ANTEPROYECTO – FICHAPP

TÍTULO

FichApp (Sistema de Gestión de Fichajes de Entrada y Salida)

Índice

1. Introducción

En los últimos años, con la entrada en vigor de diferentes normativas laborales relacionadas con el control horario, muchas empresas se han visto en la necesidad de registrar de manera fiable las jornadas laborales de sus empleados. Sin embargo, muchas de ellas, especialmente las pequeñas y medianas, siguen utilizando métodos manuales como hojas de cálculo o registros en papel, lo que conlleva imprecisiones, pérdida de tiempo y dificultades para llevar un seguimiento adecuado.

Fichapp nace como una solución tecnológica sencilla, moderna y accesible desde cualquier dispositivo, especialmente desde móviles, para facilitar a los trabajadores el fichaje de entrada y salida, así como la consulta de horarios y vacaciones. Esta aplicación está diseñada para adaptarse a empresas de todos los tamaños y necesidades, ofreciendo una experiencia intuitiva y eficiente tanto para el empleado como para los administradores.

Además de resolver una necesidad real, este proyecto tiene un enfoque didáctico, sirviendo como ejercicio práctico de aplicación de tecnologías web actuales como Vue 3, Vuetify, Node.js y MongoDB, integrando prácticas modernas de despliegue y automatización.

2. Objetivos

El objetivo general del proyecto es desarrollar una aplicación web funcional que permita gestionar las jornadas laborales de los empleados de manera digital, sencilla y eficaz. Los objetivos específicos son:

- Diseñar una interfaz moderna, clara y adaptativa a distintos dispositivos.

- Implementar un sistema de fichaje de entrada y salida.

- Calcular y mostrar en tiempo real el tiempo trabajado.

- Visualizar un historial de fichajes ordenado y filtrado por fechas.

- Consultar y mostrar los horarios laborales y días de vacaciones.

- Crear dos tipos de usuarios: empleados y administradores.(en un futuro próximo)

- Utilizar autenticación con tokens JWT para garantizar la seguridad.

- Automatizar el despliegue mediante GitHub Actions y Vercel.

Historias de usuario:

👨‍💼 Como empleado:

- Quiero registrar mi entrada y salida de forma rápida para registrar mi jornada laboral.
- Quiero ver mi jornada actual y el tiempo trabajado en tiempo real.
- Quiero consultar mis días de vacaciones disponibles, solicitados y aprobados.
- Quiero solicitar vacaciones seleccionando días sueltos o rangos de fechas.
- Quiero ver mis fichajes anteriores en un historial con fechas y tiempos.
- Quiero ver los días festivos próximos en mi localidad.
- Quiero reportar incidencias para comunicar problemas o situaciones especiales.
- Quiero ver el estado de mis incidencias para saber si están en proceso o resueltas.

👨‍💼 Como administrador:

- Quiero ver los fichajes de todos los empleados para controlar la asistencia.
- Quiero asignar horarios laborales a los empleados según su contrato.
- Quiero validar o rechazar solicitudes de vacaciones.
- Quiero modificar el saldo de vacaciones de un empleado si es necesario.
- Quiero gestionar los festivos por localidad.
- Quiero gestionar las incidencias reportadas por los empleados.
- Quiero dar de alta, modificar y eliminar usuarios fácilmente desde un panel.

- Quiero gestionar localidades para asociarlas a empleados y festivos.
- Quiero crear y modificar los tipos de contrato que definan los días de vacaciones.
- Quiero crear y editar horarios (calendarios) incluyendo los tramos horarios diarios.

Este sistema está diseñado exclusivamente para la gestión de fichajes, con el objetivo de hacer su uso lo más ágil y directo posible.

3. Descripción técnica

La aplicación desarrollada sigue una arquitectura de tipo **cliente-servidor basada en capas**, con separación clara entre frontend, lógica de negocio y persistencia de datos. Además, el backend se estructura según el patrón **Modelo–Vista–Controlador (MVC)**, lo que permite una mayor mantenibilidad y escalabilidad del sistema.

🧩 **Tecnologías empleadas por capa:**

- **Capa de presentación (Vista del sistema)**
  Implementada como una SPA (Single Page Application) con **Vue 3**, **Vuetify** y **TypeScript**.
  Se encarga de la interacción con el usuario, el envío de peticiones al backend mediante **Axios** y la visualización dinámica de los datos.
  La gestión del estado global se realiza con **Pinia**.

