UAX FP
Universidad
Alfonso X el Sabio

Cuadro de Mando Interactivo para una Clínica Dental

2º Administración de Sistemas Informáticos en Red Presencial

Conesa de Quadros, Guillermo

Tutor del TFG

UAX FP
Universidad
Alfonso X el Sabio

DEDICATORIA (OPCIONAL)

ÍNDICES

ABSTRACT

Este documento presenta el desarrollo de un Cuadro de Mando Interactivo para la gestión de la clínica dental ConeDental. El sistema está basado en una arquitectura cliente-servidor, con un backend en Flask y MySQL que proporciona una API para gestionar clínicas, pacientes, doctores, citas, materiales y otros datos clave. El frontend, desarrollado con HTML, CSS y JavaScript, ofrece una interfaz visual con gráficos dinámicos, tablas interactivas y alertas automáticas para optimizar la toma de decisiones.

El sistema incluye un mecanismo de recopilación automática de datos y notificaciones inteligentes sobre citas, stock de materiales y disponibilidad del personal. Su diseño escalable y personalizable permite adaptarlo a diferentes necesidades de la gestión clínica.

This document presents the development of an Interactive Dashboard for managing the ConeDental clinic. The system is based on a client-server architecture, with a Flask and MySQL backend providing an API to manage clinics, patients, doctors, appointments, materials, and other key data. The frontend, built with HTML, CSS, and JavaScript, offers a visual interface with dynamic charts, interactive tables, and automated alerts to enhance decision-making.

The system includes an automated data collection mechanism and smart notifications for appointments, material stock, and staff availability. Its scalable and customizable design allows adaptation to various needs in clinic management in the dental sector

JUSTIFICACIÓN DEL PROYECTO

En el ámbito odontológico, una gestión eficiente de una clínica es fundamental para garantizar un servicio de calidad a los pacientes y optimizar los recursos disponibles. Sin embargo, muchas clínicas aún dependen de métodos tradicionales, como registros en papel o sistemas informáticos desactualizados, lo que dificulta la organización de citas, el control de stock de materiales y la supervisión del personal. La motivación principal de este proyecto es desarrollar un Cuadro de Mando Interactivo que permita centralizar y visualizar en tiempo real toda la información relevante de una clínica dental, facilitando la toma de decisiones basada en datos.

Actualmente, existen soluciones comerciales para la gestión de clínicas dentales, que ofrecen funciones similares como Grafana. No obstante, estas aplicaciones suelen ser costosas, poco personalizables y requieren licencias de uso, lo que puede limitar su accesibilidad para pequeñas y medianas clínicas. A diferencia de estas herramientas, el Cuadro de Mando Interactivo de ConeDental está diseñado para ser una solución escalable y adaptable, permitiendo una integración completa con la base de datos de la clínica y personalización según sus necesidades específicas.

El sistema está dirigido a clínicas dentales y profesionales de la odontología que buscan mejorar la gestión de su práctica a través de una plataforma moderna y accesible. Al ser un sistema interactivo con alertas, facilita la identificación de problemas en la operativa diaria, como la falta de materiales o la sobrecarga de citas, contribuyendo a una administración más eficiente. En comparación con otras soluciones existentes, este proyecto destaca por su enfoque en la automatización, visualización intuitiva y accesibilidad sin costos de licencia, lo que lo convierte en una alternativa viable y competitiva para clínicas en crecimiento.

INTRODUCCIÓN

El Cuadro de Mando Interactivo de ConeDental optimiza la gestión clínica con un sistema centralizado y automatizado. Entre sus principales funciones destacan:

Gestión de Clinicas, Pacientes, Doctores, etc: Permite administrar datos de manera estructurada y accesible.

Control de Citas: Visualización y organización eficiente del calendario de consultas.

Supervisión de Inventario: Monitorea materiales dentales y medicamentos con alertas de stock bajo.

Visualización de Datos en Tiempo Real: Usa gráficos dinámicos y tablas interactivas para facilitar la toma de decisiones.

Automatización de Alertas: Notificaciones sobre eventos críticos como citas próximas o escasez de insumos.

Para un funcionamiento óptimo, el sistema debe cumplir con los siguientes requisitos:

Base de Datos MySQL con estructuras optimizadas para gestionar clínicas, citas y materiales.

