DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN Y UNIFICACIÓN DE BASES DE DATOS MÉDICAS

DEVELOPMENT OF AN APPLICATION FOR THE MANAGEMENT AND UNIFICATION OF MEDICAL DATABASES

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTOR
LAURA REYES CABALLERO

DIRECTOR
ANTONIO SARASA CABAZUELO

GRADO EN INGENIERÍA DE COMPUTADORES
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE UNA APLICACIÓN PARA LA GESTIÓN Y UNIFICACIÓN DE BASES DE DATOS MÉDICAS

DEVELOPMENT OF AN APPLICATION FOR THE MANAGEMENT AND UNIFICATION OF MEDICAL DATABASES

TRABAJO DE FIN DE GRADO DE INGENIERÍA DE COMPUTADORES

CALIFICACIÓN: 8,5

AUTOR
LAURA REYES CABALLERO

DIRECTOR
ANTONIO SARASA CABAZUELO

CONVOCATORIA: JUNIO 2025

GRADO EN INGENIERÍA DE COMPUTADORES
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

17 DE MAYO DE 2025

AGRADECIMIENTOS

Quiero agradecer y dedicar unas palabras a mi padre. Catedrático de la Universidad Complutense, me enorgullece enormemente graduarme en la misma institución en la que volcaste tanto esfuerzo, pasión y amor por el conocimiento y la enseñanza. Siempre fuiste un ejemplo de dedicación, de trabajar con vocación y de hacer las cosas desde el compromiso y el corazón.

Gracias por transmitirme tu generosidad, tus ganas y tu actitud positiva ante todo, y, sobre todo, por quererme, apoyarme y educarme de la mejor forma que podría imaginar.

Aunque no estés presente en esta etapa de mi vida, que tanta ilusión te hubiese hecho, estás conmigo en cada logro.

Y, sobre todo, gracias mamá. Por ser mi apoyo incondicional, por el amor que me das cada día. Por estar siempre a mi lado, confiar en mí incluso cuando yo no lo hacía, y por recordarme una y otra vez que puedo con todo. Gracias por ver siempre lo mejor en mí, por tu infinita paciencia, tu fuerza y tu manera de cuidar.

RESUMEN

Aplicación para la unificación y gestión de bases de datos médicas

Este trabajo de Fin de Grado presenta el desarrollo de una aplicación para la unificación y gestión de bases de datos médicas relacionadas con trasplantes renales. La herramienta permite a los administradores de distintos hospitales importar sus propias bases de datos, independientemente del formato, y así consolidar la información en una base de datos unificada. Además, un mismo hospital puede añadir nuevos archivos que contengan información adicional sobre pacientes previamente registrados, lo que permite centralizar y acceder fácilmente a todos los datos disponibles.

La aplicación incorpora un sistema de consultas avanzadas que permite filtrar los datos mediante condiciones personalizables. Ofrece también la posibilidad de obtener estadísticas generales de la aplicación y base de datos unificada, y exportar tanto la base de datos completa como subconjuntos limitados por los resultados de las consultas. Estas funcionalidades hacen de la herramienta una solución útil no solo para la gestión clínica, sino también para el análisis de datos y la investigación médica.

Palabras clave

Base de datos, hospital, aplicación web, unificación, consultas, investigación clínica.

ABSTRACT

Application for the unification and management of medical databases

This Bachelor's Thesis presents the development of an application for the unification and management of medical databases related to kidney transplants. The tool allows administrators from different hospitals to import their own databases, regardless of the format, and consolidate the information into a unified database. Additionally, a single hospital can add new files containing further information about previously registered patients, enabling centralized and easy access to all available data.

The application includes an advanced query system that allows data to be filtered using customizable conditions. It also offers the possibility to generate reports, obtain statistics, and export both the complete database and subsets based on query results. These features make the tool a valuable solution not only for clinical management but also for data analysis and medical research.

Keywords

Database, hospital, web application, unification, queries, clinical research.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Motivación

En el ámbito médico, la disponibilidad y correcta gestión de los datos clínicos es fundamental. Tanto para garantizar una atención eficiente a los pacientes como para impulsar la investigación sanitaria. Sin embargo, uno de los principales desafíos que enfrentan los profesionales sanitarios y los investigadores es la fragmentación de la información. Esto se debe a que diferentes hospitales almacenan sus datos clínicos en distintos formatos, y muchos hospitales tienen múltiples archivos independientes de datos sobre pacientes. Esta dispersión dificulta enormemente la consolidación de la información, su análisis conjunto y la extracción de conclusiones relevantes a partir de la información disponible.

A partir de esta necesidad surge la motivación principal de este Trabajo de Fin de Grado: desarrollar una herramienta que permita unificar bases de datos médicas heterogéneas, facilitando así la integración y consulta eficiente de la información clínica. La aplicación, denominada *UniMed*, busca no solo resolver problemas técnicos relacionados con la compatibilidad entre base de datos, sino también ofrecer una plataforma accesible que facilite el trabajo colaborativo entre centros sanitarios y fomente la investigación basada en datos. De este modo, UniMed aspira a convertirse en un recurso valioso para mejorar la gestión clínica y apoyar el avance científico en el ámbito de los trasplantes renales.

Dado que la herramienta UniMed maneja información médica sensible, la seguridad y protección de los datos son aspectos a tener en cuenta. El sistema restringe el acceso a datos sensibles en función del hospital al que pertenece el usuario, de modo que sólo el personal autorizado de cada centro puede visualizar su información específica. Además, la gestión de usuarios está centralizada en administradores, quienes controlan la asignación de permisos para garantizar que el acceso a los datos sea adecuado.

1.2 Objetivos

Esta herramienta tiene como finalidad facilitar la integración de datos procedentes de diferentes hospitales y fuentes, superando los problemas derivados de la heterogeneidad en el formato y estructura de las bases de datos.

Para alcanzar este objetivo general, se plantearon varios objetivos específicos. En primer lugar, diseñar una interfaz de usuario intuitiva que permita a los administradores importar sus propias bases de datos, mapear las columnas y consolidar la información en una base de datos unificada.

Además, implementar un sistema de identificación que permita añadir nuevos pacientes o ampliar la información de pacientes ya existentes, garantizando la consistencia de los datos.

Un componente clave de la aplicación, es el desarrollo de un módulo de consultas avanzadas que permita aplicar filtros personalizados y realizar búsquedas complejas sobre los datos unificados.

La herramienta también incluirá funcionalidades para la visualización de estadísticas generales de la aplicación y exportación de resultados.

1.3 Plan de trabajo

El desarrollo del sistema comenzó con un análisis de requisitos, cuyo objetivo fue comprender en profundidad las necesidades del proyecto, definir claramente las funcionalidades principales y establecer las bases sobre las que se construiría la aplicación. A partir de este análisis, se dividieron las funcionalidades en distintas iteraciones, cada una con una fecha orientativa de finalización. Estas iteraciones incluyeron: gestión de usuarios, gestión de la base de datos, explotación de la base de datos y gestión de roles.

Una vez delimitadas las funcionalidades, se abordó el diseño de la arquitectura del proyecto, descrita detalladamente en la 4.1. En paralelo, se seleccionaron las tecnologías y herramientas necesarias para el desarrollo, que se presentan, más adelante, en el Capítulo 3 - También fue fundamental definir la estructura de la base de

datos, estableciendo sus distintas tablas y relaciones, aspecto que se analiza en profundidad en la 4.2.

En la Figura 1-1 se muestra un diagrama de Gantt que representa de forma visual la planificación temporal del proyecto, la división del trabajo descrita previamente.

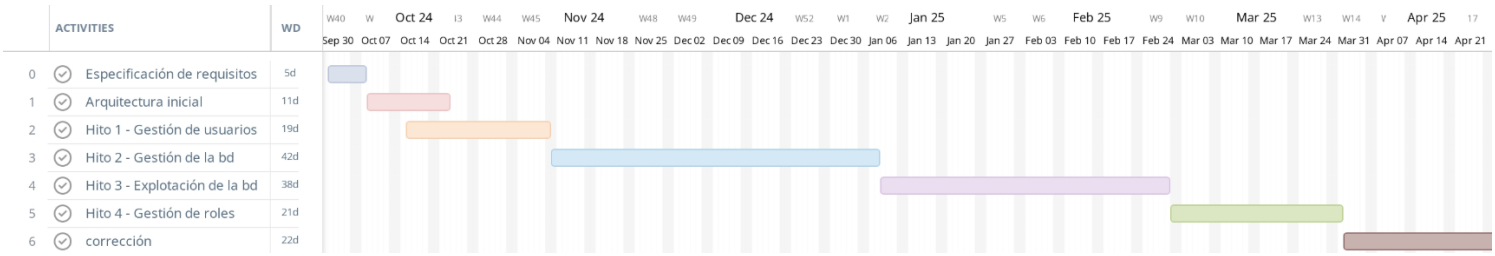


Figura 1-1. Diagrama de Gantt

Introduction

Motivation

In the medical field, the availability and proper management of clinical data is essential not only to ensure efficient patient care, but also to drive healthcare research. However, one of the main challenges faced by healthcare professionals and researchers is the fragmentation of information. This is largely due to the fact that different hospitals store clinical data in various formats, and many of them manage multiple independent files containing patient information. This dispersion significantly hinders the consolidation of data, joint analysis, and the extraction of meaningful conclusions from the available information.

This need forms the primary motivation behind this Bachelor's Thesis: to develop a tool capable of unifying heterogeneous medical databases, thereby facilitating the integration and efficient querying of clinical information. The application, named UniMed, aims not only to address technical issues related to database compatibility, but also to provide an accessible platform that encourages collaborative work between healthcare centers and promotes data-driven research. In doing so, UniMed aspires to become a valuable tool for improving clinical data management and supporting scientific progress in the field of kidney transplantation.

Since the UniMed tool handles sensitive medical information, data security and protection are critical aspects to consider. The system restricts access to sensitive data based on the hospital to which the user belongs, ensuring that only authorized personnel from each center can view their specific information. Additionally, user management is centralized with administrators who control the assignment of permissions to guarantee appropriate data access.

Goals

The main purpose of this tool is to facilitate the integration of data from different hospitals and sources, overcoming issues arising from the heterogeneity in the format and structure of medical databases.

To achieve this overall goal, several specific objectives were defined. First, to design an intuitive user interface that enables administrators to import their own databases, map columns, and consolidate the information into a unified database.

In addition, to implement an identification system that allows new patient records to be added or existing ones to be expanded, ensuring data consistency.

A key component of the application will be the development of an advanced query module that enables customized filtering and complex searches across the unified data.

The tool will also include features for visualization of general application statistics, and export of results.

Workplan

The development of the system began with a requirements analysis, aimed at thoroughly understanding the project's needs, clearly defining its main functionalities, and establishing the foundation upon which the application would be built. Based on this analysis, the functionalities were divided into different iterations, each with a tentative completion date. These iterations included: user management, database management, database exploitation, and role management.

Once the functionalities were defined, the architectural design of the project was addressed, which is described in detail in 4.1. In parallel, the technologies and tools necessary for development were selected, as presented later in Capítulo 3 - . It was also essential to define the database structure, establishing its various tables and relationships, a topic discussed in depth in 4.2.

Figure 1.1 presents a Gantt chart that visually represents the project’s timeline and the previously described division of work.

Capítulo 2 - Estado de la cuestión

Antes de iniciar el desarrollo del sistema, se llevó a cabo una revisión de herramientas existentes que abordan necesidades similares, con el propósito de identificar soluciones ya implantadas y valorar en qué medida podían servir como inspiración, punto de partida o comparación. Esta exploración permitió comprender cómo se están resolviendo problemas parecidos en otros contextos.

2.1 REDCap

Una de las plataformas más destacadas encontradas durante esta revisión fue REDCap (Research Electronic Data Capture) [1], un sistema ampliamente utilizado en entornos clínicos y de investigación biomédica. REDCap permite a los usuarios diseñar formularios de recolección de datos, gestionar proyectos colaborativos y exportar la información recopilada para su análisis. Sin embargo, está principalmente orientado a capturar datos nuevos a través de formularios definidos previamente, y no a integrar, mapear o transformar datos ya existentes procedentes de fuentes distintas. Esto lo hace poco adecuado cuando se necesita unificar múltiples bases de datos con estructuras diversas, como ocurre en el caso del presente proyecto.

Además, aunque ofrece opciones de búsqueda y filtros básicos, no proporciona un entorno flexible e intuitivo para construir consultas complejas o personalizadas. Tampoco permite modificar libremente su arquitectura interna para ampliar funcionalidades específicas. Estas limitaciones, comunes también en otras herramientas similares, reforzaron la decisión de desarrollar un sistema propio desde cero, adaptado a los requisitos concretos del proyecto tanto en términos de integración de datos como de su posterior explotación.

