cada formulario. Cada columna debía contener suficiente espacio para almacenar toda la información ingresada en los campos de la interfaz, por lo que se tomaron como *TEXTO*, además se tiene un campo identificador como un entero, que relaciona la tabla *patients* con la tabla creada llamada *patient\_analysis* y un registro de tiempo para cuando se crea una nueva entrada en esta tabla de la BD, como se muestra en la figura 2.

Para asegurar que la tabla mantuviera las relaciones, el desarrollo del backend permite la creación de un paciente, que mediante un botón en la interfaz genera un primer registro en la respectiva tabla *patients* y al mismo tiempo lo crea sobre la tabla *patient\_analysis* para garantizar que el identificador sea único en ambas y haga referencia a un único paciente, haciendo de cada ingreso un nuevo registro que pueda tener diferentes evaluaciones para un mismo paciente. Es importante resaltar que el funcionamiento del backend de la plataforma está basado en el framework ExpressJS, el cual permite la creación de un servicio de comunicación entre la BD y la plataforma mediante la implementación de una API con metodología CRUD, es decir, que tiene la capacidad de crear, leer, actualizar y borrar información.

Dado que la información contenida en los formularios es de carácter sensible y personal, se optó entonces por la aplicación de un formato de seguridad por medio de la encriptación de los datos, para esto se implementó protocolo conocido como *Advanced Encryption Standard* (AES) 256, ampliamente utilizado en el ámbito de la seguridad cibernética ya que permite volver indescifrable la información usando una clave única de 256 bits, dividiendo los datos en bloques, sustituyendo y mezclando los bytes de forma que sea un sistema impenetrable [14].

Asimismo, con el objetivo de documentar de forma sistemática las acciones llevadas a cabo durante el proceso de instalación, configuración y personalización de OpenMRS, se diseñó una plantilla para la documentación cualitativa (Anexo XVI) de la adaptación del sistema. Esta plantilla permitió registrar de manera organizada los pasos ejecutados durante la implementación, las decisiones técnicas adoptadas, los cambios realizados en la estructura de la base de datos y los formularios del sistema, así como la evaluación de su cumplimiento normativo y su idoneidad para el almacenamiento de bioseñales. Este documento también incluyó observaciones cualitativas sobre los retos encontrados, las posibilidades de escalabilidad y propuestas de mejora.

La construcción de estas plantillas respondió a la necesidad de garantizar una recolección de datos coherente con los objetivos de análisis comparativo y teórico del proyecto, asegurando al mismo tiempo trazabilidad, transparencia metodológica y rigurosidad en la evaluación de los sistemas seleccionados.

Para concluir el diseño de la plataforma, era fundamental la inclusión de un campo que facilitara el manejo de archivos, en este caso de los estudios realizados a los pacientes relacionados con las complicaciones neurológicas del mismo. Con el fin de hacer un primer acercamiento, se implementó un framework conocido como MulterJS con el cual se pueden almacenar dichos archivos de forma local en el servidor donde está corriendo el aplicativo de la plataforma web. Gracias a un método simple (Anexo XVII), el framework permite guardar los archivos con un identificador único como lo es una marca de tiempo en un formato que incluye hasta los milisegundos, por lo que es imposible que se guarden archivos con nombres duplicados, además de que la aplicación funciona de forma asíncrona, lo que quiere decir que no hace solicitudes en simultáneo, permitiendo que se cargue la información en orden de llegada. Una de las ventajas principales en este caso, está relacionada con el mantenimiento de los datos de forma local en

donde se ejecuta la plataforma, ya que comparado con el almacenamiento en la nube es mucho más rentable y de bajo costo.

## V. RESULTADOS Y DISCUSIÓN

La implementación de cada software permitió un análisis mucho más detallado de forma individual, lo cual fue esencial a la hora de determinar las capacidades propias relacionadas con la usabilidad de las aplicaciones. Para este caso, una vez ejecutados los repositorios se hizo un análisis exploratorio de las características básicas que ofrece cada uno en cuanto a la interfaz y el manejo de la información.

Los resultados obtenidos para OscarEMR no fueron los más amigables, ya que la documentación no incluía información suficiente para una implementación rápida y completa por medio de contenedores, sino que se esperaba la instalación paso a paso de los requerimientos y complementos necesarios para su ejecución en un entorno de Ubuntu. Por este motivo, se tomó un acercamiento un poco más investigativo que permitió recolectar los siguientes datos registrados en la plantilla de la siguiente tabla.

INFORMACIÓN GENERAL DEL SOFTWARE					
Nombre del Software:	Oscar EMR	Repository Oficial o Página Web:	<a href="https://oscargalaxy.org/downloads/">https://oscargalaxy.org/downloads/</a>	Lenguaje(s) de Programación:	Java
Versión Evaluada:	19.0	Licencia de Uso:	GPL v2	Requisitos del Sistema:	Java 8+, MySQL, Tomcat
CUMPLIMIENTO NORMATIVO Y SEGURIDAD DE DATOS					
Cumplimiento HIPAA:	Compatible con norma HIPAA	Cumplimiento Ley 1581 de 2012 (Colombia):	Requiere ajustes para ser compatible	Mecanismos de Autenticación y Control de Acceso:	Mediante usuario y contraseña, cada usuario tiene permisos definidos dentro de la plataforma
Cumplimiento GDPR:	Requiere ajustes para ser compatible	Otras Normativas Aplicables:	N/A	Cifrado de Datos en Tránsito y Reposo:	No se encontró información específica
INTEGRACIÓN CON BIOSEÑALES					

<b>Capacidad de Almacenar Datos EEG:</b>	No hay detalles que especifiquen la capacidad de almacenamiento, pero se prevé la necesidad de implementar módulos adicionales	<b>Módulos Existentes Relacionados con Bioseñales:</b>	La documentación existente no reporta módulos específicos para el manejo de bioseñales	
<b>Soporte para Formatos de Bioseñales:</b>	La documentación existente no reporta módulos específicos para el manejo de bioseñales	<b>Facilidad de Extensión (Plugins, Módulos):</b>	Es amigable con la personalización, pero requiere un conocimiento técnico más avanzado para poder implementar extensiones específicas	
<b>EVALUACIÓN FUNCIONAL</b>				
<b>Interfaz de Usuario:</b>	Amigable y enfocada en el manejo de pacientes, agendamientos y registros médicos	<b>Comunidad y Soporte Activo:</b>	A través de foros y canales de soporte	<b>Adecuación para Neurociencia Aplicada:</b>  Requiere personalizaciones significativas, módulos adicionales, o integración con otros sistemas
<b>Documentación Técnica Disponible:</b>	Hay documentación disponible en la página web pero limitada en cuanto a profundidad y detalle	<b>Escalabilidad y Flexibilidad del Sistema:</b>	Escalable para enfoques clínicos generales, no tan específicos como la neurociencia ya que requiere trabajo de adaptación adicional	

Tabla 1. Plantilla de ficha técnica aplicada a los resultados del software OscarEMR.

Una de las características más resaltables en general para OscarEMR es la capacidad de personalización mediante módulos adicionales, lo cual le permite ajustarse a las necesidades del usuario. Del mismo modo, esto representa una desventaja dado que el objetivo requiere la capacidad de almacenar documentos y archivos relacionados con los estudios del paciente, lo que hace difícil su implementación en este caso particular, además, al tener varios requisitos para su funcionamiento como la creación de una BD desde cero, implica que se debe tener suficiente capacidad de almacenamiento en hardware.

Este análisis ayudó a descartar este software como una opción a implementar, por lo que se continuó la investigación con la información recolectada sobre GNU Health. Una vez se ejecutó y se pudo ingresar con las credenciales mencionadas anteriormente, se puede apreciar una interfaz intuitiva, con una barra lateral que permite el direccionamiento a través de los módulos que vienen

con la aplicación por defecto, como lo son la gestión de personas, gestión administrativa, financiera y en salud, que es la más relevante para este estudio.

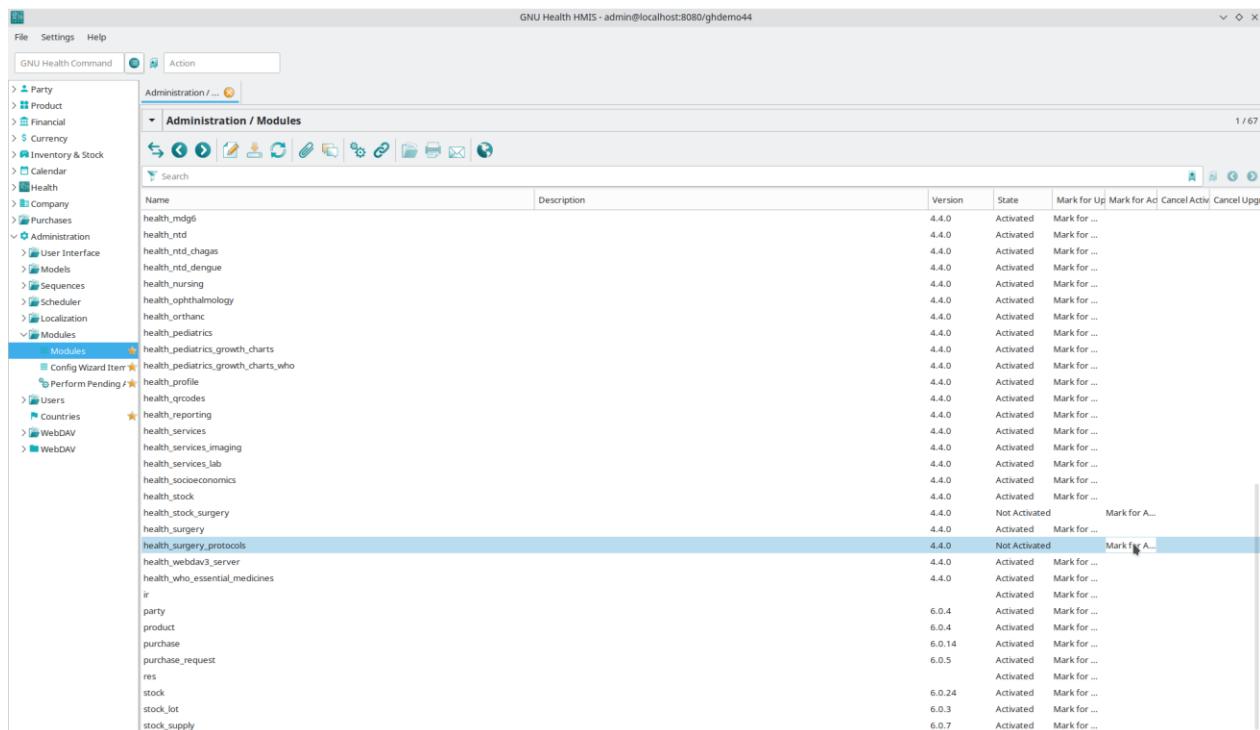


Figura 3. Captura de la interfaz de GNU Health en el apartado de Administración de módulos.

Como se observa en la figura 3, el software tiene la particularidad de ser bastante adaptable a las necesidades, incluso con varios módulos adicionales ya creados por la misma comunidad de desarrolladores. Esto califica como una gran ventaja del software, ya que se pueden añadir y quitar módulos a gusto, siempre y cuando se tengan los permisos de usuario adecuados para hacerlo. Algunos de los módulos también tenían la posibilidad de completar información detallada del paciente separada por categorías como el caso de dieta y ejercicio, sexualidad, obstetricia, pediatría, entre otras.