Backend en Flask (Python) con una API RESTful para procesar y servir datos.

Frontend en HTML, CSS y JavaScript, con fetch() para actualizar información en tiempo real.

Gráficos en Chart.js para representación visual de estadísticas clave.

Sistema de Alertas Automáticas para notificar sobre eventos relevantes.

OBJETIVOS

R01 – El sistema debe ejecutarse en Ubuntu

- R01F01 – El servidor debe estar instalado y configurado en Ubuntu.
    - R01F01T01 – Instalar Ubuntu Server y configurar dependencias necesarias (Python, MySQL, Flask, etc.).
    - R01F01T01P01 – Verificar la instalación ejecutando python3 --version y mysql --version.
- R01F02 – El sistema debe iniciar automáticamente al arrancar Ubuntu.
    - R01F02T01 – Configurar systemd para ejecutar Flask como un servicio.
    - R01F02T01P01 – Reiniciar el servidor y comprobar que el servicio está activo con systemctl status flaskapp.

R02 – La base de datos debe estar implementada en MySQL

- R02F01 – La base de datos debe contener las tablas necesarias para la gestión de la clínica.
    - R02F01T01 – Crear las tablas clinicas, pacientes, doctores, citas, materiales, medicamentos, proveedores, etc.
    - R02F01T01P01 – Ejecutar SHOW TABLES; en MySQL para verificar la creación de las tablas.
- R02F02 – Se debe garantizar la integridad y relaciones entre los datos.
    - R02F02T01 – Definir claves primarias, foráneas y restricciones adecuadas.
    - R02F02T01P01 – Insertar datos de prueba y verificar restricciones (INSERT, SELECT, DELETE).

R03 – El backend debe estar desarrollado en Flask

- R03F01 – Implementar una API RESTful con Flask.
    - R03F01T01 – Crear endpoints para CRUD de clínicas, pacientes, doctores, citas, materiales y medicamentos.
    - R03F01T01P01 – Probar los endpoints con Postman (GET, POST, PUT, DELETE).
- R03F02 – Gestionar la autenticación con tokens JWT.

- R03F02T01 – Implementar login y generación de tokens JWT.
- R03F02T01P01 – Hacer una solicitud con y sin token para verificar el acceso.

R04 – Implementación de CORS para permitir el acceso al backend

- R04F01 – Permitir solicitudes desde el frontend al backend.
  - R04F01T01 – Configurar flask-cors para permitir el acceso desde distintos orígenes.
  - R04F01T01P01 – Hacer una solicitud fetch() desde JavaScript y verificar que no hay errores de CORS.
- R04F02 – Restringir el acceso a dominios específicos si es necesario.
  - R04F02T01 – Configurar Flask-CORS para aceptar solo solicitudes desde un dominio autorizado.
  - R04F02T01P01 – Intentar hacer una solicitud desde un dominio no permitido y verificar el bloqueo.

R05 – Desarrollo del frontend en HTML, CSS y JavaScript

- R05F01 – Diseñar una interfaz responsiva para la gestión de la clínica.
  - R05F01T01 – Crear una página principal con navegación en HTML y CSS.
  - R05F01T01P01 – Abrir index.html en un navegador y comprobar la carga correcta de estilos y estructura.
- R05F02 – Implementar la conexión con el backend usando JavaScript (fetch()).
  - R05F02T01 – Obtener y mostrar datos dinámicamente desde la API Flask.
  - R05F02T01P01 – Inspeccionar la consola del navegador para verificar la respuesta de la API.