2.2 OpenClinica

Otra herramienta analizada fue OpenClinica [2], una plataforma de código abierto orientada a la gestión de ensayos clínicos y la captura electrónica de datos (EDC). OpenClinica destaca por ofrecer un entorno robusto para la definición de estudios, el seguimiento de pacientes y la validación de datos, cumpliendo con

estándares regulatorios exigentes. No obstante, al igual que otras soluciones del ámbito clínico, está centrada principalmente en la entrada controlada de datos durante estudios diseñados desde cero.

A pesar de su robustez, OpenClinica presenta limitaciones importantes en el contexto de integración flexible de datos médicos heterogéneos. Aunque permite la carga de datos por lotes y la integración mediante API, no está diseñada para unificar automáticamente archivos con estructuras distintas o provenientes de múltiples formatos. Los datos deben estar previamente estructurados y adaptados al diseño del estudio, lo que implica una normalización técnica que puede ser compleja y poco accesible para muchos centros hospitalarios.

2.3 I2b2

i2b2 (Informatics for Integrating Biology and the Bedside) [3] es una plataforma de código abierto desarrollada por la Universidad de Harvard, pensada para facilitar el uso secundario de datos clínicos con fines de investigación biomédica. Es decir, permite a los investigadores aprovechar la información recogida habitualmente en hospitales —como diagnósticos, tratamientos, resultados de laboratorio, edad o sexo— para realizar estudios, identificar grupos de pacientes con ciertas características, y analizar patrones clínicos.

Uno de los puntos fuertes de i2b2 es que no solo almacena datos, sino que los organiza y estructura de forma que se puedan consultar fácilmente. Por ejemplo, agrupa la información en categorías como medicación, procedimientos, visitas médicas, condiciones clínicas o datos demográficos. A través de su interfaz gráfica, los usuarios pueden construir consultas complejas sin necesidad de escribir código, seleccionando combinaciones de criterios clínicos para filtrar pacientes que cumplan ciertas condiciones. Esta funcionalidad lo convierte en una herramienta muy útil para estudios de viabilidad o generación de hipótesis.

Un aspecto que diferencia a i2b2 de otras plataformas más centradas en la recogida de datos, como REDCap o OpenClinica, es que i2b2 está diseñada para integrar datos que ya existen, aunque vengan de sistemas distintos o con formatos heterogéneos. Para ello, permite definir mapeos y jerarquías de conceptos a través de

una ontología propia. Sin embargo, esta capacidad de integración no es automática: requiere un proceso técnico previo que implica transformar los datos originales (por ejemplo, desde una base de datos hospitalaria) a un formato compatible con la estructura interna de i2b2. También es necesario construir una ontología personalizada que dé sentido a esos datos dentro del sistema.

En resumen, i2b2 es una herramienta potente para investigación clínica con datos reales, pero su puesta en marcha implica una configuración inicial compleja, que demanda conocimientos técnicos y una inversión considerable de tiempo.

2.4 Análisis comparativo de soluciones existentes

Una vez analizadas las principales herramientas utilizadas en el ámbito de la gestión y explotación de datos clínicos (REDCap, OpenClinica e i2b2), resulta útil realizar una comparativa directa entre ellas. Esta comparativa permite visualizar de forma clara las principales diferencias en cuanto a funcionalidades, facilidad de integración, orientación al usuario, y capacidades analíticas.

El objetivo de esta sección es destacar no solo las fortalezas de cada solución, sino también las carencias o limitaciones que motivaron la necesidad de desarrollar una herramienta propia, más ajustada a las necesidades específicas del proyecto. A continuación, se presenta una tabla, Tabla 2-1, que resume los aspectos más relevantes evaluados en cada una de estas plataformas.

Como se puede ver en la comparativa, aunque herramientas como REDCap, OpenClinica o i2b2 ofrecen funcionalidades útiles, ninguna se adapta por completo a las necesidades concretas de este proyecto. Todas presentan ciertas limitaciones, ya sea en la integración flexible de bases de datos heterogéneas, en la personalización del filtrado, o en la sencillez de uso.

Además, este sistema busca centralizar la información clínica de los pacientes procedente de distintas fuentes, de forma unificada y accesible, algo que estas herramientas no resuelven de forma directa o sencilla. Por ello, se optó por desarrollar una aplicación propia que permitiera un mayor control sobre el tratamiento de los datos, facilitando tanto su integración como su análisis posterior.

Característica	REDCap	OpenClinica	I2b2
Enfoque principal	Recogida y gestión de datos clínicos	Ensayos clínicos y recogida regulada	Reutilización de datos clínicos para investigación
Gestión de formularios	Sí, con interfaz gráfica intuitiva	Sí, detallado y orientado a estudios clínicos	No orientado a recogida directa; trabaja con datos ya existentes
Consultas avanzadas	Limitadas; más enfocadas a exportación	Limitadas; orientadas a informes y monitoreo	Sí; interfaz gráfica para filtrado complejo de cohortes
Visualización de datos	Básica (tablas, reportes simples)	Básica; informes predefinidos	Moderada; incluye cuadros de mando de cohortes
Unificación de bases de datos	No integrada de forma nativa	No integrada; requiere procesos ETL externos	Sí, mediante ontologías y transformación previa
Mapeo de columnas	No; requiere coincidencia exacta de estructura	No; requiere normalización previa	Sí, pero requiere configuración técnica compleja
Facilidad de uso	Alta; orientado a usuarios clínicos sin conocimientos técnicos	Media; requiere formación y conocimientos del dominio clínico	Baja; requiere conocimientos técnicos en bases de datos y ontologías

Tabla 2-1. Comparativa de herramientas

Capítulo 3 - Tecnologías y herramientas empleadas

El desarrollo de UniMed ha requerido el uso de distintas tecnologías, herramientas y librerías que han facilitado la implementación, gestión y despliegue del proyecto. A continuación, se describen las principales.

3.1 Entorno de desarrollo

Para el desarrollo de este proyecto se ha utilizado Visual Studio Code (VS Code) [4] como entorno principal de trabajo. Se trata de un editor ampliamente extendido en el ámbito del desarrollo web, elegido por su facilidad de uso, su rapidez y la gran variedad de funciones que ofrece gracias a sus extensiones.

VS Code permite adaptar el entorno a las necesidades concretas del proyecto, facilitando tareas como la organización del código, la depuración, el control de versiones o el trabajo con estructuras por carpetas. Su interfaz intuitiva y personalizable ha contribuido a mantener una estructura clara durante el desarrollo, mejorando así la productividad y la experiencia general de trabajo.

3.2 Control de versiones

Para la gestión del código y el control de versiones se ha utilizado Git [5], una herramienta ampliamente extendida en el desarrollo de software por su fiabilidad y capacidad para mantener un historial completo de cambios. Git permite trabajar de manera estructurada, facilitando la creación de ramas, para el desarrollo de nuevas funcionalidades o la corrección de errores, sin afectar a la versión principal del proyecto.

El repositorio se ha alojado en GitHub [6], lo que ha permitido una gestión remota segura del código y su sincronización entre dispositivos. A lo largo del desarrollo se ha trabajado principalmente con dos ramas: una rama principal (master) que contiene la versión estable del proyecto, y una rama secundaria dedicada al desarrollo activo, donde se implementaban y probaban los cambios antes de integrarlos a la rama principal.

Además, se ha utilizado la aplicación Fork [7] como cliente de Git para escritorio. Esta herramienta ofrece una interfaz visual que ha facilitado la gestión del repositorio,

haciendo más accesibles tareas como realizar commits, merges o la navegación entre ramas.

3.3 Lenguajes de programación

En el desarrollo de la aplicación se han utilizado varios lenguajes de programación, tanto para la parte del servidor como para la interfaz de usuario. Para la lógica de la aplicación y la gestión del servidor se ha empleado PHP, un lenguaje ampliamente utilizado en aplicaciones web, que permite una integración sencilla con bases de datos y un desarrollo ágil del backend.

Para la creación de la interfaz de usuario, se emplearon HTML y CSS para la estructura y el diseño visual, mientras que JavaScript se utilizó para la interactividad y la gestión dinámica de los elementos en pantalla.

3.4 Frameworks y librerías

Para estructurar el backend de la aplicación se ha empleado Slim Framework [8], una herramienta ligera y flexible basada en PHP que permite gestionar rutas y peticiones de manera eficiente. Se ha utilizado para definir las distintas rutas de acceso a la aplicación, asignar cada una a un controlador específico y gestionar la lógica de autenticación mediante middleware. Esto ha permitido mantener una organización clara del código y separar adecuadamente la lógica de negocio de la presentación.

El patrón seguido se aproxima a una estructura por capas, donde Slim actúa como intermediario entre las rutas y los controladores, y estos a su vez se encargan de manipular los datos y preparar las vistas. Además, se ha utilizado el contenedor de dependencias de Slim para configurar y renderizar las vistas mediante la librería PHP-View, que permite mostrar archivos .php como plantillas, facilitando así la separación entre lógica y presentación.

Para la importación de bases de datos en formato Excel, se ha utilizado la librería PhpSpreadsheet [9], que permite leer, procesar y extraer datos de archivos .xlsx y .xls. Esta funcionalidad es clave en la aplicación, ya que gran parte de los datos médicos que se desean unificar provienen de hojas de cálculo.

Estas herramientas han sido seleccionadas por su compatibilidad, ligereza y facilidad de integración en el entorno PHP. Gracias a ellas ha sido posible desarrollar una aplicación estructurada, modular y eficiente.

3.5 Base de datos

Para la gestión de datos, la aplicación utiliza un sistema de bases de datos relacional basado en MariaDB, que es una bifurcación de MySQL ampliamente compatible y utilizada en entornos de desarrollo web. La administración de la base de datos se ha realizado a través de phpMyAdmin [10], una herramienta de administración visual que facilita tareas como la creación de esquemas, inserción de datos o ejecución de consultas SQL desde una interfaz web.

Todo el entorno de servidor se ha ejecutado mediante XAMPP [11], un paquete que integra en un solo instalador Apache (servidor web), MariaDB (base de datos), PHP y Perl. Su uso ha permitido lanzar el proyecto localmente de forma sencilla, sin necesidad de instalar y configurar cada componente por separado. Con solo iniciar XAMPP, se activa tanto el servidor como la base de datos, lo que agiliza el desarrollo y las pruebas.

3.6 Recursos externos

En la interfaz de usuario, además de HTML, CSS y JavaScript, se emplearon diversas librerías y recursos externos para mejorar la experiencia visual y la usabilidad. Entre ellas destacan las librerías de iconos Boxicons [12] y Font Awesome [13], que proporcionan iconos vectoriales claros y modernos para enriquecer la interfaz gráfica. También se utilizó la fuente tipográfica Roboto [14], importada desde Google Fonts, que contribuye a una presentación limpia y legible. Para optimizar la interacción en elementos como los selectores desplegables, se incorporó la librería Choices.js [15], la cual ofrece funcionalidades avanzadas como búsqueda y selección múltiple, mejorando así la experiencia del usuario durante el filtrado y la consulta de datos. Además, para la visualización dinámica e interactiva de datos se integró la librería Chart.js [16], que permite crear gráficos modernos y responsivos, facilitando la interpretación visual de la información clínica en la aplicación.

Capítulo 4 - Arquitectura y modelo de datos

4.1 Arquitectura general

La aplicación ha sido desarrollada siguiendo una arquitectura multicapa, un enfoque que divide el sistema en diferentes capas con responsabilidades bien diferenciadas. Esta organización permite una mayor claridad estructural, facilita el mantenimiento del código y promueve la escalabilidad del proyecto.

Dentro de este enfoque, también se sigue el patrón de diseño Modelo-Vista-Controlador (MVC), ampliamente utilizado en el desarrollo de aplicaciones web. Este patrón permite dividir la lógica de la aplicación en capas bien definidas, separando la lógica de presentación, la lógica de negocio y el acceso a datos.

4.1.1 Capas

4.1.1.1 Controladores

Los controladores son los responsables de recibir y gestionar las peticiones del usuario. En función del tipo de solicitud, los controladores procesan los datos de entrada, coordinan la lógica necesaria a través de modelos y repositorios, y determina qué vista debe ser mostrada como respuesta. Cada controlador está enfocado a una funcionalidad concreta de la aplicación (usuarios, importación, visualización, etc.).

4.1.1.2 Modelos

La capa de modelos representa las entidades principales del sistema. Estas clases definen la estructura de los datos con sus propiedades y métodos asociados, y se utilizan para transportar información de forma coherente entre las distintas capas de la aplicación. Estas clases encapsulan la información y la lógica asociada a cada tipo de dato.