The screenshot shows the OB/GYN module of a software application. At the top, there are tabs for General Info, Surgeries, Functioning and Disability, Socioeconomics, Lifestyle, OB/GYN (which is selected), Genetics, and Medication. Below the tabs, there are two sections: 'General' and 'Screening'. Under 'General', there are fields for Fertile (checked), Pregnant (unchecked), Menarche age (12), Menopausal (unchecked), and Menopause age (empty). Under 'OB summary', there are fields for Pregnancies (1), Premature (0), Abortions (0), and Stillbirths (0). The 'Menstrual History' section includes fields for Date (09/26/2023), LMP (09/01/2023), Length (5), frequency (eumenorrhea), volume (normal), Regular (checked), Dysmenorrhea (unchecked), Reviewed (Cordara, Cameron), and Institution (GNU SOLIDARIO HOSPITAL). A toolbar at the bottom right contains various icons.

Figura 4. Captura del módulo de Obstetricia/Ginecología tomado de [15].

The screenshot shows the Lifestyle module of the software. At the top, there are tabs for General Info, Surgeries, Functioning and Disability, Socioeconomics, Lifestyle (selected), OB/GYN, Genetics, and Medication. Below the tabs, there are three sections: 'Diet and Exercise', 'Addictions', and 'Sexuality'. Under 'Diet and Exercise', there are fields for Exercise (checked), Minutes/day (40), Hours of sleep (7), and Sleeps at daytime (unchecked). Under 'Diet', there are fields for Vegetarian (unchecked), Belief (empty), Meals per day (3), Eats alone (unchecked), Coffee (checked), Cups per day (2), Soft drinks (sugar) (checked), Salt (checked), Currently on a diet (unchecked), and Diet info (empty). A large empty text area is at the bottom.

Figura 5. Captura del módulo de Estilo de vida tomado de [16].

En las figuras 4 y 5, se aprecian algunas características que se tuvieron en cuenta a la hora de la creación de los formularios y la particularidad que ya venían contemplados en algunos de los módulos preinstalados con el software, dándole un punto de ventaja sobre el anterior. Otra de las particularidades, se observó en el módulo de imagenología, que permite la carga de archivos en formato de imagen (png, jpg, etc), algo limitado para el tipo de formato que se suele usar en los archivos de resultados biomédicos, que suelen ser gdf, csv, bdf/edf, entre otros.

INFORMACIÓN GENERAL DEL SOFTWARE					
Nombre del Software:	GNU Health	Repositorio Oficial o Página Web:	<a href="https://www.gnuhealth.org">https://www.gnuhealth.org</a>	Lenguaje(s) de Programación:	Python
Versión Evaluada:	4.2	Licencia de Uso:	GPL v3	Requisitos del Sistema:	Linux, PostgreSQL, Tryton server
CUMPLIMIENTO NORMATIVO Y SEGURIDAD DE DATOS					
Cumplimiento HIPAA:	No se declara cumplimiento explícito, pero el	Cumplimiento Ley 1581 de 2012 (Colombia):	No se indica cumplimiento específico, pero es	Mecanismos de Autenticación y	Roles de usuario, control de acceso por módulos

	diseño modular y cifrado de base de datos permite alinearse con buenas prácticas de confidencialidad		adaptable mediante configuración local y auditorías	<b>Control de Acceso:</b>	
<b>Cumplimiento GDPR:</b>	Sí, GNU Health declara su alineación con el GDPR europeo, incluyendo derechos del paciente y consentimiento informado.	<b>Otras Normativas Aplicables:</b>	OMS/WHO ICD-10, LOINC, HL7 (a través de integraciones), SNOMED (limitado).	<b>Cifrado de Datos en Tránsito y Reposo:</b>	Configuración externa de TLS/SSL y soporte de cifrado de BD

#### INTEGRACIÓN CON BIOSEÑALES

<b>Capacidad de Almacenar Datos EEG:</b>	No nativa, pero se puede agregar mediante desarrollo de módulos personalizados.	<b>Módulos Existentes Relacionados con Bioseñales:</b>	No existen módulos específicos para EEG u otras señales electrofisiológicas al momento
<b>Soporte para Formatos de Bioseñales:</b>	No directo; requiere desarrollo de parsers o módulos externos que gestionen formatos EDF, BDF, etc.	<b>Facilidad de Extensión (Plugins, Módulos):</b>	Alta. GNU Health está diseñado para ser modular

#### EVALUACIÓN FUNCIONAL

<b>Interfaz de Usuario:</b>	Basada en cliente gráfico de escritorio (Tryton GTK), también tiene interfaz web mediante GNU Health Federation	<b>Comunidad y Soporte Activo:</b>	A través de foros y canales de soporte	<b>Adecuación para Neurociencia Aplicada:</b>	Requiere personalizaciones significativas, módulos adicionales, o integración con otros sistemas
<b>Documentación Técnica Disponible:</b>	Documentación en <a href="https://docs.gnuhealth.org/his/">https://docs.gnuhealth.org/his/</a> , manual de instalación, administración y desarrollo	<b>Escalabilidad y Flexibilidad del Sistema:</b>	Altamente escalable mediante Federation. Modularidad permite personalización por especialidad médica		

Tabla 2. Plantilla de ficha técnica aplicada a los resultados del software GNU Health.

GNU Health es un sistema robusto orientado principalmente a la salud pública y la gestión hospitalaria integral. Sus principales fortalezas se encuentran en la capacidad de estructurar datos clínicos de forma precisa, el manejo de indicadores de salud poblacional y su integración con módulos como laboratorio, farmacia y administración hospitalaria. Sin embargo, su arquitectura basada en Tryton, aunque modular, presenta ciertas limitaciones cuando se desea extender el sistema hacia el manejo de bioseñales, como EEG, ya que no existen módulos predefinidos para este propósito y su desarrollo requeriría trabajo adicional. La interfaz, más técnica y menos intuitiva, puede representar una barrera para usuarios sin experiencia técnica, y aunque permite configuraciones de seguridad, no cuenta con certificaciones explícitas de cumplimiento con normativas como HIPAA.

Los resultados iniciales para la ejecución del software contenido en el repositorio de OpenMRS, mostraron una interfaz amigable con un menú inicial interactivo para redirigirse a los módulos existentes en la aplicación base. De primera mano, se tienen diferentes módulos dirigidos al manejo de datos de pacientes, con la capacidad de registrar y crear, además de poder registrar signos vitales, generar reportes y administración de documentos.

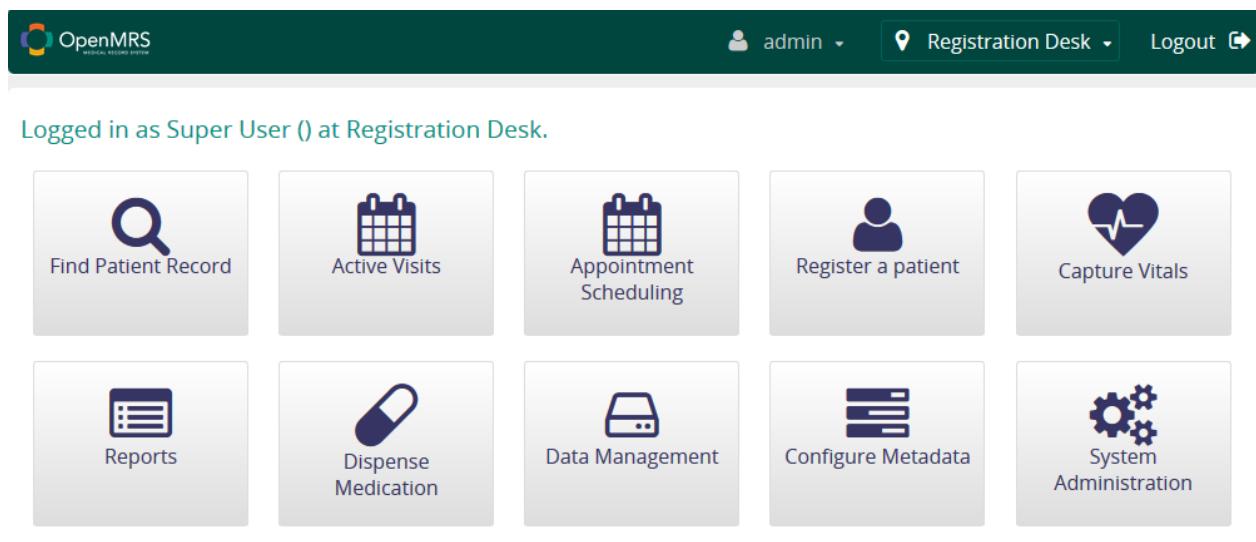


Figura 6. Captura del menú inicial de OpenMRS.

El repositorio utilizado, al ser una plantilla del software, solo incluye los módulos básicos mencionados antes, pero al mismo tiempo tiene la opción de utilizar algo llamado las “*herramientas de implementador*” similar al panel de administración de GNU Health. Este en su caso, es un poco más avanzado ya que permite ajustar, agregar o quitar elementos de la interfaz dependiendo del módulo seleccionado como se puede apreciar en la siguiente figura.

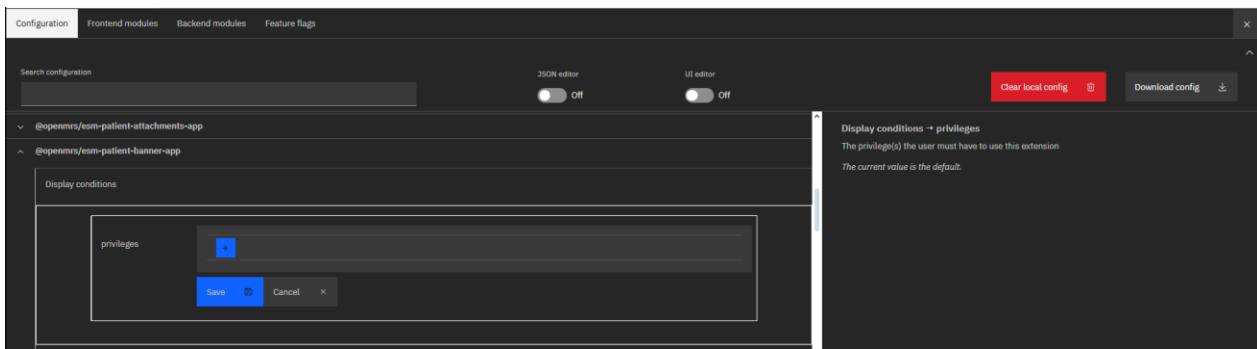


Figura 7. Captura del panel de herramientas del implementador de OpenMRS.



Figura 8. Captura del panel de accesos del encabezado de OpenMRS. De izquierda a derecha tiene las opciones de buscar, herramientas del implementador, registrar paciente, gestión de usuario y menú de la aplicación.

También está la opción de modificar los módulos desde código, pero uno de los principales retos fue entender cómo puede hacerse, ya que mucho del código fuente no se encuentra disponible entre los archivos que se descargan al clonar el repositorio, por lo que es necesario entender y revisar muy a fondo la documentación del software. Gracias a ella se pudo extraer mucha información relevante, la cual sirvió para completar la tabla de la ficha técnica del software OpenMRS.