R06 – Visualización de datos con Chart.js

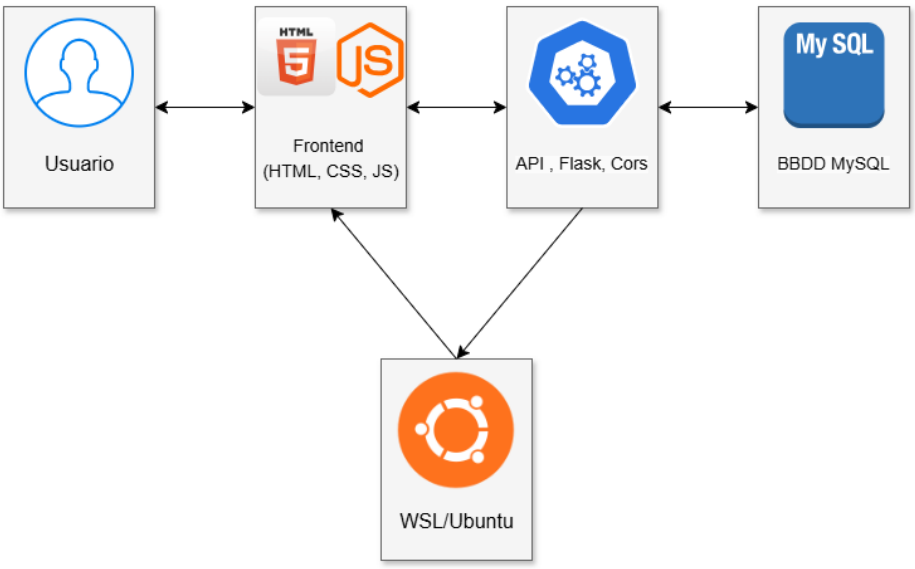
- R06F01 – Representar estadísticas con gráficos dinámicos.
  - R06F01T01 – Implementar gráficos para citas por doctor y pacientes por tratamiento.
  - R06F01T01P01 – Comparar los datos mostrados en los gráficos con los valores en la base de datos.
- R06F02 – Permitir la actualización dinámica de los gráficos.

- R06F02T01 – Actualizar gráficos en tiempo real al recibir nuevos datos.
- R06F02T01P01 – Simular la creación de una nueva cita y comprobar la actualización del gráfico.

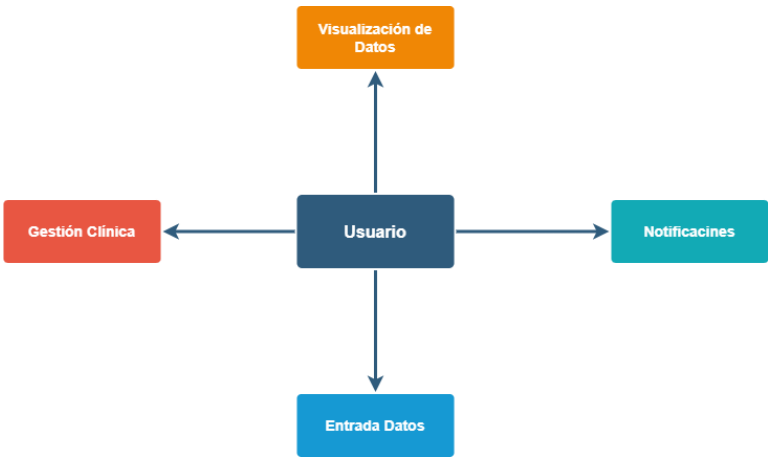
DESCRIPCIÓN

Arquitectura

1. **El usuario** interactúa con la interfaz web (Frontend en HTML, CSS y JavaScript).
2. **El frontend** envía solicitudes a la **API Flask (Backend)** para obtener y gestionar datos.
3. **El backend en Flask** procesa las solicitudes y accede a la **Base de Datos MySQL** para recuperar o actualizar información.
4. **El servidor Ubuntu** aloja el backend y la base de datos, asegurando la conectividad y procesamiento.
5. **El frontend** recibe los datos procesados y actualiza la interfaz en tiempo real, mostrando información y gráficos dinámicos.