4.1.1.3 Repositorios

El acceso a la base de datos está centralizado en los repositorios. Cada repositorio contiene los métodos necesarios para realizar operaciones CRUD (crear, leer,

actualizar y eliminar) sobre la base de datos de forma estructurada. Están asociados a un modelo concreto, el cual guarda los resultados obtenidos transformados a dichos objetos.

4.1.1.4 Vistas

La capa de vistas se encarga de presentar los datos al usuario final. Estas plantillas contienen únicamente lógica de presentación, sin realizar ningún tipo de procesamiento de datos. Reciben la información ya preparada por los controladores y la muestran de forma estructurada y coherente.

4.1.2 Framework Slim

La aplicación utiliza el framework Slim, un microframework para PHP que permite definir rutas HTTP y gestionarlas de forma sencilla. A través de Slim, se configuran los endpoints de la aplicación, especificando el método HTTP (GET, POST, etc.), la ruta y el controlador que debe encargarse de la petición. Además, se define el middleware general de la aplicación, como la gestión de sesiones o la verificación del inicio de sesión del usuario.

4.1.3 Organización del código

El código fuente de la aplicación está estructurado de forma modular. La organización sigue una convención común en proyectos PHP que utilizan un enfoque multicapa, y gestión de dependencias mediante Composer.

4.1.3.1 Public/

Este directorio contiene el punto de entrada de la aplicación, a través del archivo index.php con Slim. También se incluyen aquí los recursos accesibles públicamente como hojas de estilo, scripts e imágenes. Es la única carpeta que debe estar expuesta directamente al servidor web.

4.1.3.2 Src/

Contiene el núcleo de la lógica de la aplicación. Está dividido en subdirectorios según la función de cada componente:

- Controllers/: agrupa los distintos controladores que gestionan las peticiones del usuario y definen el flujo lógico del sistema.
- Models/: incluye las clases que representan las entidades del dominio, utilizadas para transportar datos de forma estructurada.
- Repositories/: contiene las clases que encapsulan la lógica de acceso a la base de datos.
- Views/: almacena las plantillas de presentación que generan la salida HTML.

Esta separación respeta el patrón arquitectónico seguido y permite mantener una clara responsabilidad en cada módulo.

4.1.3.3

4.1.3.4 Vendor/

Este directorio es generado automáticamente por Composer al instalar las dependencias del proyecto. Contiene el código de terceros, como el framework Slim y otras librerías utilizadas por la aplicación. No debe modificarse manualmente.

4.1.3.5 Archivos en la raíz del proyecto

Además de las carpetas principales, existen varios archivos clave en la raíz del proyecto:

- Composer.json: define las dependencias del proyecto y la configuración de autoload. Es utilizado por Composer para gestionar paquetes externos.
- Composer.lock: registra la versión exacta de cada dependencia instalada para garantizar entornos consistentes.
- Config.php: contiene la configuración global de la aplicación, como los parámetros de conexión a la base de datos u otras constantes de uso común.

4.2 Base de datos

La base de datos constituye uno de los pilares fundamentales del sistema desarrollado. Es el componente encargado de almacenar y estructurar toda la información clínica, organizativa y de usuario que permite el funcionamiento de la aplicación. Para dar soporte a la lógica de negocio, se ha diseñado un esquema relacional que permite representar de forma coherente y escalable los distintos elementos implicados: usuarios, hospitales, pacientes y datos clínicos.

4.2.1 Hospitales

La tabla hospitales recoge los centros participantes en el sistema. Cada hospital tiene un identificador único, nombre y un campo con el correo electrónico del administrador correspondiente. Este correo es clave para el proceso de registro: si un usuario intenta registrarse con ese correo, el sistema reconoce que debe tener el rol de administrador y le otorga los permisos adecuados. Esta validación permite establecer un control de acceso seguro, garantizando que solo un responsable pueda gestionar los datos del centro.

4.2.2 Usuarios

La tabla usuarios gestiona la información asociada al acceso a la plataforma y a los distintos perfiles de usuario. Incluye campos como el id, nombre completo, nombre de usuario, correo electrónico, contraseña cifrada, rol (administrador, médico o investigador), identificador del hospital al que está vinculado y un campo booleano (registrado) que indica si el usuario ha completado el proceso de registro.

De todos estos campos, únicamente id, email y rol son obligatorios. El campo id es autoincremental y actúa como clave primaria. El campo email es único, lo que evita la duplicación de direcciones y permite un control más riguroso del acceso. El rol, por defecto, se establece como investigador, a menos que se indique lo contrario.

Este comportamiento está diseñado para permitir el registro progresivo: cuando un administrador introduce manualmente un correo electrónico de un médico, se genera un registro parcial en esta tabla, que contiene únicamente el id, email, rol (en este caso, médico) y el identificador del hospital correspondiente. Posteriormente,

cuando ese médico accede a la plataforma e inicia el proceso de registro, el sistema completa automáticamente su perfil con los datos personales proporcionados. En cambio, si un usuario intenta registrarse sin que su correo haya sido previamente autorizado por un administrador, se le asigna automáticamente el rol de investigador, con permisos más restringidos.

4.2.3 Pacientes

La tabla pacientes contiene información básica sobre los pacientes registrados por cada hospital. Cada entrada incluye un identificador único, el nombre del paciente, sus iniciales y la fecha del trasplante. La combinación de iniciales y fecha está restringida para que sea única, lo que permite evitar duplicidades.

4.2.4 infoColumnas

La tabla infoColumnas actúa como metacapa para describir las variables clínicas importadas. Por cada columna de los datos cargados por los hospitales, se registra aquí su nombre técnico, una descripción legible, el tipo de dato (por ejemplo, texto, número o fecha), y un campo booleano (visible) que permite a los administradores determinar si esa columna debe mostrarse en la interfaz. De este modo, se ofrece una herramienta de personalización que mejora la claridad y pertinencia de la visualización de datos para distintos usuarios.

4.2.5 tablaDatos

La tabla tablaDatos almacena los valores clínicos reales vinculados a cada paciente. Su diseño se ha concebido para ser flexible y escalable: contiene de forma fija el identificador del hospital y el identificador del paciente, que establecen relaciones con las tablas Hospitales y Pacientes, respectivamente. A partir de esta estructura base, las columnas correspondientes a variables clínicas se añaden dinámicamente en el momento en que se importan nuevos archivos proporcionados por los distintos centros.

Este enfoque permite consolidar en una misma tabla información de diferentes hospitales, con estructuras heterogéneas, sin necesidad de rediseñar el modelo para

cada caso. Además, al estar separada de la tabla de metadatos (infoColumnas), se facilita la integración, visualización y mantenimiento de los datos.

4.2.6 Diagrama

En la Figura 4-1. Diagrama de base de datosFigura 4-1, se muestra el correspondiente diagrama de la base de datos, que ilustra de forma visual la estructura general del modelo, así como las relaciones entre las tablas descritas en los apartados anteriores.

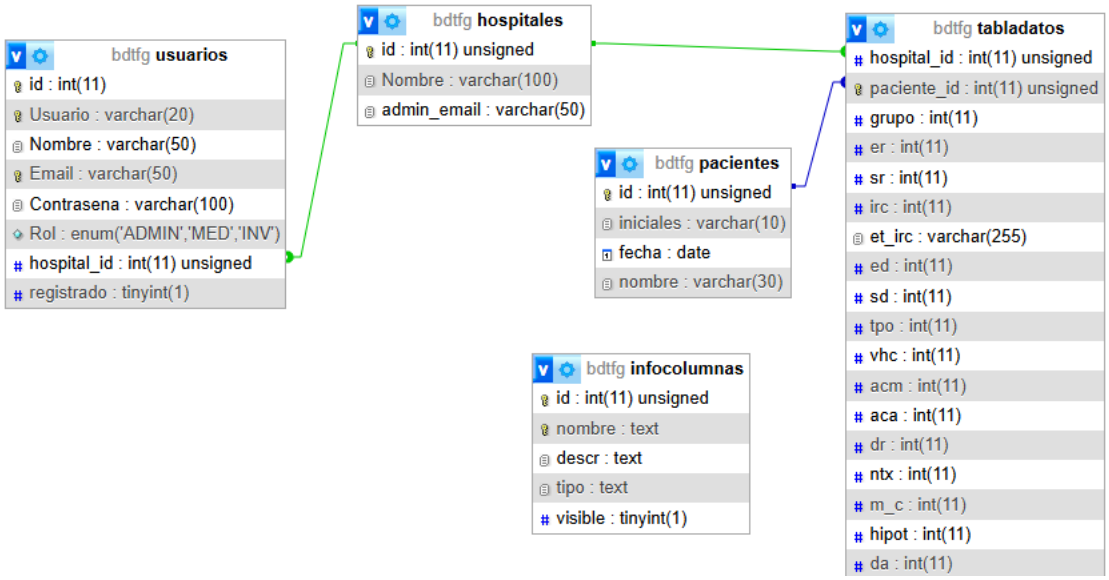


Figura 4-1. Diagrama de base de datos

Capítulo 5 - Diseño e implementación

Este capítulo se centra en describir algunos de los procesos internos más relevantes y complejos de cada módulo. Aquellos que requieren más transformaciones, verificaciones o una lógica particular para garantizar su correcto funcionamiento, y que por ello merecen una explicación específica.

5.1 Módulo de unificación

Este módulo comprende la inserción de un archivo, mapeo de sus columnas con las de la base de datos y adición de columnas nuevas. Es decir, todo el proceso relativo a unificar un archivo con la base de datos general.

5.1.1 Lectura de archivo

La aplicación permite solo archivos con las siguientes extensiones: .xls, .xlsx, .csv y .txt, evitando así que se importen formatos incompatibles. Una vez se ha importado un archivo válido, el sistema identifica automáticamente su extensión. En el caso de archivos .xls o .xlsx, se utiliza la librería PhpSpreadsheet, que mantiene el formato estructurado. En cambio, si el archivo es de tipo .csv o .txt, se realiza un análisis inicial para determinar el delimitador de columnas. Para ello, el sistema toma una muestra de líneas del archivo y cuenta la frecuencia de aparición de cada delimitador posible. A continuación, selecciona como delimitador el carácter más frecuente entre los predefinidos, como se muestra en la Figura 5-1.

```php
private function leerArchivo($archivo) {
    $extension = pathinfo($archivo, PATHINFO_EXTENSION);

    $result = $extension == 'csv' || $extension == 'txt' ? $this->leerCSV($archivo) : $this->leerExcel($archivo);
    return $result;
}

public function detectarDelimitador($ruta) {
    $delimitadores = [",", ";", "|", ":", "\t"];
    $primeraLinea = fgets(fopen($ruta, 'r'));

    $frecuencias = [];

    foreach ($delimitadores as $delimitador) {
        $frecuencias[$delimitador] = substr_count($primeraLinea, $delimitador);
    }

    return array_search(max($frecuencias), $frecuencias) ?: ',';
}
```

Figura 5-1. Lectura de archivos

5.1.2 Proceso de verificación e importación de pacientes

Uno de los aspectos más importantes del sistema es garantizar que no se dupliquen registros de pacientes al importar nuevas bases de datos. A la hora de añadir pacientes nuevos es obligatorio que esté en el archivo importado la columna de la fecha del trasplante y el nombre completo del paciente. En el caso de pacientes ya existentes es obligatoria la fecha y el nombre, pero puede ser tanto el nombre completo como sus iniciales.

Cuando un administrador importa un nuevo archivo, se le solicita que indique qué columnas corresponden al nombre del paciente y a la fecha. Por cada fila, el sistema realiza una serie de pasos que serán descritos a continuación.

5.1.2.1 Procesamiento del nombre

Se analiza si el contenido de la columna de nombre representa un nombre completo, lo cual se determina por la presencia de espacios. En ese caso, se generan automáticamente las iniciales del paciente. Para ello, se contempla que los nombres pueden venir en distintos formatos, como "Apellidos, Nombre" o "Nombre Apellidos", por lo que se hace un procesamiento previo para identificar el orden correcto y obtener las iniciales adecuadas. Este proceso se ilustra en la Figura 5-2.