#### INFORMACIÓN GENERAL DEL SOFTWARE

<b>Nombre del Software:</b>	OpenMRS	<b>Repositorio Oficial o Página Web:</b>	<a href="https://openmrs.org">https://openmrs.org</a> — Repositorio en GitHub: <a href="https://github.com/openmrs">https://github.com/openmrs</a>	<b>Lenguaje(s) de Programación:</b>	Java, HTML, JavaScript
<b>Versión Evaluada:</b>	OpenMRS Reference Application 2.13.0	<b>Licencia de Uso:</b>	Mozilla Public License 2.0	<b>Requisitos del Sistema:</b>	Java 8+, Tomcat 9+, MySQL/MariaDB

#### CUMPLIMIENTO NORMATIVO Y SEGURIDAD DE DATOS

<b>Cumplimiento HIPAA:</b>	No declarado de forma oficial, pero ofrece funcionalidades que pueden alinearse (control de acceso, logs de auditoría, cifrado).	<b>Cumplimiento Ley 1581 de 2012 (Colombia):</b>	No específicamente declarado, pero el sistema puede configurarse para cumplirla, dependiendo de la implementación.	<b>Mecanismos de Autenticación y Control de Acceso:</b>	Control por roles y privilegios Autenticación de usuarios Registro de eventos de acceso y cambios
<b>Cumplimiento GDPR:</b>	Algunas implementaciones personalizadas permiten alineación con GDPR; el núcleo del sistema puede ser adaptado a este estándar.	<b>Otras Normativas Aplicables:</b>	HL7 v2 y FHIR (mediante módulos de interoperabilidad)	<b>Cifrado de Datos en Tránsito y Reposo:</b>	Requiere configuración de HTTPS con Tomcat para cifrado en tránsito Cifrado en base de datos no habilitado por defecto, pero configurable externamente

#### INTEGRACIÓN CON BIOSEÑALES

<b>Capacidad de Almacenar Datos EEG:</b>	No de forma nativa, pero el sistema permite crear nuevos tipos de observaciones para almacenar datos estructurados	<b>Módulos Existentes Relacionados con Bioseñales:</b>	No existen módulos específicos para EEG u otras bioseñales, pero sí para imágenes médicas, laboratorio y signos vitales.
<b>Soporte para Formatos de Bioseñales:</b>	No directamente. Requiere extensión del modelo de datos o integración con sistemas externos	<b>Facilidad de Extensión (Plugins, Módulos):</b>	Muy alta. OpenMRS es altamente modular, con un sistema robusto de módulos y APIs para personalización.

EVALUACIÓN FUNCIONAL					
<b>Interfaz de Usuario:</b>	Interfaz web basada en navegador UX enfocada en registros clínicos Framework para personalizar formularios y dashboards	<b>Comunidad y Soporte Activo:</b>	Comunidad global activa Foros, Slack, y presencia en múltiples conferencias de salud digital	<b>Adecuación para Neurociencia Aplicada:</b>	No hay módulos específicos para EEG, sueño o enfermedades neurodegenerativas, pero el sistema permite su integración con desarrollo adicional
<b>Documentación Técnica Disponible:</b>	<a href="https://wiki.openmrs.org">https://wiki.openmrs.org</a> Guías para desarrolladores, administradores e implementadores	<b>Escalabilidad y Flexibilidad del Sistema:</b>	Alta escalabilidad con soporte para múltiples ubicaciones y sincronización Diseño modular permite adaptación a múltiples contextos clínicos		

Tabla 3. Plantilla de ficha técnica aplicada a los resultados del software OpenMRS.

OpenMRS, destaca por su enfoque modular y su capacidad para adaptarse a distintos contextos clínicos mediante la definición flexible de formularios y registros personalizados. Su interfaz web es más accesible y cuenta con una comunidad activa que desarrolla constantemente nuevas funcionalidades. En el contexto de la integración con bioseñales, ofrece una base más adecuada para almacenar y gestionar datos no estructurados, gracias a su diseño centrado en "observaciones" y conceptos configurables. Además, aunque no está certificado directamente en normativas de seguridad como HIPAA, sus mecanismos de autenticación, control de acceso y trazabilidad permiten adaptarse fácilmente a requisitos locales e internacionales. Por estas razones, OpenMRS se presenta como una solución más viable para proyectos orientados a neurociencia aplicada.

Sin embargo, a pesar de la exploración e implementación realizada con los 3 softwares trabajados, se identificaron limitaciones prácticas que impidieron su adopción directa en el contexto del presente proyecto. Pese a que cada uno ofrece funcionalidades valiosas, ninguno se ajustaba completamente a las necesidades específicas relacionadas con el almacenamiento y

visualización de bioseñales, particularmente señales EEG, dentro de un entorno centrado en la neurociencia aplicada.

Matriz Comparativa - Software de HCE				
Criterio	OpenMRS	GNU Health	OSCAR EMR	Observaciones
<b>Seguridad de datos del paciente</b>	Alta, cumple con HIPAA y permite configuración granular de acceso.	Alta, orientado a cumplimiento de GDPR y confidencialidad clínica.	Media, configuración inicial débil en entornos Docker no oficiales.	OpenMRS y GNU Health ofrecen mejores herramientas para cumplimiento normativo.
<b>Compatibilidad con bioseñales (EEG)</b>	No nativa, pero extensible mediante módulos personalizados.	Limitada, no incluye módulos preexistentes para EEG.	Nula, sin soporte directo ni vía extensiones documentadas.	Solo OpenMRS permitió avanzar hacia una posible integración a través del backend.
<b>Facilidad de instalación y uso</b>	Media, requiere conocimiento previo de Java, Tomcat y base de datos.	Media, instalación compleja con dependencias en Tryton y Postgres.	Baja, debido a la complejidad de los requisitos base.	OSCAR EMR fue difícil de ejecutar, ya que la configuración es desde cero.
<b>Flexibilidad y personalización</b>	Alta, altamente modular y orientado a entornos personalizables.	Media, requiere programación avanzada en Python para cambios profundos.	Baja, arquitectura cerrada en cuanto a modificaciones.	OpenMRS destaca por su diseño modular y orientado a personalización clínica.
<b>Cumplimiento de normativas locales</b>	Parcial, requiere ajustes para alinearse a la Ley 1581 de 2012.	Parcial, no incluye marco normativo colombiano por defecto.	Bajo, sin documentación que contemple normativas latinoamericanas.	Ninguno está listo para Colombia, pero OpenMRS se adapta mejor gracias a su estructura extensible.
<b>Soporte comunitario y actualizaciones</b>	Activo, con buena documentación y foros.	Activo, aunque más enfocado en salud pública global.	Limitado, comunidad técnica	OpenMRS cuenta con mayor soporte institucional y

			más pequeña y fragmentada.	continuidad de desarrollo.
<b>Documentación disponible</b>	Amplia y actualizada, tanto para usuarios como desarrolladores.	Técnica pero dispersa, centrada en Tryton y Sahana Eden.	Escasa y en su mayoría técnica; pocas guías para usuarios finales.	OpenMRS es el más accesible para aprender y adaptar con base en documentación oficial.
<b>Requisitos técnicos</b>	Moderados, requiere stack Java y servidor web (Tomcat, MySQL).	Altos, requiere stack Tryton + Postgres + GNU/Linux bien configurado.	Altos, ejecutable sin contenedores, requiere configurar desde cero Tomcat y MariaDB.	OSCAR EMR es el más exigente técnicamente. GNU Health puede ser más sencillo con Docker, pero sin soporte oficial pierde confiabilidad.

Tabla 4. Matriz comparativa de los 3 softwares implementados.

En el caso particular de OscarEMR, las barreras técnicas fueron evidentes tanto en la complejidad de la instalación como en la falta de documentación clara y módulos existentes para el manejo de datos electrofisiológicos. Además, su rigidez estructural y la ausencia de flexibilidad en la personalización de componentes representan obstáculos para una adaptación fluida y orientada a los objetivos.

Aunque OpenMRS demostró ser la solución más flexible y con mayor potencial de personalización, su implementación completa a través del sistema tradicional de módulos presentaba una curva de configuración elevada y un tiempo de integración superior al disponible para el desarrollo. Si bien su base de datos ofrecía una estructura sólida, la interfaz predeterminada no permitía una visualización intuitiva ni directa de bioseñales ni de formularios clínicos adaptados al dominio neurofisiológico sin intervenciones extensas en el código.

Por estas razones, se optó por una solución intermedia: aprovechar la BD de OpenMRS, que permite una gestión estructurada y segura de los datos clínicos, y construir una nueva interfaz web personalizada. Esta decisión permitió mantener la integridad y solidez de la información, mientras

se desarrollaba una plataforma visual enfocada en las necesidades específicas de almacenamiento de datos con foco neurológico, incluyendo el almacenamiento de estudios de bioseñales.

<b>Documentación Cualitativa - Adaptación del software seleccionado</b>	
<b>Aspecto analizado</b>	Adaptación de la BD y arquitectura de OpenMRS para el almacenamiento de la historia clínica y bioseñales.
<b>Descripción del proceso realizado</b>	Se utilizó la BD como pilar para el almacenamiento de los datos clínicos. Se diseñó una nueva tabla con el fin de almacenar toda la información en un mismo lugar y se implementaron medidas de seguridad mediante la encriptación de la información sensible.
<b>Ventajas observadas</b>	La documentación permite entender el funcionamiento de algunas de sus características como es el caso de la BD lo cual facilita la integración de forma externa sin necesidad de afectar las funcionalidades existentes.
<b>Limitaciones encontradas</b>	OpenMRS no contempla de forma nativa el manejo de archivos binarios ni la gestión de datos de alta frecuencia como los generados por dispositivos EEG. Además, su sistema de módulos requiere conocimientos específicos de desarrollo Java, lo que dificultó extender el sistema de forma directa.
<b>Configuraciones específicas realizadas</b>	Se creó una tabla nueva para almacenar la información de forma estructurada para cada formulario, mediante la encriptación de los datos se le añadió un nivel más de seguridad y se agregó la posibilidad de almacenar las rutas en el servidor donde se guardan los archivos relacionados a un paciente
<b>Propuestas de mejora</b>	A futuro se puede desarrollar un enfoque neurológico para el software, con la capacidad de gestionar información y estudios

de esta área, manteniendo la estructura modular y arquitectura base de OpenMRS.

Tabla 5. Documentación cualitativa de las adaptaciones aplicadas al software OpenMRS .

En la tabla 5, se registraron las adaptaciones que se tomaron en cuenta para la implementación del nuevo desarrollo. Iniciando con la adecuación del repositorio de medvault-frontend [13] y usando como referencia el encabezado y la barra lateral, se diseñó un menú inicial sencillo y con botones que redireccionan al usuario a los diferentes módulos como se puede apreciar en la siguiente figura.

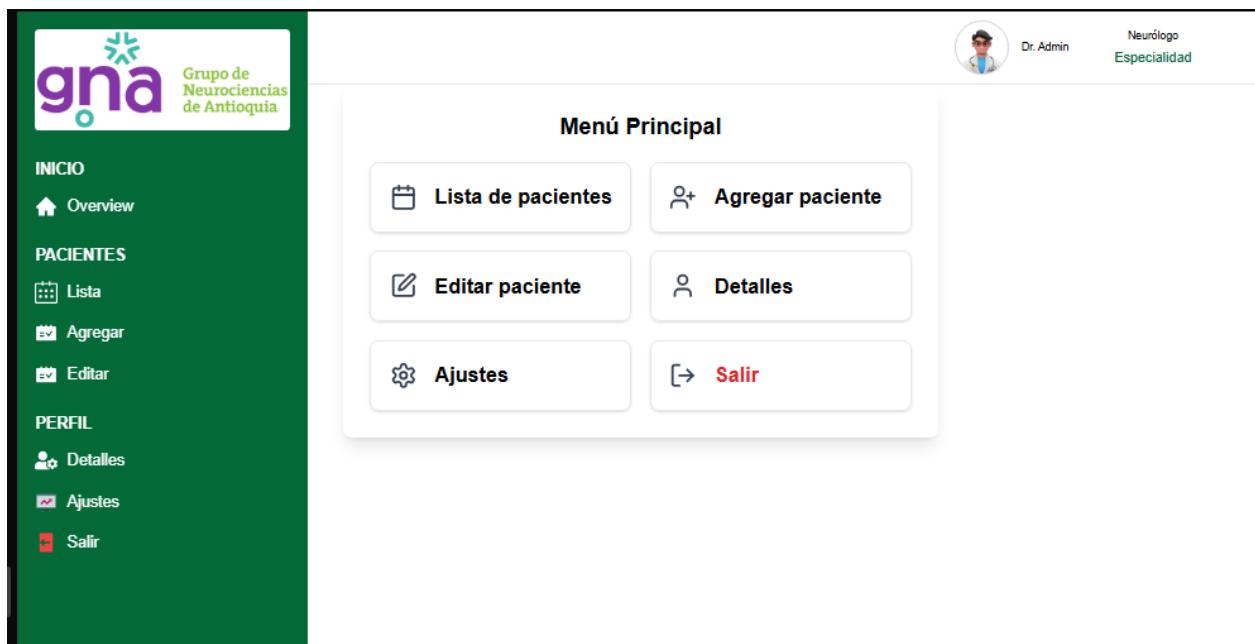


Figura 9. Captura del menú inicial de la plataforma web desarrollada.