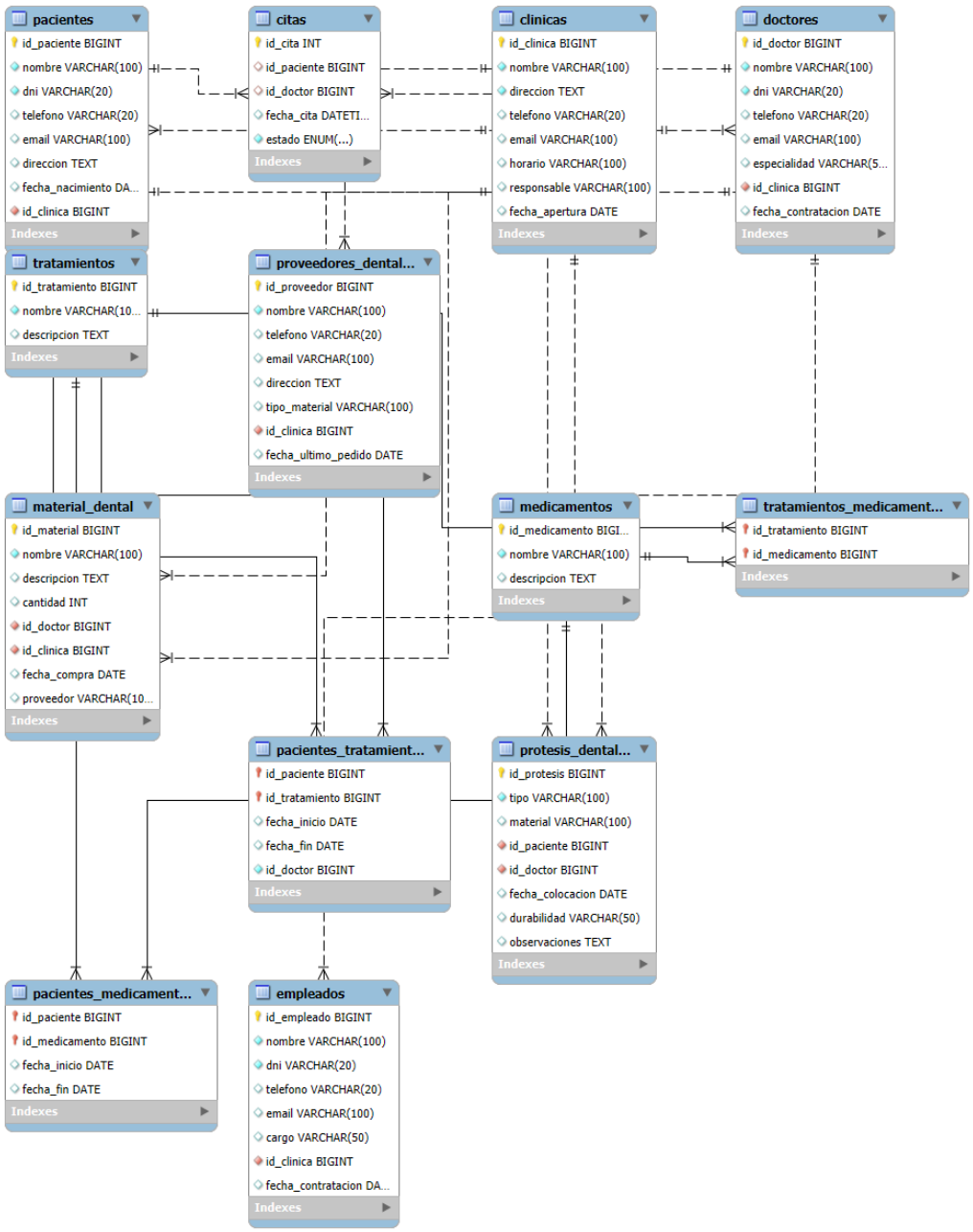
Casos de uso.



Categoría	Usuario	Visualización de Datos	Notificaciones	Entrada de Datos
Descripción	Autenticación	Dashboard	Alertas automáticas	Registro de Información
Funcionalidad	El usuario inicia sesión en el sistema.	Muestra estadísticas con gráficos dinámicos.	Genera alertas sobre citas y stock bajo.	Permite ingresar nuevos datos en el sistema.
Precondiciones	Usuario registrado en la BD.	Usuario autenticado.	Usuario autenticado.	Usuario autenticado.
Postcondiciones	Acceso concedido o denegado.	Datos cargados y mostrados.	Notificación enviada.	Datos guardados correctamente.
Datos de Entrada	Usuario, contraseña.	Solicitudes al backend.	Eventos críticos detectados.	Información del paciente, cita, material, etc.
Datos de Salida	Mensaje de éxito o error.	Gráficos generados.	Mensaje de alerta.	Confirmación de operación.
Tablas	usuarios	N/A	citas, materiales, pacientes	Varios (dependiendo del módulo)
Interfaces	LoginInterface	DashboardInterface	AlertaInterface	FormularioInterface

DISEÑOS

Diagrama de la base de datos. Con detalle de campos.