Las iniciales se extraen tomando la primera letra de cada palabra significativa del nombre y los apellidos, ignorando palabras comunes como artículos o preposiciones (por ejemplo: *de*, *la*, *los*), pero teniendo en cuenta los casos de nombres y apellidos compuestos para asegurar que se reflejan correctamente en las iniciales. Así, un nombre como "López de las Heras, José Francisco" se transforma en las iniciales JFLH.

```php
private function generarIniciales($nombre) {
      $nombre = trim($nombre);

      if(strpos($nombre, ',') !== false) {
          [$apellidos, $nombres] = array_map('trim', explode(',', $nombre, 2));

          $inicialesNombre = $this->obtenerInicialesDePalabras($nombres);
          $inicialesApellido = $this->obtenerInicialesDePalabras($apellidos);

          return $inicialesNombre . $inicialesApellido;

      }

      return $this->obtenerInicialesDePalabras($nombre);
  }

  private function obtenerInicialesDePalabras($nombre) {
      $excluir = ['de', 'la', 'del', 'las', 'los', 'el', 'al', 'a', 'y', 'en'];
      $palabras = preg_split('/\s+/', trim($nombre));
      $iniciales = '';

      foreach ($palabras as $palabra) {
          if (strlen($palabra) > 0 && !in_array(strtolower($palabra), $excluir)) {
              $iniciales .= strtoupper($palabra[0]);
            }
        }

        return $iniciales;
  }
```

Figura 5-2. Proceso de generar iniciales

5.1.2.2 Procesamiento de la fecha

Para asegurar que las fechas ingresadas en distintos formatos sean interpretadas correctamente y uniformemente, el sistema implementa un proceso de normalización. Como se muestra en la Figura 5-3, primero, mediante una expresión regular, se detectan y sustituyen los nombres de los meses en español, ya sea en forma completa o abreviada, por su equivalente en inglés. Esto permite que la función encargada del análisis de fechas reconozca correctamente el mes independientemente del idioma. Por ejemplo, tanto "1-agosto-1998" como "1-ago-1998" se transforman en "1-Aug-1998".

Posteriormente, la fecha se evalúa contra múltiples formatos predefinidos que admiten variaciones comunes en la representación del día, mes y año. Por ejemplo, el día y el mes pueden estar expresados con uno o dos dígitos (como "1" o "01"), mientras que el año puede ser de dos o cuatro dígitos (por ejemplo, "98" o "1998"). Siguiendo con el ejemplo del primero de agosto de 1998, se permiten formatos como "1/8/98",

"01/08/1998", "1/Aug/1998", o "01/Aug/98. También acepta formatos que contemplan el orden año-mes-día, como 1998/8/01.

Además, el sistema acepta varios tipos de delimitadores entre los componentes de la fecha, tales como barras (/), guiones (-) o puntos (.), por lo que también se validan formatos como "1-8-1998", "01.08.98" o "1998.Aug.01".

```php
function normalizarFecha($fecha) {
        // Traducción de meses en español a inglés abreviado
        $mesesEspanol = [
            'ene' => 'Jan', 'feb' => 'Feb', 'mar' => 'Mar', 'abr' => 'Apr',
            'may' => 'May', 'jun' => 'Jun', 'jul' => 'Jul', 'ago' => 'Aug',
            'sep' => 'Sep', 'oct' => 'Oct', 'nov' => 'Nov', 'dic' => 'Dec'
        ];

        // Reemplazar meses en español por su equivalente en inglés
        $fecha = preg_replace_callback(
            '/\b(' . implode('|', array_map(fn($m) => $m . '\w*', array_keys($mesesEspanol))) . ')\b/i',
            function ($matches) use ($mesesEspanol) {
                $mesEsp = strtolower(substr($matches[1], 0, 3));
                return $mesesEspanol[$mesEsp] ?? $matches[1];
            },
            $fecha
        );

        // Lista base de formatos
        $formatosBase = [
            'd/m/Y', 'd/m/y',
            'd/M/Y', 'd/M/y',
            'd/n/Y', 'd/n/y',
            'j/n/Y', 'j/n/y',
            'j/m/Y', 'j/m/y',
            'j/M/Y', 'j/M/y',
            'Y/m/d', 'Y/n/j',
            'Y/m/j', 'Y/n/d',
            'Y/M/d', 'Y/M/j'
        ];

        // Expandir con todos los delimitadores
        $delimitadores = ['/', '-', '.'];
        $formatosFinales = [];
        foreach ($formatosBase as $formato) {
            foreach ($delimitadores as $del) {
                $formatosFinales[] = str_replace('/', $del, $formato);
            }
        }

        // Probar cada formato
        foreach ($formatosFinales as $formato) {
            $dt = DateTime::createFromFormat($formato, $fecha);
            if ($dt && $dt->format($formato) === $fecha) {
                return $dt->format('Y-m-d');
            }
        }

        return null;
    }
```

Figura 5-3. Proceso de normalizar fecha

En total, se admiten 90 formatos diferentes de fechas. Si la fecha proporcionada coincide con alguno de estos formatos y es válida, se convierte automáticamente a un formato estándar único en el sistema, YYYY-MM-DD (1998-08-01). En caso contrario, la fecha se considera inválida y se añade la fila correspondiente a esa fecha a un listado de errores.

5.1.2.3 Verificación en la base de datos

Se busca un paciente existente en la base de datos del hospital con esas mismas iniciales y la misma fecha de nacimiento. Si se encuentra, se actualiza la fila con los datos nuevos (si hay). Si no se encuentra coincidencia, se realiza una búsqueda secundaria por nombre exacto y fecha (por si acaso se hubiesen generado erróneamente las iniciales). Si aun así no se encuentra, se considera que se trata de un paciente nuevo.

5.1.2.4 Inserción o rechazo

Una vez se ha verificado que el paciente no existe, si el nombre era completo se procede a la inserción del registro del paciente en la base de datos. Si no es nombre completo no se puede añadir al paciente (ya que es uno de los requisitos) y la fila se añade a un listado de errores.

5.2 Módulo de consultas

Una funcionalidad esencial en este trabajo, es permitir a los usuarios la realización de consultas avanzadas para permitir un filtrado personalizado de la base de datos y su respectivo análisis. Para ello, se ha implementado una interfaz gráfica intuitiva que será explicada posteriormente, concretamente, en el manual de uso, Apéndice B - . Una vez aplicados todos los filtros, la aplicación realiza los siguientes pasos, descritos a continuación, para conseguir un correcto filtrado de los datos.

5.2.1 Creación de JSON

En primer lugar, el sistema analiza la estructura visual generada por el usuario en la interfaz. Esta estructura está compuesta por un grupo raíz que puede contener múltiples condiciones, así como subgrupos anidados dentro de él. Cada grupo (ya sea raíz o subgrupo) contiene un operador lógico seleccionado por el usuario, como "AND" u "OR", que define cómo se deben combinar sus condiciones internas y subgrupos.

Como se contempla en la Figura 5-4, el sistema comienza recorriendo este grupo raíz. Para ello, accede a los elementos DOM que representan condiciones y subgrupos. Por cada grupo, se extraen todas las condiciones definidas: el campo de la base de

datos seleccionado, el operador de comparación (como “=”, “!=”, “<”, “contains”, etc.), y el valor asociado (si aplica). Algunas condiciones no requieren valor, como “is null” o “is not null”, y eso también es contemplado durante el análisis.

Además de las condiciones, el sistema inspecciona si existen subgrupos dentro de cada grupo. Para cada subgrupo detectado, el proceso se repite recursivamente: se accede a su operador lógico, se leen sus condiciones y sus propios subgrupos internos, si los hubiera. Esta recursividad permite construir una representación completa del árbol de filtros definidos visualmente, sin importar el nivel de anidamiento.

```
function construirGrupoJSON(grupoDiv) {
  const operador = grupoDiv.querySelector(".grupo-operador").value;
  const condicionesDiv = grupoDiv.querySelector(".condiciones");
  const condiciones = [];

  condicionesDiv.querySelectorAll(".condicion").forEach(condDiv => {
    const campo = condDiv.querySelector(".campo").value;
    const operadorCond = condDiv.querySelector(".operador").value;
    const valorInput = condDiv.querySelector(".valor");
    const valor = valorInput ? valorInput.value.trim() : null;

    if (operadorCond === "is null" || operadorCond === "is not null") {
      condiciones.push({ campo, operador: operadorCond, valor: null });
    } else if (valor !== null && valor !== "") {
      condiciones.push({ campo, operador: operadorCond, valor });
    }
  });

  const subgruposDiv = grupoDiv.querySelector(".subgrupos");
  const subgrupos = [];
  subgruposDiv.querySelectorAll(":scope > .grupo").forEach(subgrupoDiv => {
    subgrupos.push(construirGrupoJSON(subgrupoDiv));
  });

  return {
    operador,
    condiciones,
    subgrupos
  };
}
```

Figura 5-4. Construcción de estructura de los filtros

Todo este recorrido da lugar a la construcción de una estructura de datos en forma de objeto anidado, donde cada grupo contiene su operador, una lista de condiciones y una lista de subgrupos, que a su vez tienen la misma estructura. Esta representación interna refleja fielmente la lógica de filtrado que el usuario definió visualmente.

Una vez finalizado el recorrido y construida esta estructura, se convierte a un formato JSON que se envía a través de una solicitud HTTP. Dicha solicitud la tratará el controlador correspondiente, que hace de intermediario con el repositorio de la tabla a consultar, y envía los resultados de vuelta al cliente, donde se muestran en una tabla HTML.

En la Figura 5-5 se muestra un ejemplo de la estructura JSON generada por la interfaz de filtrado si se introdujeran los siguientes criterios: la edad del receptor debe ser mayor que 30, la edad del donante debe contener el número 5, y, además, el receptor o el donante debe ser varón (considerando que el valor 1 representa al sexo masculino).

```
{
  "operador": "AND",
  "condiciones": [
    { "campo": "er", "operador": ">", "valor": "30" },
    { "campo": "ed", "operador": "contains", "valor": "5" }
  ],
  "subgrupos": [
    {
      "operador": "OR",
      "condiciones": [
        { "campo": "sr", "operador": "=", "valor": "1" },
        { "campo": "sd", "operador": "=", "valor": "1" }
      ],
      "subgrupos": []
    }
  ]
}
```

Figura 5-5. Ejemplo de JSON

5.2.2 JSON a consulta SQL

El objeto en formato JSON generado es enviado al servidor, donde comienza el proceso de convertirlo en una consulta SQL válida que se pueda ejecutar sobre la base de datos. El objetivo de este proceso es traducir la estructura lógica enviada por el usuario en una cláusula WHERE de SQL, que permita obtener exactamente los datos que cumplen con esos filtros. Este proceso se contempla en la Figura 5-6 y es explicado a continuación.

```php
private function procesarGrupo(array $grupo, array &$params): ?string
{
    $operador = strtoupper($grupo['operador'] ?? 'AND');
    $condicionesSQL = [];

    // Procesar condiciones del grupo
    foreach ($grupo['condiciones'] ?? [] as $condicion) {
        $campo = $condicion['campo'];
        $op = strtoupper($condicion['operador']);
        $valor = $condicion['valor'] ?? null;

        if (in_array($op, ['IS NULL', 'IS NOT NULL'])) {
            $condicionesSQL[] = "`$campo` $op";
        } elseif ($op === 'CONTAINS') {
            $condicionesSQL[] = "`$campo` LIKE ?";
            $params[] = "%$valor%";
        } else {
            $condicionesSQL[] = "`$campo` $op ?";
            $params[] = $valor;
        }
    }

    // Procesar subgrupos recursivamente
    foreach ($grupo['subgrupos'] ?? [] as $subgrupo) {
        $subSQL = $this->procesarGrupo($subgrupo, $params);
        if ($subSQL !== null && $subSQL !== '') {
            $condicionesSQL[] = "($subSQL)";
        }
    }

    if (empty($condicionesSQL)) {
        return null;
    }

    // Unir condiciones y subgrupos con el operador del grupo
    return implode(" $operador ", $condicionesSQL);
}
```

Figura 5-6. Proceso de condición 'where' para filtrado

El servidor comienza por leer el grupo principal (el grupo raíz). Dentro de este grupo puede haber varias condiciones y también otros subgrupos. Cada condición se traduce a una expresión SQL (por ejemplo, si el usuario seleccionó que el campo "edad" sea mayor que 30, se transforma en: edad > ?). Si el filtro es del tipo "contiene", se utiliza la cláusula LIKE, y si es "es nulo" o "no es nulo", se añade directamente esa condición sin necesidad de un valor.

Cada vez que se encuentra un valor, este no se coloca directamente en la consulta, sino que se guarda aparte en una lista de parámetros. De esta forma, se evita el riesgo de inyección de código y se asegura la ejecución segura de la consulta.

Luego, si el grupo contiene subgrupos, estos se procesan de forma recursiva, es decir, aplicando el mismo procedimiento que con el grupo principal. Se generan las condiciones correspondientes y se combinan usando el operador lógico definido para ese subgrupo. Este uso de recursividad permite construir consultas complejas, con varios niveles de condiciones anidadas, respetando la lógica definida por el usuario.

Finalmente, se prepara y ejecuta la consulta SQL utilizando los parámetros que se fueron guardando, y se obtiene un conjunto de datos que cumplen exactamente con los filtros definidos. Estos datos se envían de vuelta al cliente para ser mostrados en pantalla.