En la figura 9, tanto los botones del menú principal como los de la barra lateral, llevan a las respectivas rutas que se encargan de diferentes funciones como mostrar el listado de pacientes registrados, agregar y editar pacientes. Los botones que llevan a los detalles y ajustes del usuario registrado no tienen aplicaciones desarrolladas, ya que el enfoque iba al almacenamiento de historias y pacientes, no del manejo de usuarios en el portal.

**Ficha Paciente**
[Crear paciente](#)

---

**Nombre:**

**Apellidos:**

**Identificación:**

**Tipo de identificación:**

**Nacimiento:**

 mm/dd/yyyy

**Edad:**

**Teléfono:**

**Email:**

**Género:**

Masculino
  Femenino
  Otro

---

[Consulta](#)
[Ant. generales](#)
[Ant. personales](#)
[Ant. sexuales](#)
[Examen físico](#)
[Examen neurológico](#)
[Subir archivos](#)

**Información general de la consulta**

Motivo de consulta:

Justificación

Enfermedad actual:

Historia actual

[Guardar](#)

Figura 10. Formulario de creación de paciente y de motivo de consulta.

Teniendo en cuenta la figura 1, donde se especificaron los formularios a desarrollar, se diseñó el formulario de registro de paciente, el cual se aprecia en la figura 10 en la parte superior al crear la ficha del paciente, la cual es el primer paso que debe hacer el usuario, completando los datos básicos de la persona que está siendo atendida. Una vez se completan, debe presionar el botón de crear paciente, que genera el identificador de este sobre la base de datos y va a ser el que dé la pauta de dónde va almacenada el resto de información de la consulta. Adicionalmente, en la parte inferior de la figura 10, se puede ver el primer formulario creado con el motivo de la consulta, aquí cabe resaltar que cada formulario se genera de forma ágil al recorrer las pestañas específicas, por

lo que es necesario que al completar la información el usuario guarde presionando el botón, llevando los datos a su respectivo campo en la BD.

The screenshot shows a web-based medical record system. At the top, there is a navigation bar with tabs: Consulta, Ant. generales, Ant. personales, Ant. sexuales, Examen físico (which is highlighted in blue), Examen neurológico, and Subir archivos. Below the navigation bar, the main content area is divided into sections:

- Signos Vitales**: This section contains six input fields for vital signs, each with an example value:
  - Presión Arterial: Ej: 120/80 mmHg
  - Frecuencia Cardíaca: Ej: 72 bpm
  - Frecuencia Respiratoria: Ej: 16 rpm
  - Temperatura: Ej: 36.5
  - Saturación de Oxígeno: Ej: 98%
  - Índice de Masa Corporal: Ej: 24.5
- Observaciones**: A large text area for general observations, with the placeholder text "Ej: Posición de toma de presión arterial y otras observaciones".
- Exploración Física**: This section contains six input fields for physical examination findings, each with a "Observaciones" placeholder:
  - Cabeza:
  - Cuello:
  - Tórax:
  - Abdomen:
  - Extremidades:
  - Piel:
- Otras Observaciones**: A large text area for additional observations, with the placeholder text "Escribe cualquier otra observación relevante...".

At the bottom center of the form is a blue "Guardar" (Save) button.

Figura 11. Formulario de examen físico con sus respectivos campos de información.

Continuando con los formularios, en la figura 11 se ven los campos específicos para el análisis físico que realiza el profesional sobre el paciente, son datos necesarios en cada atención ya que dan una visión inicial y general del estado actual del paciente. Finalmente, el último paso del

usuario debe ser cargar los archivos de bioseñales o estudios biomédicos realizados al paciente a través del formulario mostrado en la siguiente figura 12.

**Ficha Paciente**

Nombre:

Apellidos:

Identificación:

Tipo de identificación:

Nacimiento:  mm/dd/yyyy

Edad:

Teléfono:

Email:

Género:

Masculino  Femenino  Otro

Consulta    Ant. generales    Ant. personales    Ant. sexuales    Examen físico    Examen neurológico    Subir archivos

**Cargue de archivos**

Cargar archivos de estudios/bioseñales

Choose File No file chosen

Guardar

Figura 12. Formulario de carga de archivos.

Como es necesaria la visualización general de los registros, un apartado para listar los pacientes permite una gestión rápida para el usuario en dado caso que sea requerido modificar un registro. La siguiente figura muestra el diseño de la tabla que permite ver los pacientes que han sido atendidos e ingresados en el sistema.

Listado de pacientes								
Buscar por documento o nombre								Buscar
Tipo documento	Número documento	Nombre	Apellido	Género	Fecha nacimiento	Edad	Motivo de consulta	Acciones
CC	1234567899	David	Perez	M	2003-10-31	21	Dolor de cabezaaa	<button>Editar</button>
CC	1214744202	Daniel	Pérez	M	1998-09-15	26	Dolor en la parte frontal de la cabeza con punzones en la sien	<button>Editar</button>
TI	1233243241	Juan	Perez	M	2013-10-28	11	Sin datos	<button>Editar</button>
CC	1231231231	Jose	Gomez	M	1997-11-29		Dolor en la sien	<button>Editar</button>
CC	43444343434	Andres	Tovar	O	1997-11-10		Sin datos	<button>Editar</button>
CC	4141244124	Duvan	Zapata	M	2002-11-30	22	Sin datos	<button>Editar</button>
cc	121221212	Daniel	Perez Viana	M	2012-02-02	13	Sin datos	<button>Editar</button>

Figura 13. Vista del listado de pacientes registrados.

La figura 13 tiene una tabla generada con información básica y rápida de los pacientes, dándole una idea al usuario de qué registro debe atender según las necesidades de edición. La vista cuenta con un campo de búsqueda que permite filtrar los pacientes ya sea con el número de identificación o por sus nombres, ofreciendo velocidad para encontrar los registros específicos. El botón editar permite ver la información que ya fue cargada con anterioridad y como lo indica, hacer modificaciones sobre los datos.

La plataforma necesitó del desarrollo backend para poder comunicarse con la BD, entonces mediante una metodología CRUD es posible llevar la información que envía el usuario al interactuar con los botones, generando alertas de éxito o fallo durante la creación de los registros como lo muestra la siguiente figura.

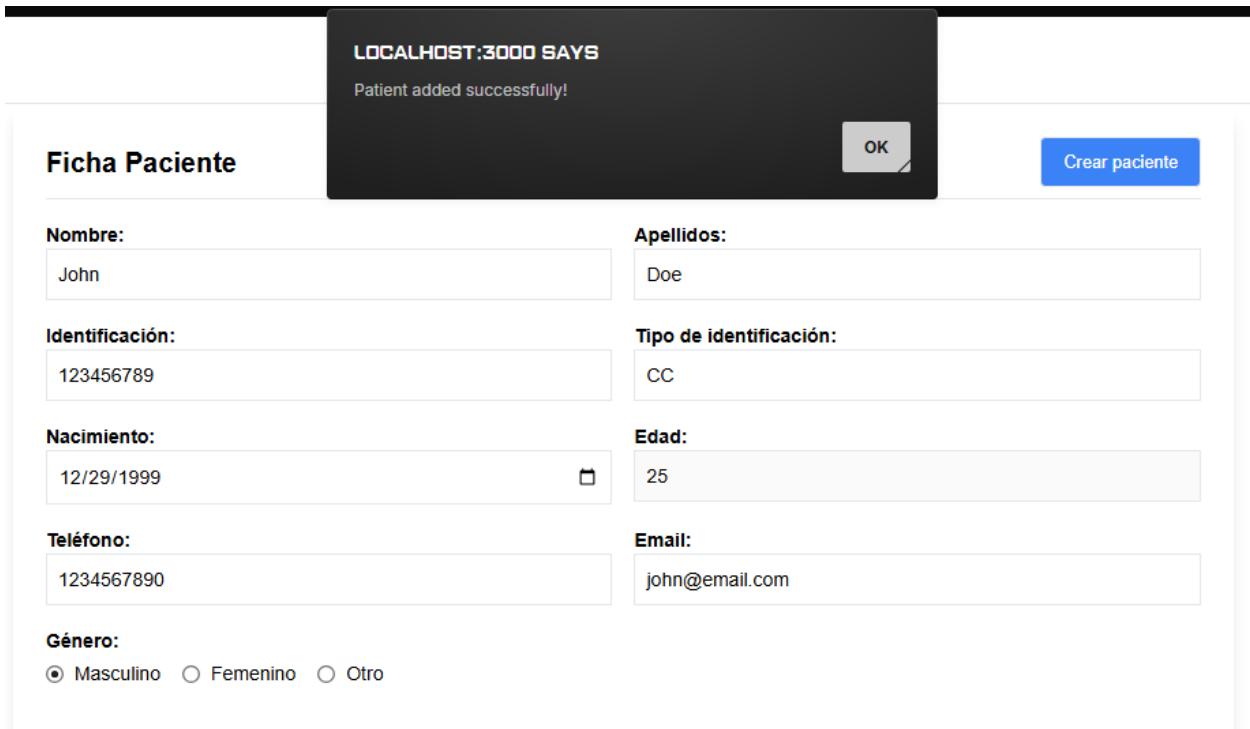


Figura 14. Alerta de creación de paciente exitosa.

Al mismo tiempo que se genera la notificación, se crea el registro en la BD, como se muestra a continuación.

patient_id	patient_data	consultation	general_records	personal_records	relational_records
75	{idnumber:'1234567899',idtype:'CC',name:'David',lastname:'Perez',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
76	{idnumber:'1234567899',idtype:'CC',name:'Daniel',lastname:'Perez',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
77	{idnumber:'1232342341',idtype:'TI',name:'Juan',lastname:'Perez',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
78	{idnumber:'1231231231',idtype:'CC',name:'Jose',lastname:'Gomez',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	JgLV8sp3xh9Hdvc/XyCjBTQ6URMdVp+ySh... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
79	{idnumber:'4344434343',idtype:'CC',name:'Andres',lastname:'Perez',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
80	{idnumber:'4141244124',idtype:'CC',name:'Duvan',lastname:'Zapata',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
81	{idnumber:'1212212121',idtype:'CC',name:'Daniel',lastname:'Perez',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
82	{idnumber:'1234567897',idtype:'CC',name:'John',lastname:'Doe',gender:'M',dateofbirth:'1985-01-01',placeofbirth:'Bogota',countryofbirth:'Colombia',maritalstatus:'Single',educationlevel:'Secondary',profession:'Software Engineer',employmentstatus:'Employed',incomelevel:'High',religion:'Christian',ethnicity:'White',nationality:'Colombian',bloodtype:'O+',height:'180cm',weight:'85kg',bloodgroup:'AB+',allergies:'None',medications:'None',vaccinations:'None'}	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL
NULL	NOTE	ITbRhR0x2D01/ta:8ZAYwIMKZTTwhsaFg/wvS... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak... I4UZGhnoFB879P/h:nhRHwJhazlEn/D0+flak...	NULL	pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T... pNtVY8CUpknttpwJx4yP0T...	NULL

Figura 15. Captura de la creación del paciente sobre la BD.