Tablas principales

1. **pacientes**
    - Almacena la información personal de los pacientes, incluyendo nombre, DNI, teléfono, correo electrónico y fecha de nacimiento.
    - Se relaciona con la tabla **clínicas** mediante el campo id_clinica.
2. **citas**
    - Registra las citas médicas con los campos id_paciente, id_doctor, fecha y estado de la cita.
    - Está vinculada a las tablas **pacientes** y **doctores**.
3. **clinicas**
    - Contiene información sobre las clínicas, como nombre, dirección, teléfono, correo electrónico, horario de atención, responsable y fecha de apertura.
4. **doctores**
    - Almacena los datos de los doctores, incluyendo especialidad, DNI, teléfono, correo electrónico, clínica asociada y fecha de contratación.
    - Se relaciona con **clínicas**, **citas**, **tratamientos**, **materiales dentales** y **prótesis dentales**.
5. **empleados**
    - Contiene información de los empleados de la clínica, incluyendo su nombre, DNI, teléfono, correo electrónico, cargo, clínica asociada y fecha de contratación.

Tablas relacionadas con tratamientos y medicamentos

6. **tratamientos**
    - Registra los tratamientos disponibles, con su respectivo nombre y descripción.
7. **pacientes_tratamientos**

- Establece la relación entre pacientes y tratamientos, incluyendo las fechas de inicio y fin del tratamiento, así como el doctor responsable.

8. **medicamentos**
    - Almacena información sobre los medicamentos utilizados en la clínica, con sus respectivos nombres y descripciones.
9. **pacientes_medicamentos**
    - Relaciona a los pacientes con los medicamentos que se les han prescrito, registrando la fecha de inicio del tratamiento.
10. **tratamientos_medicamentos**

- Define la relación entre los tratamientos y los medicamentos asociados a cada uno.

Tablas relacionadas con materiales y prótesis

11. **material_dental**

- Contiene información sobre los materiales dentales disponibles en la clínica, incluyendo nombre, descripción, cantidad, proveedor, clínica asociada y fecha de compra.

12. **protesis_dental**

- Registra información sobre las prótesis dentales, como tipo, material, paciente, doctor, fecha de colocación, durabilidad y observaciones adicionales.

Tablas de proveedores

13. **proveedores_dentales**

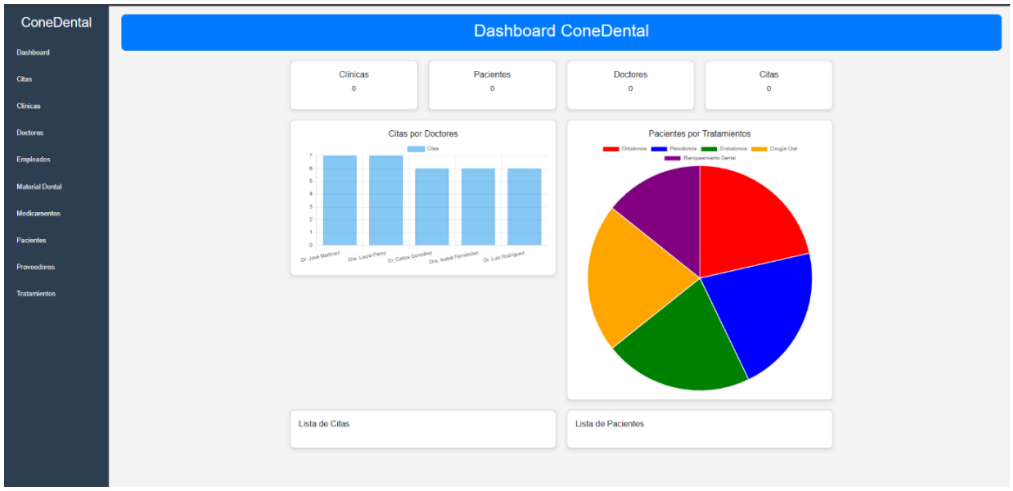
- Contiene datos sobre los proveedores de la clínica, incluyendo nombre, contacto, tipo de material suministrado, clínica asociada y fecha del último pedido.

Relaciones clave

- Los **doctores** están vinculados a **citas**, **tratamientos**, **prótesis dentales** y **materiales dentales**.

- Los **pacientes** tienen relaciones con **citas**, **tratamientos**, **medicamentos** y **prótesis dentales**.
- Los **proveedores dentales** están conectados con **materiales dentales**.
- Los **empleados** pertenecen a una **clínica**, pero no tienen una relación directa con doctores o pacientes.