La Figura 5-7 muestra cómo se traduce la estructura JSON enviada desde la interfaz en una consulta SQL real. Utiliza el mismo ejemplo que el objeto JSON visto en la Figura 5-5. En la parte superior se observa la condición WHERE resultante, con el uso de placeholders (?) para evitar inyecciones SQL. Debajo, se especifica el conjunto de parámetros que se utilizarán para sustituir esos placeholders durante la ejecución de la consulta.

Figura 5-7. Ejemplo de condición 'where'

5.3 Módulo de estadísticas

Para implementar el módulo de estadísticas, se ha usado la librería Chart.js para generar gráficos dinámicos y visualmente atractivos basados en los datos obtenidos del

servidor. La información llega en forma de listas de registros con datos numéricos y categóricos que deben organizarse para poder mostrarlos correctamente en los gráficos.

Por ejemplo, para crear el gráfico de líneas que representa la evolución del número de pacientes por año y por hospital, primero es necesario identificar todos los años distintos que aparecen en los datos, así como todos los hospitales involucrados. Esto se hace recorriendo la lista completa y guardando cada año y cada hospital que aparece, asegurándonos de no repetirlos, con el fin de tener una lista única y ordenada para usar como etiquetas en el gráfico.

Una vez se tiene la lista de años y de hospitales, se prepara un conjunto de series de datos, una por cada hospital. Cada serie debe contener el número de pacientes para ese hospital en cada año. Para ello, se recorre la lista de años y, para cada año, se busca si existe un registro que coincida con ese hospital y ese año. Si se encuentra, se añade el número de pacientes de ese registro; si no, se añade un cero para indicar que no hubo pacientes ese año.

Con estos datos organizados, se configura el gráfico de líneas indicando que el eje horizontal representará los años y que cada línea corresponderá a un hospital diferente. Todo este proceso es el correspondiente a la Figura 5-8.

```
const años = [...new Set(porAnio.map(item => item.anio))];
const hospitales = [...new Set(porAnio.map(item => item.Nombre))];

const datasetsAnio = hospitales.map(hospital => ({
    label: hospital,
    data: años.map(anio => {
        const entry = porAnio.find(p => p.anio == anio && p.Nombre == hospital);
        return entry ? entry.total : 0;
    }),
    fill: false,
    borderColor: '#' + Math.floor(Math.random()*16777215).toString(16),
}));

new Chart(document.getElementById('graficoPacientesAnio'), {
    type: 'line',
    data: {
        labels: años,
        datasets: datasetsAnio
    },
    options: {
        responsive: true,
        plugins: { title: { display: true, text: 'Nuevos pacientes por año y hospital' } }
    }
});
```

Figura 5-8. Empleo de estadísticas

Capítulo 6 - Conclusiones y trabajo futuro

6.1 Conclusiones

El desarrollo del sistema ha dado lugar a una herramienta funcional, intuitiva y eficaz para la gestión y análisis de datos clínicos, pensada para profesionales del ámbito médico e investigador. La plataforma permite tanto la inserción estructurada de información como su consulta avanzada mediante filtros, con una interfaz clara y accesible que favorece el trabajo de usuarios con distintos perfiles.

La incorporación de gráficos interactivos y paneles visuales facilita la comprensión de los datos y contribuye a una toma de decisiones más informada. Además, el sistema fomenta la reutilización y análisis de información clínica actualizada, lo que supone una base sólida para futuras investigaciones médicas.

Gracias al enfoque flexible del modelo de datos y la estructura modular de la aplicación, se ha conseguido una solución adaptable, capaz de integrarse con diferentes entornos hospitalarios y evolucionar con nuevas funcionalidades.

6.2 Trabajo futuro

A partir de la base ya desarrollada, se plantean diversas líneas de mejora y ampliación del sistema, tanto en funcionalidades como en su integración con otros entornos sanitarios y tecnológicos. A continuación, se enumeran algunas propuestas concretas:

- Historial de consultas personalizadas: permitir a los usuarios guardar las consultas realizadas, de modo que puedan reutilizarlas posteriormente sin necesidad de reconfigurar los filtros.
- Generación de reportes: incorporación de una funcionalidad que permita generar reportes personalizados a partir de consultas específicas o estadísticas seleccionadas, con el objetivo de facilitar la documentación y el análisis de la información clínica. Estos reportes podrán configurarse como visibles o privados según las necesidades del usuario.

- Estadísticas personalizadas: módulo para la creación de estadísticas personalizadas, en el que cada usuario pueda definir sus propios parámetros de análisis y visualizar resultados adaptados a sus necesidades específicas.
- Formulario de alta de hospitales: permitir que cada hospital pueda darse de alta de forma autónoma en la aplicación, proporcionando datos como el correo electrónico de su administrador y otra información relevante.
- Mejoras de seguridad: medidas de seguridad adicionales para proteger mejor los datos sensibles. Entre ellas, se incluiría el cifrado de la información sensible almacenada en la base de datos. También se considera incorporar un sistema de autenticación multifactor para fortalecer el control de acceso.
- Integración con estándares sanitarios internacionales como HL7 o FHIR: Esto permitiría que la aplicación pueda interoperar con otros sistemas clínicos de manera más eficiente, facilitando el intercambio estructurado de datos médicos entre centros, instituciones o softwares de terceros que ya operan bajo estos protocolos reconocidos internacionalmente.
- Desarrollo de APIs para integración con sistemas hospitalarios existentes: Esta funcionalidad posibilitaría una conexión directa entre el sistema y los sistemas de gestión de datos clínicos que ya utilizan los hospitales, automatizando el traspaso de información y reduciendo errores derivados de la carga manual de datos.
- Implementación de algoritmos de machine learning para análisis predictivo: Mediante el entrenamiento con los datos clínicos almacenados, podrían desarrollarse modelos capaces de detectar patrones, predecir complicaciones o ayudar en la toma de decisiones clínicas, aportando un valor añadido al sistema desde el punto de vista médico y de investigación.

Conclusions and future work

Conclusions

The development of the system has resulted in a functional, intuitive, and effective tool for the management and analysis of clinical data, designed for professionals in the medical and research fields. The platform allows for both structured data entry and advanced querying through filters, with a clear and accessible interface that supports users with different profiles.

The incorporation of interactive charts and visual panels facilitates data interpretation and contributes to more informed decision-making. Additionally, the system promotes the reuse and analysis of up-to-date clinical information, providing a solid foundation for future medical research.

Thanks to the flexible data model and the modular structure of the application, a scalable solution has been achieved—capable of adapting to different hospital environments and evolving with new functionalities.

Future Work

Building on the current system, several areas for improvement and expansion are proposed, both in terms of new functionalities and integration with other healthcare and technological environments. Below are some specific suggestions:

- History of custom queries: Enable users to save their previous queries, so they can reuse them later without needing to reconfigure the filters.6
- Report generation: Introduce a feature that allows users to generate custom reports based on specific queries or selected statistics. These reports aim to support clinical data documentation and analysis and can be set as public or private depending on user needs.
- Custom statistics: Add a module for the creation of personalized statistics, where each user can define their own analysis parameters and visualize results tailored to their specific needs.

- Hospital self-registration form: Allow hospitals to register autonomously in the application by submitting information such as the administrator's email address and other relevant data.
- Security improvements: Additional security measures to better protect sensitive data. These include encrypting sensitive information stored in the database. Incorporating a multi-factor authentication system to strengthen access control is also being considered.
- Integration with international healthcare standards such as HL7 or FHIR: This would enable the application to interoperate more efficiently with other clinical systems by supporting structured exchange of medical data across centers, institutions, or third-party software that adhere to these widely recognized protocols.
- Development of APIs for integration with existing hospital systems: This functionality would allow direct connectivity between the platform and the hospital's current clinical data systems, automating data transfer and reducing errors caused by manual entry.
- Implementation of machine learning algorithms for predictive analysis: By training models with the clinical data stored in the system, it would be possible to detect patterns, predict complications, or assist in clinical decision-making, thereby adding analytical value from both a medical and research perspective.

BIBLIOGRAFÍA

[1] [En línea]. Available: https://project-redcap.org/.

[2] [En línea]. Available: https://www.openclinica.com/.

[3] [En línea]. Available: https://www.i2b2.org/.

[4] [En línea]. Available: https://code.visualstudio.com/.

[5] [En línea]. Available: https://git-scm.com/.

[6] [En línea]. Available: https://github.com/.

[7] [En línea]. Available: https://git-fork.com/.

[8] [En línea]. Available: https://www.slimframework.com/.

[9] [En línea]. Available: https://phpspreadsheet.readthedocs.io/en/latest/.

[10] [En línea]. Available: https://www.phpmyadmin.net/.

[11] [En línea]. Available: https://www.apachefriends.org/es/index.html.

[12] [En línea]. Available: https://boxicons.com/.

[13] [En línea]. Available: https://fontawesome.com/.

[14] [En línea]. Available: https://fonts.google.com/specimen/Roboto.

[15] [En línea]. Available: https://choices-js.github.io/Choices/.

[16] [En línea]. Available: https://www.chartjs.org/docs/latest/samples/bar/stacked-groups.html.

APÉNDICES

Apéndice A - Requisitos del sistema

En este anexo se detallan los actores que intervienen en el sistema, así como la especificación de requisitos funcionales.

Actores

- Administrador: Usuario con permisos para gestionar médicos e insertar archivos de base de datos.
- Médico: Usuario con permisos para consultar los datos sensibles de los pacientes asociados a su hospital.
- Investigador: Usuario con permisos limitados para consultar datos anonimizados.

Gestión de usuarios

REQUISITO	**REGISTRARSE**
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir a los usuarios crear una cuenta en la aplicación para acceder al sistema.
PRECONDICIÓN	NA
ENTRADA	Nombre, nombre de usuario, correo electrónico, contraseña, confirmación de contraseña
ACCIONES	1. Se muestra por pantalla un formulario con los campos. 2. El usuario rellena todos los campos y pulsa el botón de registrarse. 3. El sistema valida la información. 4. Se redirige a la pantalla de login en caso de éxito
SALIDA	Redirección a la pantalla de login

POSTCONDICIÓN	Se ha creado la cuenta y la sesión no está iniciada
EXCEPCIONES	1. Mensaje de error si la contraseña y confirmación de contraseña no coinciden. 2. Mensaje de error si el nombre de usuario no cumple con los requisitos. 3. Mensaje de error si el email tiene un formato inválido. 4. Mensaje de error si el email o el nombre de usuario ya están en uso.
ACTORES	Usuario no registrado

Tabla A-1. Registro

REQUISITO	**INICIAR SESIÓN**
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir a los usuarios registrados acceder al sistema mediante sus credenciales.
PRECONDICIÓN	El usuario está registrado.
ENTRADA	Nombre de usuario, contraseña
ACCIONES	1. Se muestra por pantalla un formulario con los campos. 2. El usuario rellena todos los campos y pulsa el botón de iniciar sesión. 3. El sistema valida la información. 4. Se redirige a la pantalla principal en caso de éxito.
SALIDA	Redirección a la pantalla principal
POSTCONDICIÓN	La sesión está iniciada
EXCEPCIONES	1. Mensaje de error si el usuario no existe en el sistema. 2. Mensaje de error si la contraseña no es correcta.
ACTORES	Usuario registrado (Todos)

Tabla A-2. Inicio de sesión

REQUISITO	CERRAR SESIÓN
PRIORIDAD	Media
DESCRIPCIÓN	Permitir al usuario salir de su sesión activa.
PRECONDICIÓN	El usuario debe haber iniciado sesión previamente
ENTRADA	NA
ACCIONES	1. El usuario pulsa el botón de cerrar sesión. 2. El sistema finaliza la sesión activa del usuario. 3. Se redirige a la pantalla de login.
SALIDA	Redirección a la pantalla de login
POSTCONDICIÓN	Se ha cerrado la sesión
ACTORES	Usuario registrado (Todos)

Tabla A-3. Logout

REQUISITO	EDITAR PERFIL
PRIORIDAD	Baja
DESCRIPCIÓN	Permitir al usuario modificar sus datos personales y de acceso.
PRECONDICIÓN	El usuario debe tener su sesión iniciada.
ENTRADA	Datos del usuario a modificar (nombre, correo, contraseña, etc.)
ACCIONES	1. El usuario accede a la sección "Editar perfil". 2. Se muestra un formulario con los datos actuales. 3. El usuario modifica los campos deseados y confirma los cambios. 4. El sistema valida la información. 5. Se actualizan los datos.
SALIDA	Redirección a la pantalla principal
POSTCONDICIÓN	Se actualizan los valores en la base de datos

EXCEPCIONES	1. Mensaje de error si el nombre de usuario no cumple con los requisitos. 2. Mensaje de error si el email tiene un formato inválido. 3. Mensaje de error si el email o el nombre de usuario ya están en uso 4. Mensaje de error si la contraseña actual no es correcta. 5. Mensaje de error si la contraseña nueva y la confirmación no coinciden.
ACTORES	Usuario registrado (Todos)