En la figura 15, se ve como el paciente *John Doe* fue creado, además se ven los demás campos relacionados directamente con los formularios creados. Se aprecia también la estructura de almacenamiento de la información, para esta aplicación los datos de cada formulario se guardan con formato de objeto tipo JSON, lo que permite un tratamiento más preciso de la información y al mismo tiempo, la posibilidad de convertirlo a texto para ser encriptado y almacenado como se observa en algunos campos donde hay una cadena de caracteres aleatorios. Esas cadenas son las correspondientes al resultado de encriptar la información con el método AES 256, donde la

estructura se divide en 3 partes separadas con “:”, la primera corresponde a un vector de inicialización (IV), la segunda es la información encriptada y la tercera, una etiqueta de autenticación.

The screenshot shows a web-based application for managing patient information. At the top, a modal window displays the message "LOCALHOST:3000 SAYS" and "Archivos guardados correctamente." with an "OK" button. To the right of the modal, a button labeled "Paciente creado" is visible. Below the modal, the main form is titled "Ficha Paciente". It contains the following fields:

- Nombre:** John
- Apellidos:** Doe
- Identificación:** 123456789
- Tipo de identificación:** CC
- Nacimiento:** 12/29/1999
- Edad:** 25
- Teléfono:** 1234567890
- Email:** john@email.com
- Género:**  Masculino  Femenino  Otro

Below the form, there is a navigation bar with links: Consulta, Ant. generales, Ant. personales, Ant. sexuales, Examen físico, Examen neurológico, and Subir archivos (which is underlined, indicating it is the active tab). A large button labeled "Subir archivos" is located at the bottom of the page.

**Cargue de archivos**

Cargar archivos de estudios/bioseñales

Choose File Untitled Diagram.drawio.png

Guardar

Figura 16. Almacenamiento de archivos y notificación de éxito al cargar.

Como se mencionó la importancia de poder almacenar archivos con resultados de análisis biomédicos de los pacientes, la figura 16 muestra cómo se notifica una vez el archivo queda guardado de forma exitosa. A la BD se envía la ruta en donde queda almacenado el archivo, ya que esta no puede almacenarlo como tal. El manejo de archivos es un tema de mayor complejidad, ya

que debe ser escalable y permitir grandes cantidades, es por eso que lo más recomendable para este tipo de aplicaciones sean soluciones en la nube, donde se puede llegar a tener mucho espacio por precios asequibles, sin embargo, en este caso se implementó una solución local y limitada, ya que depende de la capacidad de almacenamiento del servidor donde se va a mantener la plataforma.

El tratamiento de las rutas de los archivos sigue la misma parametrización que los demás formularios, en el sentido de la encriptación y visualización de los datos en la interfaz gráfica de MySQL, como se ve a continuación.

physical_exam	neurological_exam	file_records	created_at
NULL	NULL	NULL	2025-03-28 02:13:48
...	NULL	NULL	2025-03-31 00:31:44
NULL	NULL	NULL	2025-03-31 00:47:11
NULL	NULL	NULL	2025-03-31 01:23:34
NULL	NULL	NULL	2025-03-31 01:52:36
NULL	NULL	NULL	2025-03-31 02:12:53
NULL	NULL	pJ+UuYn/yPMZajbA:T+uVar3htvcX00QJH91LU...	2025-05-13 18:12:43
NULL	NULL	YkQziDUeJpvTSBFF:wx6kS2Z8qPtcjtM0psnX33e...	2025-05-30 06:53:34
NULL	NULL	NULL	NULL

Figura 17. Almacenamiento encriptado de la ruta de los archivos en la BD.

Al seguir la misma estructura de información, sería posible almacenar más de un archivo por paciente, pero en este caso se relaciona solo uno debido a que no se consideró como una posibilidad al inicio del desarrollo.

## VI. CONCLUSIONES

## REFERENCIAS

### INTRO

- [1] Organización Mundial de la Salud, Global strategy on digital health 2020–2025, 2021.
- [2] A. C. Payne et al., “Open source clinical software: the OpenMRS example,” AMIA Annual Symposium Proceedings, vol. 2010, pp. 552–556, 2010.
- [3] A. Mandl, I. Kohane, and K. Mandl, “Driving Innovation in Health Systems through an Ecosystem Approach to Open Data,” *Health Affairs*, vol. 41, no. 6, pp. 821–828, Jun. 2022.

### MARCO

- [4] J. H. van der Lei, “Use and abuse of computer-stored medical records,” *Methods of Information in Medicine*, vol. 30, no. 2, pp. 79–80, 1991.
- [5] Institute of Medicine, Key Capabilities of an Electronic Health Record System, National Academies Press, 2003.
- [6] M. Häyrinen, K. Saranto, and P. Nykänen, “Definition, structure, content, use and impacts of electronic health records: A review of the research literature,” *International Journal of Medical Informatics*, vol. 77, no. 5, pp. 291–304, 2008.
- [7] A. J. Barkovich, Pediatric Neuroimaging, 5th ed., Lippincott Williams & Wilkins, 2012.
- [8] E. G. Sittig and D. C. Classen, “Safe EHR implementation: A guide to preventing unintended consequences,” *Journal of the American Medical Informatics Association*, vol. 18, no. 2, pp. 155–161, 2011.
- [9] OpenMRS, “About OpenMRS,” [Online]. Available: <https://openmrs.org/about/>
- [10] ISO/IEC 27001, Information technology – Security techniques – Information security management systems – Requirements, 2013.
- [11] Ministerio de Salud y Protección Social de Colombia, “Guía para la implementación de la Historia Clínica Electrónica Interoperable (HCEI),” Bogotá, 2021.

### METODOLOGIA

- [12] Autor desconocido, Historia Clínica Neurología [Presentación en línea], SlideShare, nov. 2011. [En línea]. Disponible en: <https://www.slideshare.net/slideshow/historia-clinica-neurologia/12055052#1>
- [13] A. Musingarimi, medvault-frontend. GitHub repository, 2024. [Online]. Disponible en: <https://github.com/Amen-Musingarimi/medvault-frontend>

[14] B. Rees, "Use AES-256 Encryption to Secure Data," Progress Blogs, Apr. 5, 2022. [Online]. Disponible en: <https://www.progress.com/blogs/use-aes-256-encryption-secure-data>

## RESULTADOS

[15] GNU Health, "Gynecology — General tab," *GNU Health Documentation*, May 2025. [Online]. Disponible en: <https://docs.gnuhealth.org/his/userguide/modules/gynecology.html#general-tab>.

[16] GNU Health, "Lifestyle — GNU Health 4.2 documentation," *GNU Health*, [Online]. Disponible en: <https://docs.gnuhealth.org/his/userguide/modules/lifestyle.html>.

## ANEXOS

### ANEXO I. Ficha técnica de documentación del software.

<b>INFORMACIÓN GENERAL DEL SOFTWARE</b>				
Nombre del Software:		Repositorio Oficial o Página Web:		Lenguaje(s) de Programación:
Versión Evaluada:		Licencia de Uso:		Requisitos del Sistema:
<b>CUMPLIMIENTO NORMATIVO Y SEGURIDAD DE DATOS</b>				
Cumplimiento HIPAA:		Cumplimiento Ley 1581 de 2012 (Colombia):		Mecanismos de Autenticación y Control de Acceso:
Cumplimiento GDPR:		Otras Normativas Aplicables:		Cifrado de Datos en Tránsito y Reposo:
<b>INTEGRACIÓN CON BIOSEÑALES</b>				
Capacidad de Almacenar Datos EEG:			Módulos Existentes Relacionados con Bioseñales:	
Soporte para Formatos de Bioseñales:			Facilidad de Extensión (Plugins, Módulos):	
<b>EVALUACIÓN FUNCIONAL</b>				
Interfaz de Usuario:		Comunidad y Soporte Activo:		Adecuación para Neurociencia Aplicada:

Documentación Técnica		Escalabilidad y Flexibilidad del Sistema:
Disponible:		

ANEXO II. Matriz comparativa de los softwares de HCE.

Matriz Comparativa - Software de HCE				
Criterio	OpenMRS	GNU Health	OscarEMR	Observaciones
Seguridad de datos del paciente				
Compatibilidad con bioseñales (EEG)				
Facilidad de instalación y uso				
Flexibilidad y personalización				
Cumplimiento de normativas locales				
Soporte comunitario y actualizaciones				
Documentación disponible				
Requisitos técnicos				

ANEXO III. Componente de barra lateral con ReactJS.

```
import { useState } from "react";
import { Link } from "react-router-dom";
import { AiFillHome, AiFillSchedule } from "react-icons/ai";
import { SlCalender } from "react-icons/sl";
import { FaUserCog } from "react-icons/fa";
import { FcStatistics } from "react-icons/fc";
import { BiSolidLogOut } from "react-icons/bi";
import { Menu, X } from "lucide-react";
import logo from "../../assets/gna-logo.png";

export default function SideBar() {
```

```

const [isOpen, setIsOpen] = useState(false);

return (
  <>
    {/* Botón para abrir el Sidebar en móviles */}
    <button
      className="fixed top-4 left-4 z-50 p-2 bg-[#026937] text-white rounded
md:hidden"
      onClick={() => setIsOpen(!isOpen)}
    >
      {isOpen ? <X size={24} /> : <Menu size={24} />}
    </button>

    {/* Sidebar */}
    <nav
      className={`fixed top-0 left-0 h-full w-48 bg-[#026937] shadow-md transform
${isOpen ? "translate-x-0" : "-translate-x-full"} transition-transform
md:translate-x-0 md:w-56 lg:w-64 p-2 space-y-4 overflow-y-auto`}
    >
      {/* Logo */}
      <div className="flex items-center justify-center p-2">
        <Link to="/" className="">
          <img src={logo} alt="logo" className="w-30 h-auto bg-white rounded" />
        </Link>
      </div>

      <div className="space-y-4 text-sm">
        {/* Sección Inicio */}
        <div>
          <p className="text-white font-semibold px-2 mb-1">INICIO</p>
          <Link to="/" className="flex items-center space-x-2 p-2 rounded
hover:bg-gray-200 text-white hover:text-black">
            <AiFillHome size={20} />
            <p>Overview</p>
          </Link>
        </div>

        {/* Sección Pacientes */}
        <div>
          <p className="text-white font-semibold px-2 mb-1">PACIENTES</p>
          <Link to="/patients" className="flex items-center space-x-2 p-2
rounded hover:bg-gray-200 text-white hover:text-black">
            <SlCalender size={20} />
            <p>Lista</p>
          </Link>
        </div>
      </div>
    </nav>
  </>
)

```

```

        </Link>
        <Link to="/add-patient" className="flex items-center space-x-2 p-2 rounded hover:bg-gray-200 text-white hover:text-black">
            <AiFillSchedule size={20} />
            <p>Aregar</p>
        </Link>
        <Link to="/edit-patient" className="flex items-center space-x-2 p-2 rounded hover:bg-gray-200 text-white hover:text-black">
            <AiFillSchedule size={20} />
            <p>Editar</p>
        </Link>
    </div>

    {/* Sección Perfil */}
    <div>
        <p className="text-white font-semibold px-2 mb-1">PERFIL</p>
        <Link to="/profile/:id" className="flex items-center space-x-2 p-2 rounded hover:bg-gray-200 text-white hover:text-black">
            <FaUserCog size={20} />
            <p>Detalles</p>
        </Link>
        <Link to="/settings" className="flex items-center space-x-2 p-2 rounded hover:bg-gray-200 text-white hover:text-black">
            <FcStatistics size={20} />
            <p>Ajustes</p>
        </Link>
        <Link to="/logout" className="flex items-center space-x-2 p-2 rounded text-red-500 hover:bg-red-400">
            <BiSolidLogOut size={20} />
            <p className="text-white">Salir</p>
        </Link>
    </div>
</div>
</nav>

    {/* Fondo oscuro cuando el Sidebar está abierto */}
    {isOpen && (
        <div
            className="fixed inset-0 bg-black opacity-50 md:hidden"
            onClick={() => setIsOpen(false)}
        />
    )}
    </>
);