- **Capa de lógica de negocio (Controlador)**
  Desarrollada en **Node.js** con **Express**, define la **API REST** que actúa como intermediaria entre la vista y los datos.
  Esta capa contiene los controladores que reciben las solicitudes, validan los datos, aplican reglas de negocio y devuelven las respuestas al cliente.

- **Capa de persistencia (Modelo)**
  Se utiliza **MongoDB** como base de datos NoSQL, tanto en entorno local (mediante Docker) como en entorno remoto con **MongoDB Atlas**.
  Los modelos de datos se definen mediante **Mongoose**, permitiendo una estructura clara y validaciones en los esquemas.

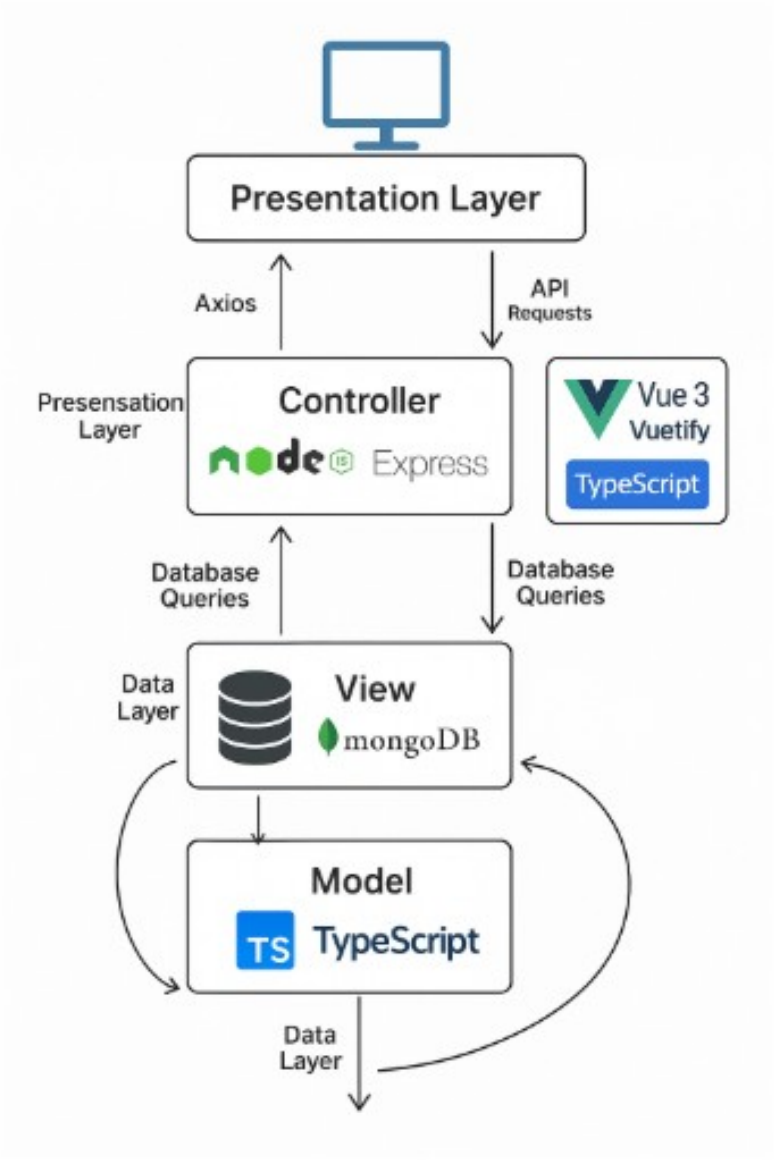
🔐 Autenticación y seguridad

El sistema implementa **autenticación mediante tokens JWT**. La sesión del usuario se mantiene de forma persistente en el cliente utilizando almacenamiento cifrado con AES.

☁️ Despliegue y contenedores

El frontend está desplegado en **Vercel**, mientras que el backend se ejecuta de forma local y también se prepara para producción mediante **Docker**.
Se contempla su despliegue en plataformas como **Render** o **Railway**.
Se emplean workflows de **GitHub Actions** para facilitar la integración y despliegue continuo (CI/CD).