Tabla A-4. Editar perfil

REQUISITO	**DARSE DE BAJA**
PRIORIDAD	Media
DESCRIPCIÓN	Permitir al usuario eliminar su cuenta y todos sus datos personales asociados.
PRECONDICIÓN	El usuario debe tener su sesión iniciada.
ENTRADA	Contraseña
ACCIONES	1. Se muestra un formulario para volver a introducir la contraseña. 2. El usuario ingresa su contraseña y pulsa el botón de eliminar . 3. El sistema valida la información.
SALIDA	Redirección a la pantalla de login
POSTCONDICIÓN	El usuario ya no existe en el sistema y sesión cerrada
ACTORES	Usuario registrado (Todos)

Tabla A-5. Dar de baja

REQUISITO	AÑADIR PERMISO A MÉDICO
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir al administrador de un hospital añadir a un médico para que tenga los permisos correspondientes.
PRECONDICIÓN	El usuario debe tener su sesión iniciada.
ENTRADA	Correo electrónico
ACCIONES	1. El sistema muestra un modal para añadir el correo. 2. El usuario ingresa el correo y pulsa el botón de eliminar. 3. El sistema valida el correo. 4. El sistema le asigna el rol de médico a dicho correo.
SALIDA	Se muestra por pantalla el correo añadido en la lista de correos de los médicos del hospital
POSTCONDICIÓN	Se ha añadido al sistema un usuario con el correo y rol de médico si no se ha registrado previamente. En caso contrario se ha actualizado el usuario y se le asigna el rol de médico.
EXCEPCIONES	1. Mensaje de error si el correo tiene un formato inválido. 2. Mensaje de error si el correo está asociado a otro hospital
ACTORES	Usuario registrado (Administrador)

Tabla A-6. Añadir permiso a médico

REQUISITO	ELIMINAR PERMISO A MÉDICO
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir al administrador de un hospital revocar los permisos de un médico, eliminando su rol correspondiente.
PRECONDICIÓN	El usuario debe tener la sesión iniciada y debe existir al menos un usuario con rol de médico en el hospital correspondiente.
ENTRADA	Seleccionar el botón 'x' junto al correo electrónico del médico.
ACCIONES	1. El sistema muestra un modal con la lista de correos de los médicos del hospital, cada uno con un botón 'x' al lado.

	2. El usuario pulsa la 'x' del correo que desea eliminar. 3. El sistema actualiza el rol del usuario asociado a ese correo.
SALIDA	El correo electrónico seleccionado desaparece de la lista de médicos con permisos
POSTCONDICIÓN	El registro del usuario ha sido actualizado y ahora tiene el rol de investigador
ACTORES	Usuario registrado (Administrador)

Tabla A-7. Eliminar permiso a médico

Gestión de la base de datos

REQUISITO	**IMPORTACIÓN**
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir al usuario subir archivos para su procesamiento.
PRECONDICIÓN	El usuario debe tener su sesión iniciada y tener permiso para importar archivos.
ENTRADA	Archivo .xls, .xlsx, .csv o .txt.
ACCIONES	1. El usuario selecciona un archivo con formato válido. 2. El sistema valida la extensión del archivo para evitar incompatibilidades. 3. El sistema Procesa la lectura de las columnas del archivo para el siguiente paso. 4. Se redirige a la pantalla de mapear las columnas.
SALIDA	Redirección a la pantalla de mapeo
POSTCONDICIÓN	El archivo se ha importado correctamente y se ha guardado en una ruta temporal para su posterior procesamiento.
ACTORES	Usuario registrado (Administrador)

Tabla A-8. Importación de archivos

REQUISITO	MAPEO DE COLUMNAS
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir al usuario asociar columnas del archivo importado con campos de la base de datos.
PRECONDICIÓN	Archivo importado correctamente y datos disponibles.
ENTRADA	Selección manual de la columna nombre, columna fecha y de correspondencia entre columnas del archivo y columnas de la base de datos.
ACCIONES	1. El sistema muestra un formulario para columnas obligatorias y una sección con las demás columnas y las columnas detectadas del archivo. 2. El usuario selecciona en el formulario la columna nombre y la columna fecha de su archivo. 3. El usuario selecciona para cada columna del archivo la columna destino en la base de datos y pulsa el botón de seleccionar. 4. El usuario confirma el mapeo para continuar. 5. El sistema procesa las selecciones. 6. Se redirige a la pantalla de agregar columnas si han quedado columnas sin mapear.
SALIDA	Redirección a la pantalla de agregar columnas.
POSTCONDICIÓN	Se agregan los datos de las columnas del archivo a la columna de la base de datos correspondiente.
EXCEPCIONES	Mensaje de error si no se asignan columnas obligatorias (nombre y fecha)
ACTORES	Usuario registrado (Administrador)

Tabla A-9. Mapeo de columnas

REQUISITO	AGREGAR COLUMNAS NUEVAS
PRIORIDAD	Media
DESCRIPCIÓN	Permitir al usuario crear nuevas columnas en la base de datos para columnas no mapeadas del archivo.

PRECONDICIÓN	Columnas del archivo sin mapear
ENTRADA	Descripción y tipo de la nueva columna.
ACCIONES	1. El sistema detecta columnas del archivo sin correspondencia en la base de datos. 2. Se ofrece al usuario la opción de crear nuevas columnas con un formulario con campos a rellenar para cada columna detectada. 3. El usuario especifica descripción y tipo de dato. 4. El usuario confirma la agregación. 5. El sistema crea las nuevas columnas.
SALIDA	Redirección a la pantalla principal.
POSTCONDICIÓN	La base de datos tiene las nuevas columnas.
ACTORES	Usuario registrado (Administrador)

Tabla A-10. Inserción de columnas

Explotación de la base de datos

REQUISITO	**REALIZAR CONSULTAS**
PRIORIDAD	Alta
DESCRIPCIÓN	Permitir al usuario realizar consultas filtradas sobre los datos almacenados en el sistema.
PRECONDICIÓN	El usuario debe tener su sesión iniciada
ENTRADA	Criterios de búsqueda
ACCIONES	1. El sistema muestra opciones para definir filtros. 2. El usuario introduce o selecciona los parámetros deseados. 3. El sistema ejecuta la consulta sobre la base de datos. 4. Se muestran los resultados en pantalla.
SALIDA	Filas de la base de datos que cumplen los criterios

POSTCONDICIÓN	NA
ACTORES	Usuario registrado (Todos)

Tabla A-11. Realizar consultas

REQUISITO	**VER ESTADÍSTICAS**
PRIORIDAD	Media
DESCRIPCIÓN	Mostrar al usuario gráficos y métricas que resumen la información clave almacenada en el sistema.
PRECONDICIÓN	El usuario debe tener su sesión iniciada
ENTRADA	NA
ACCIONES	1. El sistema recupera los datos relevantes desde la base de datos. 2. Los datos se procesan y transforman en formatos adecuados para graficar. 3. El sistema muestra los gráficos junto con indicadores numéricos clave. 4. El usuario puede interactuar con los gráficos.
SALIDA	Visualización dinámica e interactiva de estadísticas
POSTCONDICIÓN	NA
ACTORES	Usuario registrado (Todos)

Tabla A-12. Ver estadísticas

Apéndice B - Manual de uso

Este manual tiene como objetivo guiar al usuario en el uso básico de la aplicación desarrollada. A lo largo de este documento se describen las funcionalidades principales del sistema, los distintos roles de usuario y los pasos necesarios para realizar tareas comunes como registrar usuarios, cargar datos clínicos o consultar información.

El sistema está diseñado para ser utilizado por tres tipos de perfiles: administrador, médico e investigador.

Inicio de sesión y registro

Nada más desplegar la aplicación, la primera vista que se muestra al usuario es la pantalla de inicio de sesión, tal como se observa en la Figura B-1. Para facilitar la evaluación del sistema, se ha preconfigurado un usuario con rol de administrador en la base de datos, cuyas credenciales son:

- Usuario: admin
- Contraseña: adminpass

En este caso práctico, se utilizará dicho usuario para mostrar las funcionalidades del sistema desde la perspectiva del administrador, ya que este rol tiene acceso completo a todas las características disponibles.

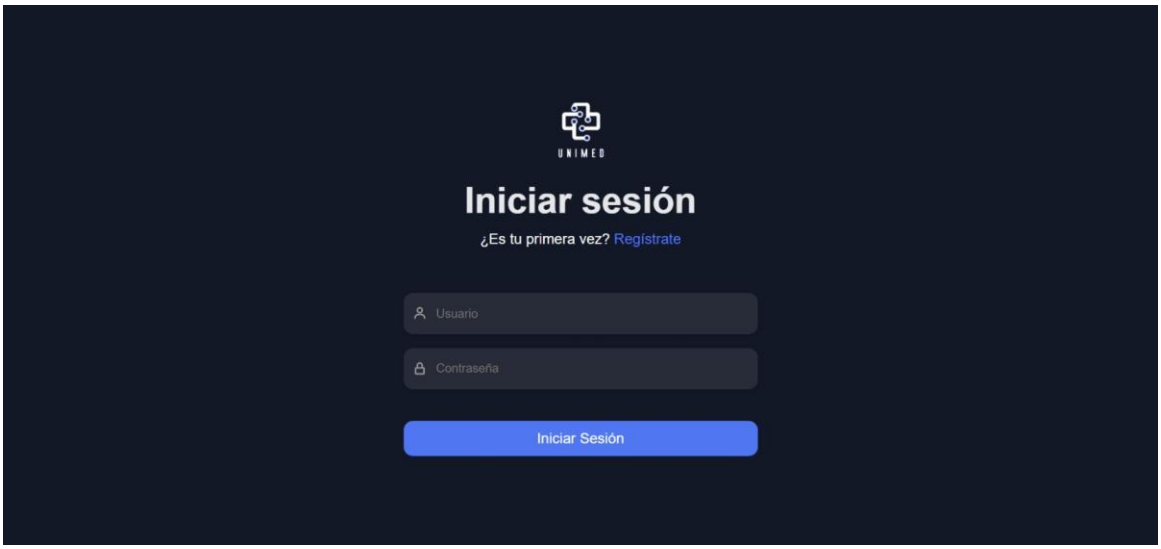


Figura B-1. Inicio de sesión

Desde la vista de inicio de sesión, se puede acceder al formulario de registro pulsando en el enlace azul "Regístrate", también visible en la a la Figura B-1. Esto redirige a la pantalla correspondiente, mostrada en la Figura B-2 .

El formulario de registro solicita los siguientes campos: nombre, nombre de usuario, correo electrónico, contraseña y confirmación de contraseña. El sistema valida automáticamente los datos introducidos y muestra mensajes de error en caso de que no se cumplan las siguientes condiciones:

- El nombre no puede contener números ni símbolos.
- El nombre de usuario debe tener entre 5 y 20 caracteres y no estar ya en uso.
- El correo electrónico debe tener un formato válido y no estar registrado previamente.
- Las contraseñas deben coincidir.

En relación con el control de acceso, el sistema distingue entre tres roles: administrador, médico e investigador. El rol se asigna automáticamente en función del correo electrónico proporcionado durante el registro. Los administradores pueden gestionar los correos electrónicos de los médicos autorizados de su hospital. Si un usuario se registra utilizando uno de estos correos validados, se le asigna automáticamente el rol de médico. En caso contrario, el sistema lo registra con el rol de investigador.

Es posible que un usuario se registre inicialmente como investigador y, tras recibir la autorización del administrador, su rol sea actualizado a médico posteriormente.

Una vez un usuario se registra con éxito, el sistema lo redirige automáticamente a la vista de inicio de sesión, donde deberá introducir las credenciales recién creadas para acceder a la aplicación.