```

```
}
```

#### ANEXO IV. Componente de encabezado con ReactJS.

```
import React from 'react';
import doctorAvatar from '../../assets/doctor-avatar.webp';
import classes from './Header.module.css';

const Header = () => {
  const doctorName = 'Admin';
  const doctorSpeciality = 'Neurólogo';
  return (
    <div className={classes.header}>
      <div className={classes.doctor_details}>
        <div className={classes.name_image_cont}>
          <img
            src={doctorAvatar}
            alt="doctor avatar"
            className={classes.doctor_avatar}
          />
          <h2 className={classes.doctor_names}>Dr. {doctorName}</h2>
        </div>
        <div className={classes.name_icon_cont}>
          <h2 className={classes.doctor_names}>{doctorSpeciality}</h2>
          <span className={classes.heading}>Especialidad</span>
        </div>
      </div>
    </div>
  );
};

export default Header;
```

#### ANEXO V. Componente de menú inicial con ReactJS.

```
import { Link } from "react-router-dom";
import { Calendar, UserPlus, Edit, User, Settings, LogOut } from "lucide-react";
// import { useState } from "react";

export default function Overview({ active }) {
```

```

const menuItems = [
  { name: "Lista de pacientes", icon: Calendar, route: "/patients" },
  { name: "Aregar paciente", icon: UserPlus, route: "/add-patient" },
  { name: "Editar paciente", icon: Edit, route: "/patients" },
  { name: "Detalles", icon: User, route: "/profile/details" },
  { name: "Ajustes", icon: Settings, route: "/profile/settings" },
  { name: "Salir", icon: LogOut, route: "/logout", className: "text-red-600" },
];

return (
  <div className="p-6 bg-white shadow-lg rounded-lg max-w-4xl mx-auto">
    <div className="text-center">
      <h2 className="text-xl font-bold mb-4">Menú Principal</h2>
    </div>
    <div className="grid grid-cols-1 md:grid-cols-2 gap-4">
      {menuItems.map((item) => (
        <Link
          to={item.route}
          key={item.name}
          className={`${p-4 border rounded-lg shadow hover:bg-gray-100 flex items-center gap-4 ${item.className || ""}}`}>
          <item.icon size={24} className="text-gray-700" />
          <span className="text-lg font-semibold">{item.name}</span>
        </Link>
      )));
    </div>
  </div>
);
}

```

## ANEXO VI. Componente de listado de pacientes con ReactJS.

```

import React, { useState, useEffect } from "react";
import SinglePatient from './SinglePatient';
import classes from './PatientsList.module.css';

const PatientsList = () => {
  const [patients, setPatients] = useState([]);
  const [searchTerm, setSearchTerm] = useState("");

```

```

// Función para obtener los pacientes desde el backend
const fetchPatients = async () => {
  try {
    const response = await fetch(`http://localhost:5000/api/patients?search=${searchTerm}`);
    const data = await response.json();
    setPatients(data);
  } catch (error) {
    console.error("Error obteniendo los pacientes:", error);
  }
};

// Llamar a la API cuando se monta el componente o cambia el filtro
useEffect(() => {
  fetchPatients();
});

return (
  <div className="p-6 bg-white shadow-lg rounded-lg max-w-4xl mx-auto">
    <div className={classes.patients_list_container}>
      <div className="m-2">
        <input
          type="text"
          placeholder="Buscar por documento o nombre"
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          className={classes.search_input}
        />
        <button type="submit" className={classes.search_btn}>
          Buscar
        </button>
      </div>

      <table className={classes.table}>
        <thead className={classes.thead}>
          <tr className={classes.table_header}>
            <th className={classes.table_header_col}>Tipo documento</th>
            <th className={classes.table_header_col}>Número documento</th>
            <th className={classes.table_header_col}>Nombre</th>
            <th className={classes.table_header_col}>Apellido</th>
            <th className={classes.table_header_col}>Género</th>
            <th className={classes.table_header_col}>Fecha nacimiento</th>
            <th className={classes.table_header_col}>Edad</th>
            <th className={classes.table_header_col}>Motivo de consulta</th>
          </tr>
        </thead>
        <tbody>
          {patients.map((patient) => (
            <tr key={patient.id}>
              <td>{patient.tipo_documento}</td>
              <td>{patient.numero_documento}</td>
              <td>{patient.nombre}</td>
              <td>{patient.apellido}</td>
              <td>{patient.genero}</td>
              <td>{patient.fecha_nacimiento}</td>
              <td>{patient.edad}</td>
              <td>{patient.motivo_consulta}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  </div>
);

```

```

        <th className={classes.table_header_col}>Acciones</th>
    </tr>
</thead>
<tbody className={classes.patients_list}>
{
    patients.map((patient) => (
        <SinglePatient patient={patient} key={patient.patient_id}>
        />
    ))
}
</tbody>
</table>
</div>
</div>
);
};

export default PatientsList;

```

#### ANEXO VII. Componente de creación de pacientes con ReactJS.

```

import { useState } from "react";
import axios from "axios";
import DynamicForm from "./DynamicForm";

export default function AddPatientCard() {
    const [activeTab, setActiveTab] = useState("Consulta");
    const [formData, setFormData] = useState({
        idnumber: "",
        idtype: "",
        name: "",
        lastname: "",
        birthdate: "",
        age: "",
        phone: "",
        email: "",
        gender: ""
    });
    const [patientId, setPatientId] = useState(null);
    const [isSubmitting, setIsSubmitting] = useState(false);

    const handleChange = (e) => {
        setFormData({ ...formData, [e.target.name]: e.target.value });
    }
}

```

```

};

const handleBirthChange = (e) => {
    // setFormData({ ...formData, [e.target.name]: e.target.value });
    const birthdate = e.target.value;

    // Calculate age based on birthdate
    const birthDateObj = new Date(birthdate);
    const today = new Date();

    let age = today.getFullYear() - birthDateObj.getFullYear();
    const monthDiff = today.getMonth() - birthDateObj.getMonth();

    // Adjust age if the birthdate hasn't occurred yet this year
    if (monthDiff < 0 || (monthDiff === 0 && today.getDate() <
birthDateObj.getDate())) {
        age--;
    }

    // Update formData with both birthdate and calculated age
    setFormData({ ...formData, birthdate: birthdate, age: age });
    console.log(formData);

};

const handleGenderChange = (e) => {
    setFormData({ ...formData, gender: e.target.value });
};

const handleSubmit = async () => {
    if (isSubmitting) return;
    setIsSubmitting(true);

    try {
        const response = await axios.post("http://localhost:5000/api/add",
formData);
        const newPatientId = response.data.patient_id;

        setPatientId(newPatientId);
        console.log(formData);

        alert(`Patient added successfully!`);
    }
}

```

```

    } catch (error) {
      console.error("Error saving patient:", error);
      alert("Failed to save patient.");
    } finally {
      setIsSubmitting(false);
    }
  };

  return (
    <div className="p-6 bg-white shadow-lg rounded-lg max-w-4xl mx-auto">
      {/* Header */}
      <div className="flex justify-between items-center mb-4 border-b pb-2">
        <h2 className="text-xl font-bold">Ficha Paciente</h2>
        <button
          onClick={handleSubmit}
          className={`${border} px-4 py-2 rounded ${patientId ? "bg-gray-400 cursor-not-allowed" : "bg-blue-500 text-white"}`}
          disabled={patientId !== null}
        >
          {patientId ? "Paciente creado" : "Crear paciente"}
        </button>
      </div>

      {/* Patient Info */}
      <div className="grid grid-cols-2 gap-4 text-sm">
        <div>
          <label className="font-semibold">Nombre:</label>
          <input type="text" name="name" value={formData.name} onChange={handleChange} className="border p-2 w-full" disabled={patientId !== null} />
        </div>
        <div>
          <label className="font-semibold">Apellidos:</label>
          <input type="text" name="lastname" value={formData.lastname} onChange={handleChange} className="border p-2 w-full" disabled={patientId !== null} />
        </div>
        <div>
          <label className="font-semibold">Identificación:</label>
          <input type="text" name="idnumber" value={formData.idnumber} onChange={handleChange} className="border p-2 w-full" disabled={patientId !== null} />
        </div>
        <div>

```

```

        <label className="font-semibold">Tipo de identificación:</label>
        <input type="text" name="idtype" value={formData.idtype}
onChange={handleChange} className="border p-2 w-full" disabled={patientId !==
null} />
    </div>
    <div>
        <label className="font-semibold">Nacimiento:</label>
        <input type="date" name="birthdate" value={formData.birthdate}
onChange={handleBirthChange} className="border p-2 w-full" disabled={patientId !==
null} />
    </div>
    <div>
        <label className="font-semibold">Edad:</label>
        <input type="number" onChange={handleChange} name="age"
value={formData.age} className="border p-2 w-full" disabled={true} />
    </div>
    <div>
        <label className="font-semibold">Teléfono:</label>
        <input type="tel" name="phone" value={formData.phone}
onChange={handleChange} className="border p-2 w-full" disabled={patientId !==
null} />
    </div>
    <div>
        <label className="font-semibold">Email:</label>
        <input type="email" name="email" value={formData.email}
onChange={handleChange} className="border p-2 w-full" disabled={patientId !==
null} />
    </div>
    <div className="col-span-2">
        <label className="font-semibold">Género:</label>
        <div className="flex gap-4 mt-1">
            <label className="flex items-center">
                <input type="radio" name="gender" value="M" checked={formData.gender
=== "M"} onChange={handleGenderChange} className="mr-2" disabled={patientId !==
null} />
                    Masculino
            </label>
            <label className="flex items-center">
                <input type="radio" name="gender" value="F" checked={formData.gender
=== "F"} onChange={handleGenderChange} className="mr-2" disabled={patientId !==
null} />
                    Femenino
            </label>
        <label className="flex items-center">

```

```

        <input type="radio" name="gender" value="0" checked={formData.gender
== "0"} onChange={handleGenderChange} className="mr-2" disabled={patientId != null} />
        Otro
    </label>
</div>
</div>
</div>

/* Tabs */
<div className="mt-6 border-b flex gap-2 overflow-x-auto">
    [{"Consulta", "Ant. generales", "Ant. personales", "Ant. sexuales", "Examen físico", "Examen neurológico", "Subir archivos"].map((tab) => (
        <button
            key={tab}
            className={`${px-3 py-2 text-sm font-medium border-b-2 ${activeTab === tab ? "border-blue-500 text-blue-500" : "border-transparent"}`}
            onClick={() => setActiveTab(tab)}
        >
            {tab}
        </button>
    )));
</div>

/* Tab Content */
<div className="p-4 bg-gray-50 mt-4 rounded-lg text-gray-600">
    <DynamicForm activeTab={activeTab} patientId={patientId} />
</div>
</div>
);
}

```

#### ANEXO VIII. Componente de motivo de consulta con ReactJS.

```

import { useState, useEffect } from "react";
import axios from "axios";

export default function ConsultationForm({ patientId }) {
  const [formData, setFormData] = useState({
    motivoConsulta: "",
    enfermedadActual: "",
  });
}

```

```

//console.log(formData);

useEffect(() => {
  const fetchConsultation = async () => {
    try {
      const response = await axios.get(`http://localhost:5000/api/patient-analysis/${patientId}`);
      if (response.data.consultation) {
        setFormData(JSON.parse(response.data.consultation));
      }
    } catch (error) {
      console.error("Error fetching consultation:", error);
    }
  };
  if (patientId) {
    fetchConsultation();
  }
}, [patientId]);

const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
};

const sendConsultationData = async (e) => {
  e.preventDefault(); // Evita el recargo de la página

  const consultationData = JSON.stringify(formData);

  try {
    await axios.post("http://localhost:5000/api/patient-analysis/saveConsultation", {
      patient_id: patientId,
      consultation: consultationData,
    });
    alert("Consulta guardada correctamente.");
  } catch (error) {
    console.error("Error saving consultation:", error);
    alert("No se pudo guardar la consulta.");
  }
};

return (