📊 Diagrama de arquitectura



4. Proceso de desarrollo

El desarrollo del proyecto seguirá una **metodología ágil**, concretamente **una adaptación de Scrum**, que permite iterar de forma continua sobre las funcionalidades y validar los avances de manera incremental. Esta metodología favorece la flexibilidad, la mejora continua y el enfoque centrado en el usuario.

Se han definido **fases de desarrollo** que agrupan funcionalidades concretas, las cuales serán validadas individualmente antes de integrarse en el sistema completo. Cada fase puede considerarse una pequeña iteración con sus propias tareas, pruebas y entregables.

Las fases del desarrollo son:

1. **Diseño de la interfaz**
    - Creación de bocetos y wireframes de las pantallas principales: login, fichaje, histórico y vacaciones.
    - Definición de la experiencia de usuario (UX).

2. **Configuración inicial del proyecto**
    - Instalación de Vue 3, Vuetify, Pinia y TypeScript.
    - Organización de carpetas y estructura de componentes.

3. **Autenticación**
    - Desarrollo del formulario de inicio de sesión.
    - Gestión del token JWT y persistencia cifrada en `localStorage`.

4. **Módulo de fichaje**
    - Registro de entradas y salidas.
    - Cálculo y visualización del tiempo trabajado en tiempo real.

5. **Visualización de jornada**
    - Desarrollo de la vista “Mi jornada” con información del día actual.

6. **Vacaciones y horarios**
    - Consulta del calendario laboral, festivos y vacaciones.
    - Solicitud y validación de vacaciones.

- Visualización y gestión de horarios.

7. **Backend propio**
    - Implementación del servidor con Node.js y Express.
    - Diseño del modelo de datos en MongoDB.
    - Desarrollo de rutas protegidas y autenticación.

8. **Dockerización e integración continua (CI/CD)**
    - Creación de `Dockerfile` y docker-compose para backend.
    - Automatización del despliegue mediante GitHub Actions y Vercel.

9. **Pruebas y documentación final**
    - Verificación de funcionalidades.
    - Redacción del manual de usuario y la memoria técnica.

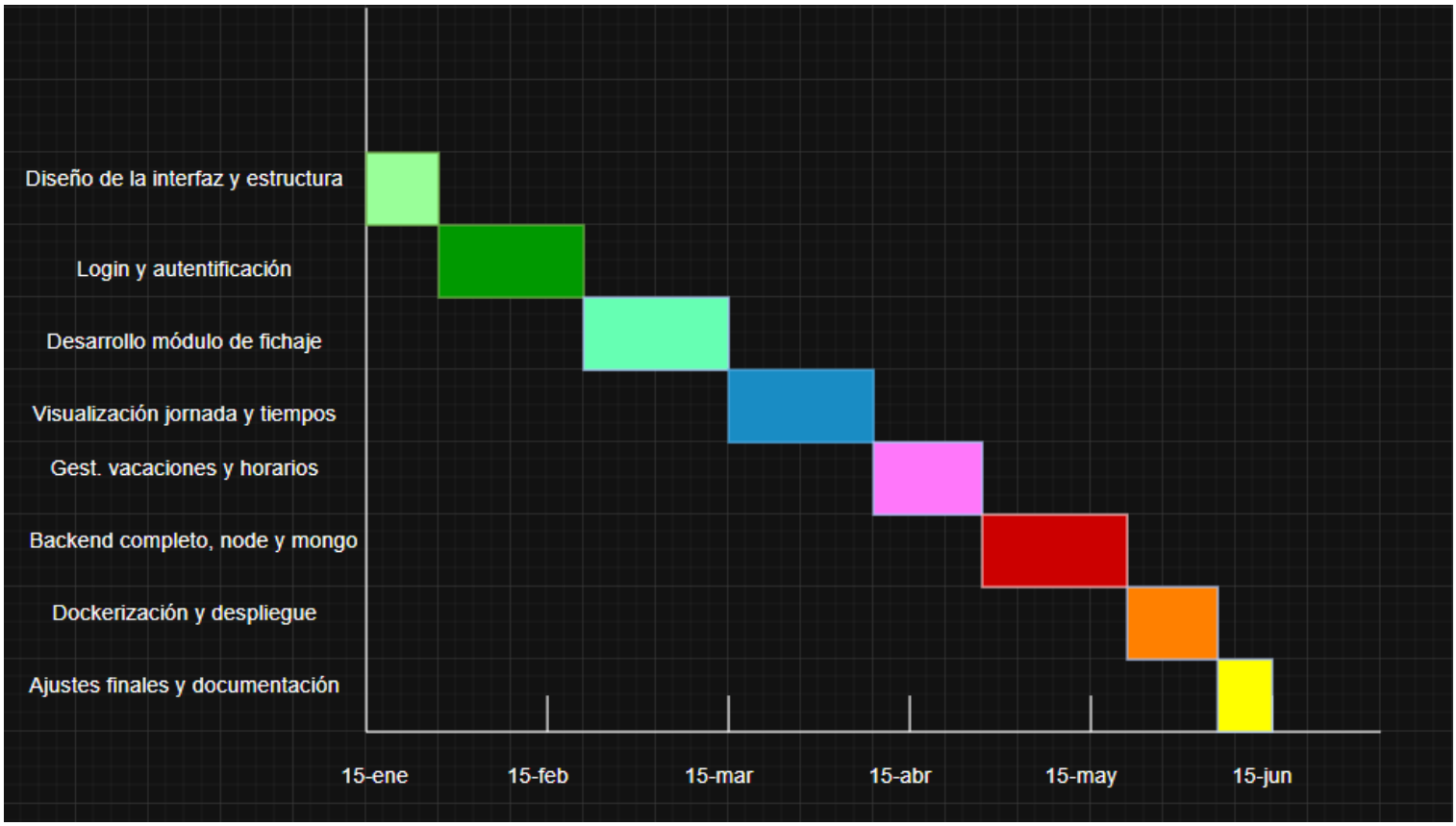
5. Planificación del trabajo y estimación temporal