Interfaces. Interesa ver la solución en diferentes tamaños o dispositivos.



Menú de Navegación (Sidebar)

En el lateral izquierdo se encuentra un menú de navegación con accesos directos a diferentes secciones del sistema. Este menú permite a los usuarios desplazarse entre las distintas funcionalidades, como la gestión de citas, doctores, empleados, materiales dentales, medicamentos y proveedores.

Tarjetas de Resumen

Debajo del encabezado se presentan cuatro tarjetas informativas que muestran el número total de clínicas, pacientes, doctores y citas registradas en el sistema. Estas tarjetas permiten una rápida visualización del estado general de la clínica

Visualización de Datos

El dashboard incluye gráficos dinámicos generados con Chart.js, los cuales representan información clave:

Gráfico de Barras: "Citas por Doctores"

Este gráfico muestra la distribución de citas entre los doctores de la clínica. Cada barra representa a un doctor, y la altura de la barra indica el número de citas asignadas

Gráfico de Pastel: "Pacientes por Tratamientos"

Este gráfico segmenta la cantidad de pacientes según los tratamientos que han recibido. Cada segmento del gráfico representa un tratamiento diferente, identificado con un color distinto.

Listados de Información

En la parte inferior del dashboard hay dos secciones destinadas a listar datos detallados:

Lista de Citas

Debería mostrar información detallada sobre las citas agendadas, incluyendo la fecha, el paciente asignado y el motivo de la cita.

Lista de Pacientes

Se espera que muestre el nombre, DNI y número de teléfono de los pacientes registrados en la clínica.

TECNOLOGÍA

Ubuntu

Sistema operativo basado en Linux, utilizado para servidores y entornos de desarrollo.

Se utiliza como la plataforma base para alojar el servidor web, la base de datos y la aplicación Flask.

MySQL

Sistema de gestión de bases de datos relacional (RDBMS) que permite el almacenamiento y manejo eficiente de datos

Se usa para gestionar la información de la clínica dental, incluyendo pacientes, doctores, citas, materiales y proveedores.

Phython - Flask

Framework web ligero para Python que facilita la creación de aplicaciones y APIs.

Se emplea para desarrollar el backend de la aplicación, gestionar peticiones HTTP y conectar con la base de datos MySQL.

Flask-CORS

Extensión para Flask que permite compartir recursos entre orígenes diferentes (Cross-Origin Resource Sharing)

Se implementa para permitir que el frontend (HTML, CSS, JavaScript) pueda comunicarse con el backend sin restricciones de seguridad.

HTML, CSS y JavaScript

Tecnologías estándar para la creación de interfaces web interactivas y visualmente atractivas.

HTML estructura la interfaz de usuario.
CSS estiliza la interfaz para mejorar la experiencia del usuario.
JavaScript permite la interacción dinámica con el backend mediante **fetch()**.

Chart.js

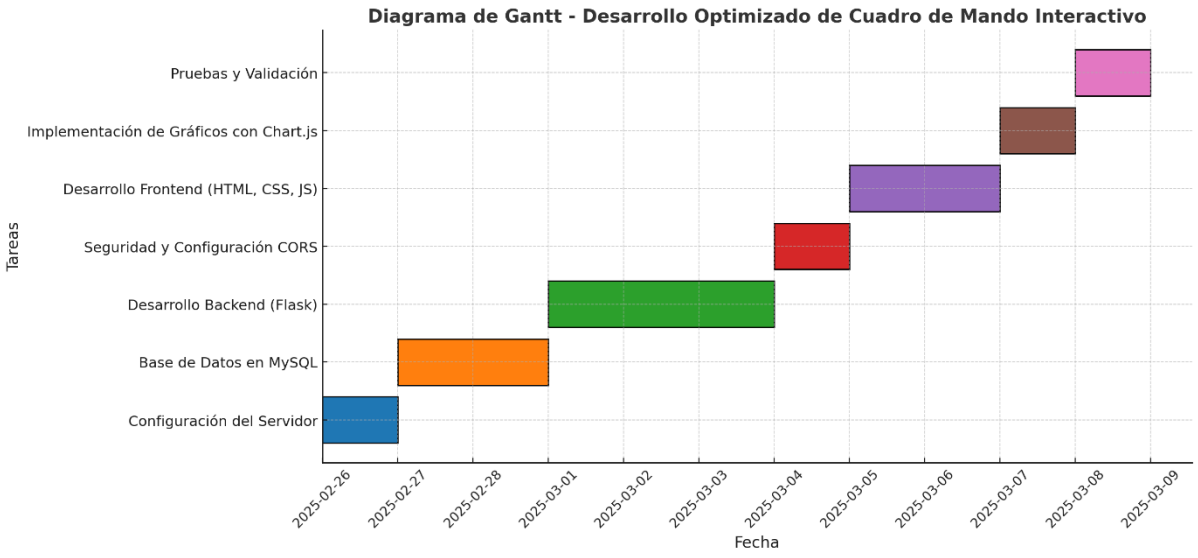
Biblioteca de JavaScript para la visualización de datos mediante gráficos interactivos.