```

```

<div className="p-4 border rounded-lg shadow-md bg-white">
  <form className="space-y-4">
    <h2 className="text-lg font-bold mb-4 text-gray-700">Información general de la consulta</h2>
    <div>
      <label className="font-semibold">Motivo de consulta:</label>
      <input
        name="motivoConsulta"
        value={formData.motivoConsulta}
        onChange={handleChange}
        placeholder="Justificación"
        className="border mb-2 p-2 w-full rounded"
      />
    </div>
    <div>
      <label className="font-semibold mt-2 py-2">Enfermedad actual:</label>
      <textarea
        name="enfermedadActual"
        value={formData.enfermedadActual}
        onChange={handleChange}
        placeholder="Historia actual"
        className="border mb-2 p-2 w-full rounded"
      />
    </div>

    <div className="text-center">
      <button
        type="button"
        onClick={sendConsultationData} // ☑ Aquí se corrige
        className="mt-4 px-6 py-3 bg-blue-500 text-white rounded-lg shadow-md hover:bg-blue-600 transition"
      >
        Guardar
      </button>
    </div>
  </form>
</div>
);
}

```

## ANEXO IX. Componente de antecedentes generales con ReactJS.

```
import { useState, useEffect } from "react";
import axios from "axios";

export default function PersonalHistoryForm({ patientId }) {
  const [formData, setFormData] = useState({
    patologicos: "",
    abuelos: "",
    padre: "",
    madre: "",
    hermanos: "",
    hijos: "",
    reaccionMedicamentos: "",
    alergias: "",
    transfusiones: "",
    traumatismos: "",
    quirurgicos: "",
    infecciosos: "",
    reproductivos: "",
    medicacion: "",
    vacunaciones: "",
    generoVida: "",
    alimentacion: "",
    vivienda: "",
    mascotas: "",
    tipoSuelo: "",
    tipoTecho: "",
    agua: "",
    serviciosSanitarios: "",
    banosDuchas: "",
    accesoInternet: ""
  });

  useEffect(() => {
    const fetchConsultation = async () => {
      try {
        const response = await axios.get(`http://localhost:5000/api/patient-analysis/${patientId}`);
        if (response.data.general_records) {
          console.log(response.data.general_records);

          setFormData(JSON.parse(response.data.general_records));
        }
      } catch (error) {
        console.error(error);
      }
    };
    fetchConsultation();
  }, []);
}
```

```

        console.error("Error fetching consultation:", error);
    }
};

if (patientId) {
    fetchConsultation();
}

}, [patientId]);

const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
};

const sendPersonalHistoryData = async () => {
    const generalRecordsData = JSON.stringify(formData);

    try {
        await axios.post("http://localhost:5000/api/patient-analysis/saveGeneralRecords", {
            patient_id: patientId,
            general_records: generalRecordsData,
        });
    }

    alert("Antecedentes guardados correctamente.");
} catch (error) {
    console.error("Error saving general records:", error);
    alert("No se pudo guardar los antecedentes.");
}
};

return (
    <div className="space-y-6">
        <div className="grid grid-cols-1 lg:grid-cols-2 gap-6">
            {/* Antecedentes Personales Patológicos */}
            <div className="p-4 border rounded-lg shadow-md bg-white">
                <h2 className="text-lg font-bold mb-4 text-gray-700">Antecedentes patológicos</h2>
                <div className="grid grid-cols-1 sm:grid-cols-2 gap-4">
                    {[{"label": "Patológicos", "name": "patologicos"}, {"label": "Abuelos/as", "name": "abuelos"}, {"label": "Padre", "name": "padre"}, {"label": "Madre", "name": "madre"}, {"label": "Hermanos/as", "name": "hermanos"}, {"label": "Hijos/as", "name": "hijos"}]
                </div>
            </div>
        </div>
    </div>
);

```

```

        { label: "Reacción a medicamentos", name: "reaccionMedicamentos" },
        { label: "Alergias", name: "alergias" },
        { label: "Transfusiones sanguíneas", name: "transfusiones" },
        { label: "Traumatismos", name: "traumatismos" },
        { label: "Quirúrgicos", name: "quirurgicos" },
        { label: "Infecciosos", name: "infecciosos" },
        { label: "Reproductivos", name: "reproductivos" },
        { label: "Medicación", name: "medicacion" },
    ].map((field, index) => (
        <div key={index}>
            <label style={{ color: "#667788", fontWeight: "bold", margin: "0" }}>
                {field.label}
            </label>
            <input style={{ border: "1px solid #ccc", width: "100%", padding: "5px" }} type="text" value={formData[field.name]} placeholder="Antecedentes" name={field.name} onChange={handleChange} />
        </div>
    )));
</div>
</div>

/* Antecedentes Personales No Patológicos */
<div className="p-4 border rounded-lg shadow-md bg-white">
    <h2 className="text-lg font-bold mb-4 text-gray-700">Antecedentes no patológicos</h2>
    <div className="grid grid-cols-1 sm:grid-cols-2 gap-4">
        [
            {
                { label: "Vacunaciones recibidas", name: "vacunaciones" },
                { label: "Género de vida", name: "generoVida" },
                { label: "Alimentación", name: "alimentacion" },
                { label: "Vivienda", name: "vivienda" },
                { label: "Mascotas", name: "mascotas" },
                { label: "Tipo de suelo", name: "tipoSuelo" },
                { label: "Tipo de techo", name: "tipoTecho" },
                { label: "Agua", name: "agua" },
                { label: "Servicios sanitarios", name: "serviciosSanitarios" },
                { label: "Baños/duchas", name: "banosDuchas" },
                { label: "Acceso a internet", name: "accesoInternet" },
            ].map((field, index) => (
                <div key={index}>

```

```

        <label          className="font-semibold"          style={{ color: "#text-gray-700" }}>{field.label}</label>
        <input
            name={field.name}
            value={formData[field.name]}
            onChange={handleChange}
            placeholder="Estado"
            className="border p-2 w-full rounded"
        />
    </div>
    ))}
</div>
</div>
</div>

/* Botón de Guardar */
<div className="text-center">
    <button
        type="button"
        onClick={sendPersonalHistoryData} // ☑ Corrección aquí
        className="mt-4 px-6 py-3 bg-blue-500 text-white rounded-lg shadow-md
        hover:bg-blue-600 transition"
    >
        Guardar
    </button>
</div>
</div>
);
}

```

#### ANEXO X. Componente de antecedentes personales con ReactJS.

```

import { useState, useEffect } from "react";
import axios from "axios";

export default function HabitsForm({ patientId }) {
    const [formData, setFormData] = useState({
        tabaquismo: "",
        alcoholismo: "",
        cafe: "",
        te: "",
        drogas: ""
    });
}

```

```

    });

//console.log(formData);

useEffect(() => {
  const fetchConsultation = async () => {
    try {
      const response = await axios.get(`http://localhost:5000/api/patient-analysis/${patientId}`);
      if (response.data.personal_records) {
        setFormData(JSON.parse(response.data.personal_records));
      }
    } catch (error) {
      console.error("Error fetching consultation:", error);
    }
  };
  if (patientId) {
    fetchConsultation();
  }
}, [patientId]);

const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
};

const sendHabitsData = async () => {
  const habitsData = JSON.stringify(formData);

  try {
    await axios.post("http://localhost:5000/api/patient-analysis/savePersonalRecords", {
      patient_id: patientId,
      personal_records: habitsData,
    });
    alert("Hábitos guardados correctamente.");
  } catch (error) {
    console.error("Error saving habits:", error);
    alert("No se pudo guardar la información.");
  }
};

return (
  <div className="p-4 border rounded-lg shadow-md bg-white">

```

```

        <h2    className="text-lg    font-bold    mb-4    text-gray-700">Antecedentes
personales</h2>
        <div className="space-y-4">
            <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
                {[

                    { label: "Tabaquismo", name: "tabaquismo" },
                    { label: "Alcoholismo", name: "alcoholismo" },
                    { label: "Café", name: "cafe" },
                    { label: "Té", name: "te" },
                    { label: "Drogas", name: "drogas" },
                ].map((field, index) => (
                    <div key={index}>
                        <label className="font-semibold">{field.label}:</label>
                        <input
                            name={field.name}
                            value={formData[field.name]}
                            onChange={handleChange}
                            placeholder="Estado"
                            className="border p-2 w-full rounded"
                        />
                    </div>
                )));
            </div>
        </div>

        <div className="text-center">
            <button onClick={sendHabitsData} className="mt-4 px-6 py-3 bg-blue-500
text-white rounded-lg shadow-md hover:bg-blue-600 transition">
                Guardar
            </button>
        </div>
    </div>
);
}

```

#### ANEXO XI. Componente de antecedentes sexuales con ReactJS.

```

import { useState, useEffect } from "react";
import axios from "axios";

export default function GinecoForm({ patientId }) {
    const [formData, setFormData] = useState({

```

```

    primerasRelacionesSexuales: "",
    vidaSexualActiva: "",
    numeroConyuges: "",
    preferenciasSexuales: "",
    relacionesExtramaritales: "",
    menarquia: "",
    cicloMenstrual: "",
    menopausia: "",
    fechaUltimaMenstruacion: "",
    numeroGestaciones: "",
  });

// console.log(formData);

useEffect(() => {
  const fetchConsultation = async () => {
    try {
      const response = await axios.get(`http://localhost:5000/api/patient-analysis/${patientId}`);
      if (response.data.relational_records) {
        setFormData(JSON.parse(response.data.relational_records));
      }
    } catch (error) {
      console.error("Error fetching consultation:", error);
    }
  };
  if (patientId) {
    fetchConsultation();
  }
}, [patientId]);

const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
};

const sendRelationalData = async () => {
  const relationalData = JSON.stringify(formData);

  try {
    await axios.post("http://localhost:5000/api/patient-analysis/saveRelationalRecords", {
      patient_id: patientId,
      relational_records: relationalData,
    });
  
```

```

        alert("Información guardada correctamente.");
    } catch (error) {
        console.error("Error al guardar:", error);
        alert("No se pudo guardar la información.");
    }
};

return (
    <div className="p-6 border rounded-lg shadow-md bg-white">
        <div className="space-y-6">
            {/* Información General */}
            <div>
                <h2 className="text-lg font-bold mb-4 text-gray-700">Información General</h2>
                <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
                    {[{
                        { name: "primerasRelacionesSexuales", label: "Primeras relaciones sexuales", placeholder: "Edad" },
                        { name: "vidaSexualActiva", label: "Vida sexual activa", placeholder: "Sí / No" },
                        { name: "numeroConyuges", label: "Número de cónyuges", placeholder: "Cantidad" },
                        { name: "preferenciasSexuales", label: "Preferencias sexuales", placeholder: "Sí / No" },
                        { name: "relacionesExtramaritales", label: "Relaciones extramaritales", placeholder: "Sí / No" },
                    ].map((field, index) => (
                        <div key={index}>
                            <label className="font-semibold text-gray-700">{field.label}:</label>
                            <input
                                name={field.name}
                                value={formData[field.name]}
                                onChange={handleChange}
                                placeholder={field.placeholder}
                                className="border p-2 w-full rounded"
                            />
                        </div>
                    )))
                </div>
            </div>
        </div>
    </div>
    /* Historial Menstrual y Gestaciones */