La duración total del proyecto será de aproximadamente 8 semanas, con una dedicación estimada de 15 horas semanales. A continuación se detalla la planificación prevista:

Nº	Actividad	Fecha inicio	Fecha fin	Duración (días)
1	Diseño de la interfaz y estructura del proyecto	15/01/2025	26/01/2025	12

2	Login y autenticación (frontend + backend básico)	27/01/2025	19/02/2025	24
3	Desarrollo del módulo de fichaje	20/02/2025	15/03/2025	24
4	Visualización de la jornada y tiempos trabajados	16/03/2025	08/04/2025	24
5	Gestión de vacaciones y horarios	09/04/2025	29/04/2025	21
6	Backend completo con Node.js y MongoDB	30/04/2025	25/05/2025	26
7	Dockerización, CI/CD y despliegue (Vercel, Render)	26/05/2025	08/06/2025	14
8	Pruebas, validación, documentación y entrega final	09/06/2025	15/06/2025	7

Diagrama de Gantt



6. Descripción de la documentación a entregar

A la finalización del proyecto FichApp, se entregará un conjunto de documentos y archivos que reflejan el desarrollo técnico, funcional y metodológico del sistema. La documentación incluye tanto materiales técnicos como de usuario, así como evidencias del proceso de trabajo. Los elementos entregables serán los siguientes:

🗂️ Código fuente completo

- Estructurado en dos repositorios o carpetas: **frontend** (Vue 3 + Vuetify + TypeScript) y **backend** (Node.js + Express + MongoDB).

- Incluye todos los componentes, servicios, rutas, controladores y modelos utilizados.

📘 Manual de usuario

- Documento dirigido al usuario final (empleado y administrador).
- Contiene capturas de pantalla de las vistas principales.
- Describe paso a paso cómo utilizar el sistema: fichaje, consulta de jornada, solicitud de vacaciones, etc.

🧩 Esquema de base de datos

- Modelo documental de MongoDB con los principales esquemas (usuarios, fichajes, vacaciones, horarios, incidencias, etc.).
- Representación visual de la estructura de datos mediante diagramas.
- Incluye relaciones y campos clave de cada colección.

📜 Memoria técnica

- Documento principal del proyecto que recoge:
    - Justificación, objetivos y alcance.
    - Arquitectura técnica y tecnologías utilizadas.
    - Proceso de desarrollo y planificación temporal.
    - Pruebas realizadas y conclusiones.

⚙️ **Configuración de despliegue**

- Descripción paso a paso del proceso de despliegue:
    - Dockerización del backend.
    - Publicación en Vercel (frontend) y Render (backend).
    - Configuración de variables de entorno.
- Se incluye copia del archivo `dockerfile`, `docker-compose.yml` y `vercel.json` si corresponde.