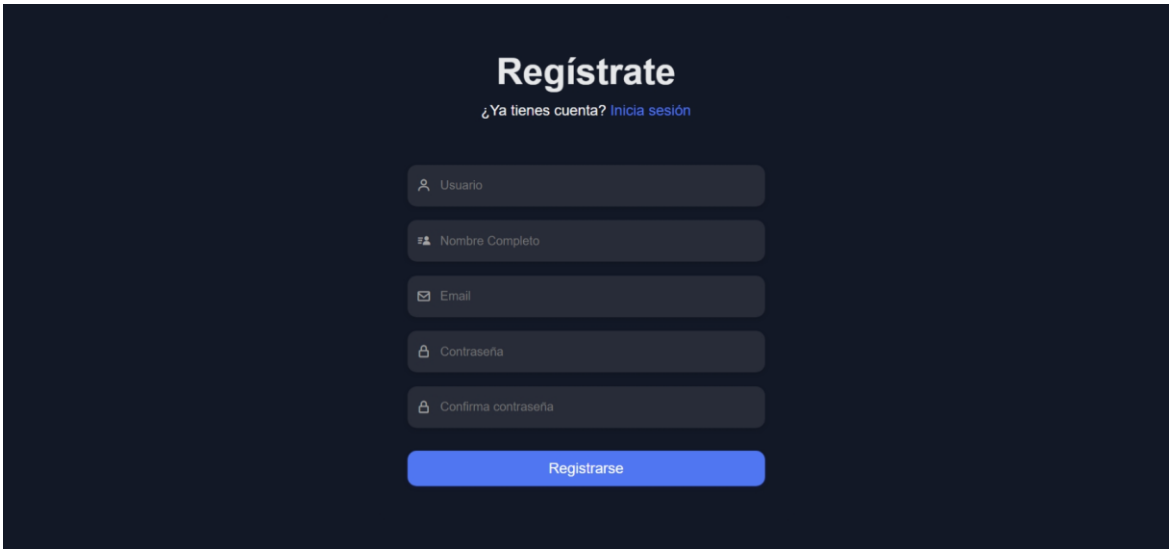


Figura B-2. Registro

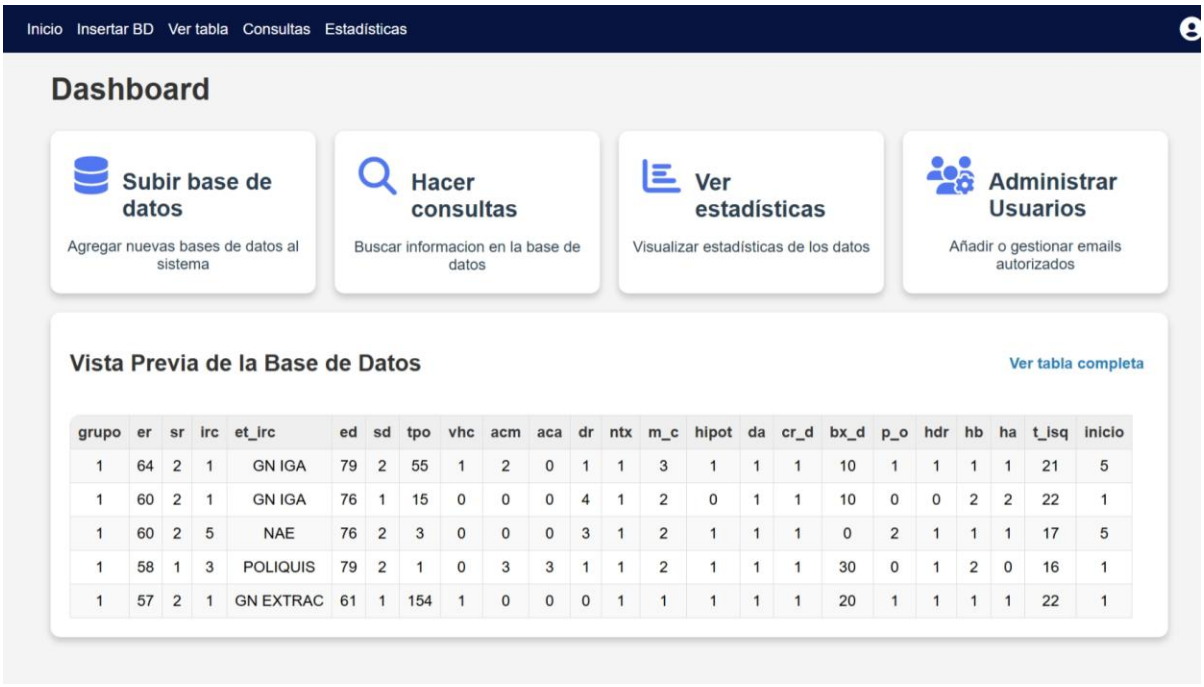
Dashboard y navegación principal

Tras iniciar sesión correctamente, el usuario es dirigido al Dashboard, la pantalla principal desde la cual se puede acceder al resto de funcionalidades del sistema (Figura B-3). Este panel de control se organiza en distintas secciones:

- Subir base de datos: disponible únicamente para administradores. Permite iniciar el flujo de carga de un archivo y realizar su correspondiente mapeo de columnas.
- Hacer consultas: da acceso a la vista de filtrado dinámico, donde es posible construir consultas personalizadas sobre los datos clínicos.
- Ver estadísticas: muestra la sección con estadísticas generales del sistema, representadas mediante gráficos interactivos.
- Administrar usuarios: también restringida a administradores. Al acceder, se abre un modal desde el cual es posible gestionar los médicos asociados a su hospital.

- Vista previa de la base de datos: permite visualizar una tabla resumida de los datos integrados, junto con un botón que da acceso a la tabla completa.

Además, la aplicación cuenta con una barra de navegación superior fija, visible en todas las vistas, que facilita el acceso directo a las funcionalidades principales sin necesidad de regresar al dashboard. En la esquina superior derecha, se encuentra un icono de perfil desde el cual se despliega un menú con opciones como cerrar sesión, editar perfil y darse de baja.



grupo	er	sr	irc	et_irc	ed	sd	tpo	vhc	acm	aca	dr	ntx	m_c	hipot	da	cr_d	bx_d	p_o	hdr	hb	ha	t_isq	inicio
1	64	2	1	GN IGA	79	2	55	1	2	0	1	1	3	1	1	1	10	1	1	1	1	21	5
1	60	2	1	GN IGA	76	1	15	0	0	0	4	1	2	0	1	1	10	0	0	2	2	22	1
1	60	2	5	NAE	76	2	3	0	0	0	3	1	2	1	1	1	0	2	1	1	1	17	5
1	58	1	3	POLIQUIS	79	2	1	0	3	3	1	1	2	1	1	1	30	0	1	2	0	16	1
1	57	2	1	GN EXTRAC	61	1	154	1	0	0	0	1	1	1	1	1	20	1	1	1	1	22	1

Figura B-3. Dashboard

Gestión de usuarios

La plataforma incorpora una funcionalidad específica para que los administradores puedan gestionar los usuarios médicos asociados a su hospital. Esta opción está disponible únicamente para cuentas con rol de administrador y se encuentra accesible desde el Dashboard, mediante el botón "Administrar usuarios".

Al hacer clic en esta opción, se abre un modal emergente que muestra una lista con todos los médicos actualmente habilitados para acceder a la aplicación dentro del mismo hospital del administrador. Esta interfaz permite tanto visualizar como modificar de forma rápida y sencilla los permisos de acceso.

En la parte superior del modal, se encuentra un campo de entrada de texto donde el administrador puede introducir una dirección de correo electrónico para añadir nuevos médicos al sistema o actualizar registros existentes. La aplicación alerta de lo siguiente:

- Formato del email: Solo se aceptan direcciones que cumplan con un formato válido.
- Conflictos de hospital: Si el correo ingresado ya está registrado en el sistema, pero asociado a un hospital diferente, se mostrará una advertencia informando que no es posible asignarlo a este hospital.

Junto a cada médico listado en la tabla del modal aparece un botón de eliminar, que permite revocar el acceso de ese usuario revocar su rol como médico. Al hacerlo, el usuario no pierde completamente el acceso al sistema, sino que pasa a tener permisos de investigador, y pierde el acceso a datos sensibles del hospital en el que estaba asignado.

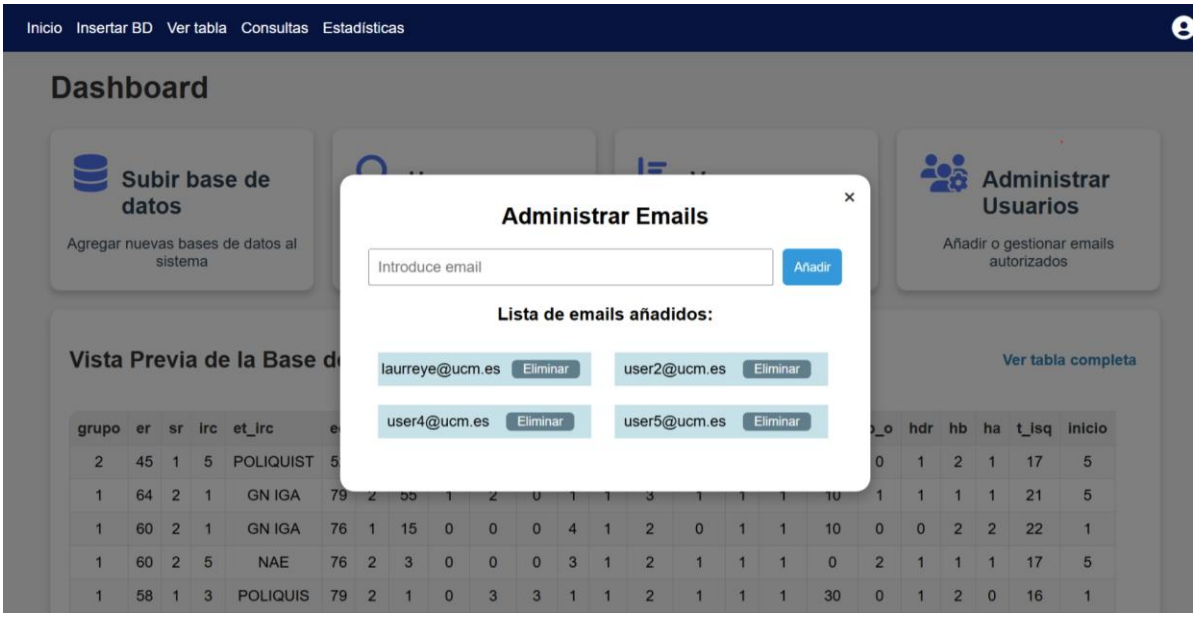


Figura B-4. Administrar usuarios

Flujo de incorporación de datos

El flujo de incorporación de nuevas bases de datos en el sistema está diseñado en tres etapas consecutivas, cada una representada por una vista claramente diferenciada. Este enfoque guiado facilita la tarea a usuarios sin conocimientos técnicos, asegurando que la información sea correctamente cargada, interpretada y normalizada dentro del sistema.

Vista de carga de archivo

Esta es la primera vista del flujo y permite al usuario seleccionar el archivo que desea incorporar. La interfaz combina un selector tradicional de archivos con una funcionalidad de arrastrar y soltar, mejorando la experiencia de uso en distintos dispositivos. Solo se aceptan formatos compatibles como .xls, .xlsx, .csv y .txt.

Visualmente, la vista presenta una barra superior con los pasos del proceso resaltados, indicando que se está en el Paso 1: Cargar Archivo. El botón de carga permanece deshabilitado hasta que se haya seleccionado un archivo válido, lo que previene errores comunes y fomenta una interacción intuitiva. La vista recién descrita se puede ver a continuación, en la Figura B-5.

Figura B-5. Vista de carga de archivo

Vista de mapeo de campos

Una vez subido el archivo, el sistema redirige a la vista de mapeo de columnas, en la cual el usuario debe asociar las columnas del archivo con los campos existentes en la base de datos del sistema. Esta vista se divide en tres secciones principales.

La primera sección es un formulario sencillo para que el usuario determine las columnas obligatorias del nombre y de la fecha. Seguidamente se encuentran dos paneles comparativos que muestran, respectivamente, las columnas detectadas de la base de datos y las columnas del archivo cargado. Por último, una sección inferior donde se visualizan las columnas ya mapeadas, permitiendo revisar y corregir emparejamientos fácilmente. En la Figura B-6 se muestra esta vista.