```

```

<div>
  <h2 className="text-lg font-bold mb-4 text-gray-700">Historial Menstrual
y Gestaciones</h2>
  <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
    {[{"name": "menarquia", "label": "Menarquía", "placeholder": "Edad"}, {"name": "cicloMenstrual", "label": "Ciclo menstrual", "placeholder": "Ciclo"}, {"name": "menopausia", "label": "Menopausia", "placeholder": "Edad"}, {"name": "fechaUltimaMenstruacion", "label": "Fecha última menstruación", "placeholder": "DD/MM/AAAA"}, {"name": "numeroGestaciones", "label": "Número de gestaciones", "placeholder": "Cantidad"}].map((field, index) => (
    <div key={index}>
      <label className="font-semibold text-gray-700">{field.label}</label>
      <input name={field.name} value={formData[field.name]} onChange={handleChange} placeholder={field.placeholder} className="border p-2 w-full rounded" />
    </div>
  )));
</div>
</div>

/* Botón Guardar */
<div className="text-center">
  <button onClick={sendRelationalData} className="mt-4 px-6 py-3 bg-blue-500 text-white rounded-lg shadow-md hover:bg-blue-600 transition">
    Guardar
  </button>
</div>
</div>
</div>
);
}

```

## ANEXO XII. Componente de examen físico con ReactJS.

```
import { useState, useEffect } from "react";
import axios from "axios";

export default function PhysicalExamForm({ patientId }) {
  const [formData, setFormData] = useState({
    presionArterial: "",
    frecuenciaCardiaca: "",
    frecuenciaRespiratoria: "",
    temperatura: "",
    saturacionOxigeno: "",
    indiceMasaCorporal: "",
    observacionesSignosVitales: "",
    cabeza: "",
    cuello: "",
    torax: "",
    abdomen: "",
    extremidades: "",
    piel: "",
    otrasObservaciones: ""
  });

  // console.log(formData);

  useEffect(() => {
    const fetchConsultation = async () => {
      try {
        const response = await axios.get(`http://localhost:5000/api/patient-analysis/${patientId}`);
        if (response.data.physical_exam) {
          setFormData(JSON.parse(response.data.physical_exam));
        }
      } catch (error) {
        console.error("Error fetching consultation:", error);
      }
    };
    if (patientId) {
      fetchConsultation();
    }
  }, [patientId]);
```

```

const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
};

const sendPhysicalExamData = async () => {
  const physicalExamData = JSON.stringify(formData);

  try {
    await axios.post("http://localhost:5000/api/patient-analysis/savePhysicalExam", {
      patient_id: patientId,
      physical_exam_records: physicalExamData,
    });
  }

  alert("Datos guardados correctamente.");
} catch (error) {
  console.error("Error al guardar los datos:", error);
  alert("No se pudo guardar la información.");
}

};

return (
  <div className="p-6 border rounded-lg shadow-md bg-white">
    <div className="space-y-6">
      <div>
        <h2 className="text-lg font-bold mb-4 text-gray-700">Signos Vitales</h2>
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
          {[{"name": "presionArterial", "label": "Presión Arterial", "placeholder": "Ej: 120/80 mmHg"}, {"name": "frecuenciaCardiaca", "label": "Frecuencia Cardíaca", "placeholder": "Ej: 72 bpm"}, {"name": "frecuenciaRespiratoria", "label": "Frecuencia Respiratoria", "placeholder": "Ej: 16 rpm"}, {"name": "temperatura", "label": "Temperatura", "placeholder": "Ej: 36.5"}, {"name": "saturacionOxigeno", "label": "Saturación de Oxígeno", "placeholder": "Ej: 98%"}, {"name": "indiceMasaCorporal", "label": "Índice de Masa Corporal", "placeholder": "Ej: 24.5"}].map((field, index) => (
        <div key={index}>
          <label className="font-semibold text-gray-700">{field.label}</label>
        </div>
      ))
    </div>
  </div>
</div>

```

```

        <input
            name={field.name}
            value={formData[field.name]}
            onChange={handleChange}
            placeholder={field.placeholder}
            className="border p-2 w-full rounded"
        />
    </div>
))
</div>
</div>

<div>
    <label className="font-semibold text-gray-700">Observaciones</label>
    <textarea
        name="observacionesSignosVitales"
        value={formData.observacionesSignosVitales}
        onChange={handleChange}
        placeholder="Ej: Posición de toma de presión arterial y otras
observaciones"
        className="border p-2 w-full rounded h-24"
    />
</div>

<div>
    <h2 className="text-lg font-bold mb-4 text-gray-700">Exploración
Física</h2>
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
        [
            { name: "cabeza", label: "Cabeza", placeholder: "Observaciones" },
            { name: "cuello", label: "Cuello", placeholder: "Observaciones" },
            { name: "torax", label: "Tórax", placeholder: "Observaciones" },
            { name: "abdomen", label: "Abdomen", placeholder: "Observaciones" },
            { name: "extremidades", label: "Extremidades", placeholder:
"Observaciones" },
            { name: "piel", label: "Piel", placeholder: "Observaciones" },
        ].map((field, index) =>
            <div key={index}>
                <label className="font-semibold text-gray-
700">{field.label}</label>
                <input
                    name={field.name}
                    value={formData[field.name]}
                    onChange={handleChange}

```

```

        placeholder={field.placeholder}
        className="border p-2 w-full rounded"
      />
    </div>
  )));
</div>
</div>

<div>
  <h2 className="text-lg font-bold mb-4 text-gray-700">Otras
Observaciones</h2>
  <textarea
    name="otrasObservaciones"
    value={formData.otrasObservaciones}
    onChange={handleChange}
    placeholder="Escribe cualquier otra observación relevante..."
    className="border p-2 w-full rounded h-24"
  />
</div>

<div className="text-center">
  <button
    onClick={sendPhysicalExamData}
    className="mt-4 px-6 py-3 bg-blue-500 text-white rounded-lg shadow-md
hover:bg-blue-600 transition"
  >
    Guardar
  </button>
</div>
</div>
);
}

```

### ANEXO XIII. Componente de examen neurológico con ReactJS.

```

import { useState, useEffect } from "react";
import axios from "axios";

export default function NeurologicalExamForm({ patientId }) {
  const [formData, setFormData] = useState({
    estadoConciencia: "",

```

```

        orientacionTiempo: "",
        orientacionEspacio: "",
        otrasObservaciones: "",
    });

// console.log(formData);

useEffect(() => {
    const fetchConsultation = async () => {
        try {
            const response = await axios.get(`http://localhost:5000/api/patient-analysis/${patientId}`);
            if (response.data.neurological_exam) {
                setFormData(JSON.parse(response.data.neurological_exam));
            }
        } catch (error) {
            console.error("Error fetching consultation:", error);
        }
    };
    if (patientId) {
        fetchConsultation();
    }
}, [patientId]);

const handleChange = (e) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
};

const sendNeuroData = async () => {
    try {
        await axios.post("http://localhost:5000/api/patient-analysis/saveNeurologicalExam", { patient_id: patientId, ...formData });
        alert("Datos guardados correctamente");
    } catch (error) {
        console.error("Error al guardar:", error);
        alert("Error al guardar los datos");
    }
};

return (
    <div className="p-6 border rounded-lg shadow-md bg-white">
        <div className="space-y-6">
            {/* SECCIÓN: Conciencia y Orientación */}
            <div>

```

```

        <h2 className="text-lg font-bold mb-4 text-gray-700">Conciencia y
Orientación</h2>
        <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">
        {[

            { name: "estadoConciencia", label: "Estado de conciencia",
placeholder: "Ej: Alerta, somnoliento" },
            { name: "orientacionTiempo", label: "Orientación en tiempo",
placeholder: "Ej: Orientado / Desorientado" },
            { name: "orientacionEspacio", label: "Orientación en espacio",
placeholder: "Ej: Orientado / Desorientado" },
        ].map((field, index) => (
            <div key={index}>
                <label className="font-semibold text-gray-
700">{field.label}</label>
                <input name={field.name} value={formData[field.name]}>
                onChange={handleChange} placeholder={field.placeholder} className="border p-2 w-
full rounded" />
            </div>
        )));
        </div>
    </div>

    {/* SECCIÓN: Otras Observaciones */}
    <div>
        <h2 className="text-lg font-bold mb-4 text-gray-700">Otras
Observaciones</h2>
        <textarea name="otrasObservaciones" value={formData.otrasObservaciones}>
        onChange={handleChange} placeholder="Escribe cualquier otra observación
relevante..." className="border p-2 w-full rounded h-24" />
    </div>

    {/* Botón de Guardar */}
    <div className="text-center">
        <button onClick={sendNeuroData} className="mt-4 px-6 py-3 bg-blue-500
text-white rounded-lg shadow-md hover:bg-blue-600 transition">
            Guardar
        </button>
    </div>
</div>
</div>
);
}

```

#### ANEXO XIV. Componente de carga de archivos con ReactJS.

```
import { useState, useEffect } from "react";
import axios from "axios";

export default function FileUpload({ patientId }) {
  const [fileData, setFileData] = useState();

  const handleChange = (e) => {
    setFileData(e.target.files[0]);
  };

  const sendFile = async (e) => {
    e.preventDefault();
    var fileForm = new FormData();
    // console.log(fileData);
    fileForm.append("file", fileData);
    fileForm.append("patientId", patientId);

    try {
      await axios.post("http://localhost:5000/api/patient-analysis/upload",
fileForm);

      alert("Archivos guardados correctamente.");
    } catch (error) {
      console.error("Error al guardar los archivos:", error);
      alert("No se pudo guardar la información.");
    }
  };
}

return (
  <div className="p-6 border rounded-lg shadow-md bg-white">
    <form className="space-y-6">
      <h2 className="text-lg font-bold mb-4 text-gray-700">Gargue de
archivos</h2>
      <div>
        <label className="font-semibold text-gray-700">Cargar archivos de
estudios/bioseñales</label>
        <input
          id="fileInput"
          type="file"
          name="file"
          onChange={handleChange}
        </div>
    </form>
  </div>
)
```

```

        placeholder="Subir archivo"
        className="border p-2 w-full rounded"
      />
    </div>

    <div className="text-center">
      <button
        onClick={sendFile}
        className="mt-4 px-6 py-3 bg-blue-500 text-white rounded-lg shadow-md
        hover:bg-blue-600 transition"
      >
        Guardar
      </button>
    </div>
  </form>
</div>
);
}

```

#### ANEXO XV. Creación de tabla *patient\_analysis*.

```

CREATE TABLE `patient_analysis` (
  `patient_id` int(11) NOT NULL,
  `patient_data` text DEFAULT NULL,
  `consultation` text DEFAULT NULL,
  `general_records` text DEFAULT NULL,
  `personal_records` text DEFAULT NULL,
  `relational_records` text DEFAULT NULL,
  `physical_exam` text DEFAULT NULL,
  `neurological_exam` text DEFAULT NULL,
  `file_records` text DEFAULT NULL,
  `created_at` timestamp NULL DEFAULT current_timestamp(),
  PRIMARY KEY (`patient_id`),
  CONSTRAINT `fk_patient` FOREIGN KEY (`patient_id`) REFERENCES `patient`(`patient_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci

```

ANEXO XVI. Tabla de documentación cualitativa de la adaptación del software seleccionado.

<b>Documentación Cualitativa - Adaptación del software seleccionado</b>	
Aspecto analizado	
Descripción del proceso realizado	
Ventajas observadas	
Limitaciones encontradas	
Configuraciones específicas realizadas	
Propuestas de mejora	

ANEXO XVII. Métodos de carga y nombrado de archivos.

```
const multer = require('multer');

const storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, 'uploads/') // Destination folder
  },
  filename: function (req, file, cb) {
    cb(null, Date.now() + '-' + file.originalname) // Naming the file
  }
})
const upload = multer({ storage })

module.exports = { upload };
```