Se utiliza para representar estadísticas clave en el cuadro de mando, como citas por doctor y pacientes por tratamiento.

METODOLOGÍA

Considerando una tarifa de 40€ por hora, el costo total estimado se reduce a 2.400€.

- Configuración del servidor Ubuntu y dependencias en 7 horas, incluyendo la instalación de los paquetes necesarios y la automatización del servicio.
- Implementación de la base de datos en MySQL en 10 horas, con la creación de tablas optimizadas y definición de relaciones.
- Desarrollo del backend con Flask y API RESTful en 13 horas, enfocándose en endpoints eficientes para la gestión de la clínica.
- Integración de seguridad con JWT y configuración de CORS en 6 horas, garantizando el acceso seguro al sistema.
- Desarrollo del frontend en HTML, CSS y JavaScript en 12 horas, asegurando una interfaz intuitiva y responsiva.
- Implementación de gráficos dinámicos con Chart.js en 9 horas, facilitando la visualización de datos en tiempo real.
- Pruebas y validación del sistema en 3 horas, verificando el correcto funcionamiento de cada módulo.



README y GIT.

https://github.com/guillecone/TFG.git

TRABAJOS FUTUROS

- Filtros dinámicos (por fecha, paciente, doctor)
- Alertas y notificaciones (bajo stock de medicamentos, citas pendientes)
- Optimización del rendimiento (carga más rápida, caché)
- Agregar más gráficos
- Mejorar el diseño
- Optimizar la base de datos

CONCLUSIONES

Conclusión profesional del proyecto.

REFERENCIAS

- Bootstrap. (s.f.). Introduction to Bootstrap. Bootstrap. Recuperado el 7 de marzo de 2025, de https://getbootstrap.com/docs/5.3/getting-started/introduction/
- Chart.js. (s.f.). Getting Started. Chart.js. Recuperado el 7 de marzo de 2025, de https://www.chartjs.org/docs/latest/
- Mozilla Developer Network (MDN). (s.f.). Using Fetch. MDN Web Docs. Recuperado el 7 de marzo de 2025, de https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch
- MySQL. (s.f.). MySQL 8.0 Reference Manual. MySQL Documentation. Recuperado el 7 de marzo de 2025, de https://dev.mysql.com/doc/refman/8.0/en/
- Python Software Foundation. (s.f.). Flask Documentation (2.3.x). Python.org. Recuperado el 7 de marzo de 2025, de https://flask.palletsprojects.com/en/2.3.x/
- W3Schools. (s.f.). CSS Flexbox Guide. W3Schools. Recuperado el 7 de marzo de 2025, de https://www.w3schools.com/css/css3_flexbox.asp
- MDN Web Docs. (s.f.). Introduction to CORS. Mozilla Developer Network. Recuperado el 7 de marzo de 2025, de https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
- Microsoft. (s.f.). WSL Documentation: Windows Subsystem for Linux. Microsoft Learn. Recuperado el 7 de marzo de 2025, de https://learn.microsoft.com/en-us/windows/wsl/
- DigitalOcean. (s.f.). How To Install and Use MySQL on Ubuntu 20.04. DigitalOcean Tutorials. Recuperado el 7 de marzo de 2025, de https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04
- FreeCodeCamp. (s.f.). A Beginner's Guide to REST APIs. FreeCodeCamp. Recuperado el 7 de marzo de 2025, de https://www.freecodecamp.org/news/rest-api-tutorial-restful-web-services-for-beginners/