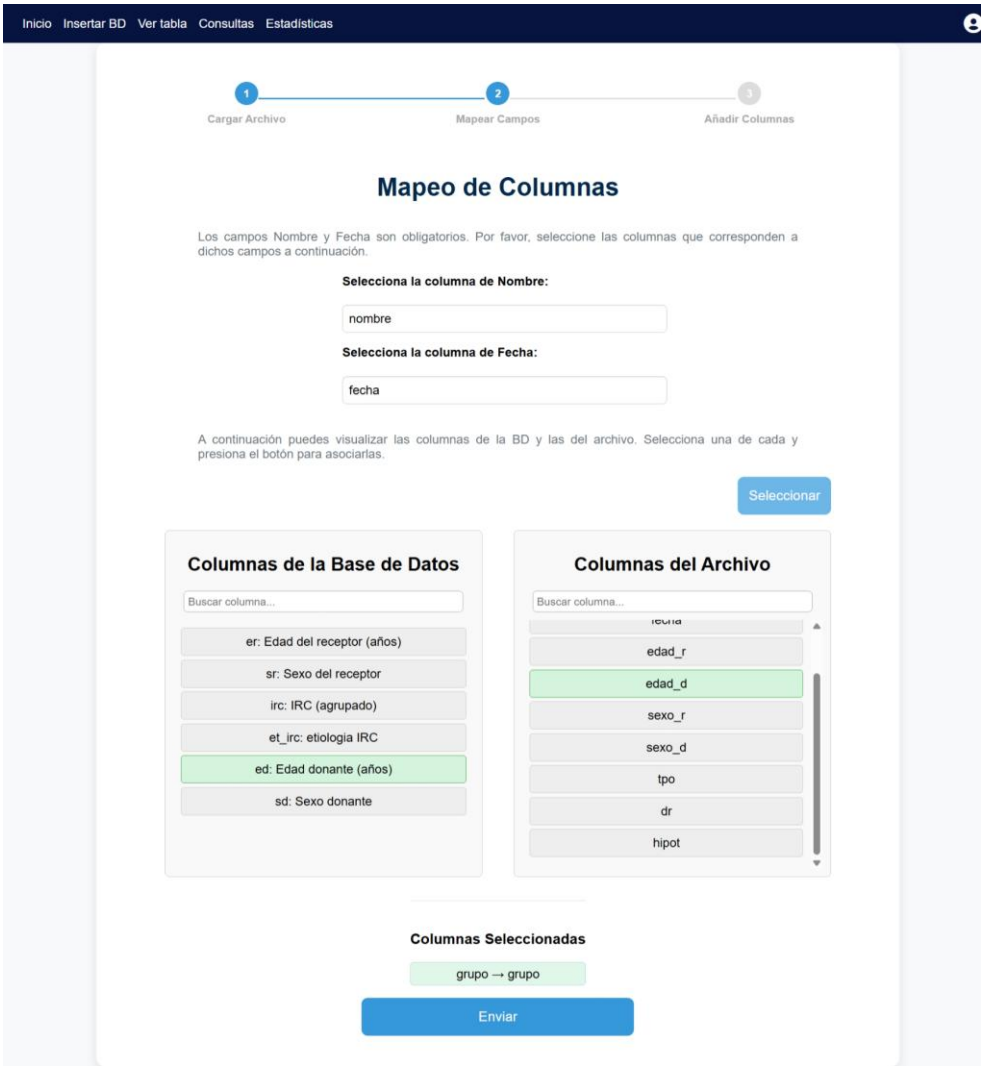


Figura B-6. Vista de mapeo de columnas

El usuario solo puede seleccionar una opción por cada lista de columnas. La vista incorpora, para cada panel, un cuadro de búsqueda que facilita el emparejamiento de columnas. En el caso de las columnas de la base de datos, la búsqueda se filtra tanto por el nombre como por la descripción, la cual también se muestra en pantalla junto al nombre correspondiente. Esto permite al usuario comprender mejor el contenido de cada columna existente. Esta funcionalidad es especialmente necesaria en bases de datos médicas, donde es común que los nombres de las columnas estén abreviados y no siempre resulten comprensibles para los profesionales sanitarios a simple vista.

Vista de añadir columnas nuevas

Una vez ha acabado el proceso de emparejamiento de columnas, se accede a la vista de añadir columnas pendientes. Si quedan columnas del archivo sin asociar a ningún campo del sistema , estas aparecerán en esta vista acompañadas de un cuadro de texto para la descripción de la columna y un desplegable para el tipo de dato. Aquí, el usuario puede optar por conservar estas columnas, proporcionando una descripción textual para facilitar su interpretación futura, y seleccionar el tipo de dato correspondiente (INT, VARCHAR, FLOAT, DATE).

Figura B-7. Vista de inserción de nuevas columnas

Consultas

La vista de consultas permite a los usuarios construir filtros complejos para buscar pacientes de manera flexible e intuitiva. A través de una interfaz visual, se ofrece un grupo principal, llamado grupo raíz, que siempre está presente en pantalla y sobre el cual se pueden añadir condiciones o crear subgrupos adicionales que permiten anidar lógica de filtrado utilizando operadores como AND u OR. Esta jerarquía facilita representar estructuras lógicas complejas sin que el usuario tenga que escribir código o conocer SQL. Además arriba del todo aparece un cuadro de selección para poder seleccionar que columnas quieres que se muestren.

Cada condición dentro de un grupo permite seleccionar una columna de la base de datos, un operador de comparación y un valor de referencia. El campo para introducir el valor se adapta automáticamente según el tipo de dato de la columna seleccionada, ya sea texto, número o fecha. Esto ayuda a prevenir errores al momento de definir filtros y hace que el sistema sea más comprensible para usuarios no técnicos.

El panel de control incluye tres botones principales: uno para exportar la tabla, otro para aplicar los filtros definidos y el último para limpiar la interfaz, eliminando todas las condiciones y subgrupos, dejando únicamente una condición vacía inicial para comenzar de nuevo. Al aplicar los filtros, se construye un objeto JSON con toda la lógica definida en la interfaz, que se envía al servidor mediante una petición POST, como se explicó en el 5.2. El resultado de la consulta se muestra en una tabla generada dinámicamente, donde se reflejan los datos de los pacientes que cumplen los criterios definidos. Si no se encuentran coincidencias, se informa al usuario de forma clara. La Figura B-8 muestra esta vista con el mismo ejemplo que se utilizó anteriormente en el 5.2, para explicar la lógica interna.

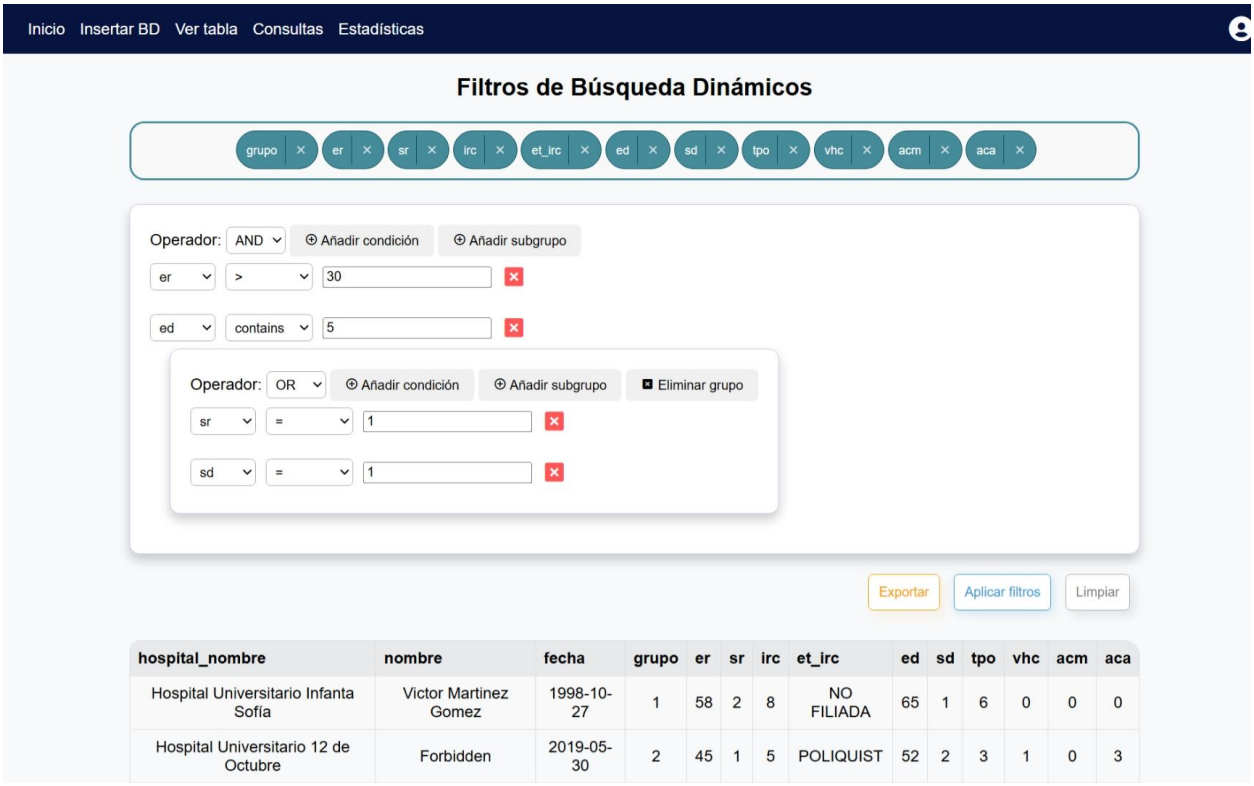


Figura B-8. Vista de consultas avanzadas

Estadísticas

La sección de estadísticas generales proporciona una visión clara y comprensible del estado del sistema a través de representaciones gráficas interactivas. Su objetivo es facilitar la interpretación de los datos clave, permitiendo al usuario comprender, de un solo vistazo, la actividad y evolución de la aplicación.

Al acceder a esta vista, se presentan distintos gráficos dispuestos de forma organizada. A la izquierda, un gráfico de pastel muestra de manera proporcional la distribución de pacientes por hospital. Al centro, un gráfico de barras apiladas resume la cantidad de trasplantes realizados por mes y hospital, lo que permite observar la actividad mensual acumulada y su variabilidad entre centros. A la derecha, un gráfico de radar representa la proporción de usuarios según su rol dentro del sistema (administradores, médicos e investigadores), ofreciendo una visión clara del perfil operativo de quienes acceden a la plataforma.

La parte inferior de la vista contiene un gráfico de líneas que representa la evolución anual de nuevos pacientes, diferenciados por hospital, lo que aporta una perspectiva más histórica. Este gráfico está acompañado de un conjunto de cuatro indicadores clave, organizados en un formato de cuadrícula. Allí se destacan cifras absolutas como el total de pacientes registrados, el número de hospitales activos, la cantidad total de usuarios y el año con mayor número de trasplantes realizados.

La interacción con estos elementos es dinámica y responsiva. Los colores se asignan de manera aleatoria para facilitar la distinción visual de cada categoría, y los gráficos se actualizan automáticamente en base a los datos en tiempo real que el sistema tenga almacenados.

Los gráficos son interactivos: al posicionarte sobre ellos podrás ver los valores exactos. Además, te permite hacer clic en las leyendas para ocultar o mostrar elementos como hospitales o tipos de usuarios. Esta vista se representa en la Figura B-9.

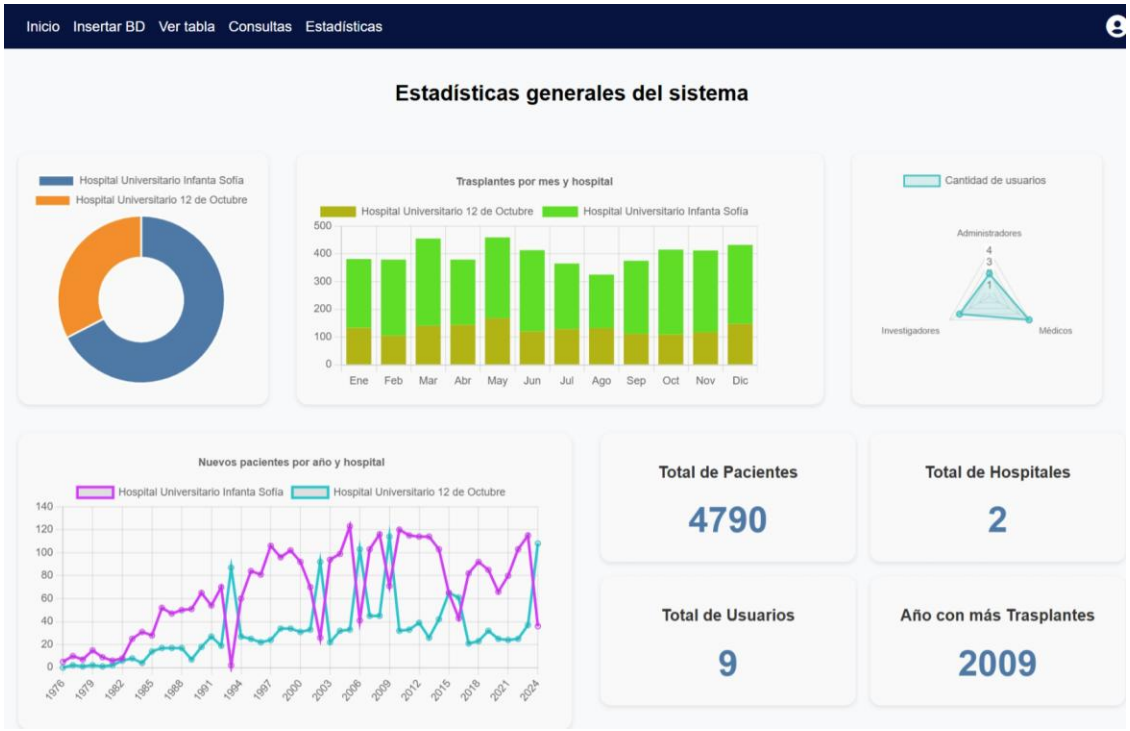


Figura B-9. Vista de estadísticas