🗓️ **Diagrama de Gantt**

- Representación gráfica de la planificación temporal del proyecto.
- Refleja tareas, fechas de inicio y fin, y duración de cada fase.
- Elaborado según el cronograma del punto 5.

🔄 **Documento de control de versiones (Git)**

- Historial de commits y ramas del repositorio.
- Se detallan las funcionalidades añadidas progresivamente.
- Evidencia del uso de Git y GitHub como sistema de control de versiones distribuido.

7. Medios materiales necesarios

Para el desarrollo completo del proyecto FichApp, se requieren los siguientes recursos técnicos, herramientas y servicios. A continuación se detallan sus características, versiones, posibles costes y en qué fases del proyecto serán utilizados:

🖥️ Ordenador personal con acceso a internet

- **Descripción**: Equipo de trabajo principal donde se desarrollará todo el proyecto (frontend, backend, despliegue y pruebas).
- **Requisitos mínimos**: Procesador de 4 núcleos, 8 GB de RAM, sistema operativo Windows 10, macOS o Linux.
- **Versión**: Libre elección (se ha trabajado en entorno Windows 11).
- **Coste**: Ninguno adicional (propio del estudiante).
- **Fases**: Todas las fases del proyecto.

👨‍💻 Editor de código – Visual Studio Code

- **Descripción**: Entorno de desarrollo ligero y ampliamente extendido, con soporte para Vue, TypeScript, Node.js y extensiones útiles (ESLint, Prettier, etc.).
- **Versión**: Visual Studio Code 1.87.0 o superior.
- **Coste**: Gratuito.
- **Fases**: Todas las fases de desarrollo y pruebas.

🌐 Cuenta en GitHub

- **Descripción**: Plataforma de control de versiones y colaboración en el desarrollo. Se utilizará para almacenar el código fuente, documentar cambios y automatizar despliegues.
- **Versión**: GitHub Free (usuario individual).
- **Coste**: Gratuito.
- **Fases**: Desde la fase de configuración inicial hasta la entrega final (incluye CI/CD).

🚀 Cuenta en Vercel

- **Descripción**: Plataforma de despliegue automatizado del frontend, con integración directa con GitHub y optimizada para proyectos en Vue.
- **Versión**: Vercel Free Plan.
- **Coste**: Gratuito.
- **Fases**: Fase de despliegue (Dockerización y CI/CD) y entrega final.

🐳 Docker Desktop

- **Descripción**: Herramienta de contenedorización que permite ejecutar el backend localmente y preparar el despliegue en plataformas como Render.
- **Versión**: Docker Desktop 4.27.2 o superior.
- **Coste**: Gratuito para uso personal o educativo.
- **Fases**: Fase 6 (backend), fase 7 (Dockerización) y pruebas.

🌐 Navegador web actualizado

- **Descripción**: Herramienta imprescindible para probar la interfaz de usuario y depurar la aplicación.
- **Versión**: Se recomienda Google Chrome 120.0 o superior o Mozilla Firefox 120.
- **Coste**: Gratuito.
- **Fases**: Todas las fases de desarrollo y pruebas.

🍃 MongoDB Atlas (base de datos en la nube)

- **Descripción**: Servicio de base de datos MongoDB en la nube. Permite alojar la base de datos del proyecto y hacer pruebas desde cualquier entorno.
- **Versión**: MongoDB Atlas Free Tier (M0).

- **Coste**: Gratuito (hasta 512 MB de almacenamiento).
- **Fases**: Fase 6 (backend), fase 7 (CI/CD), fase 8 (pruebas y entrega).

✅ Resumen

Todos los recursos utilizados en el desarrollo de FichApp han sido seleccionados por su disponibilidad gratuita, compatibilidad con tecnologías modernas y facilidad de uso. La mayoría son multiplataforma y ampliamente documentados, lo cual facilita el desarrollo y aprendizaje.

8. despliegue

El proyecto FichApp cuenta con un despliegue dividido en dos partes principales: **frontend** y **backend**, los cuales están alojados y automatizados con herramientas modernas de integración y entrega continua.

🚀 Despliegue del frontend

El frontend, desarrollado en **Vue 3 + Vuetify + TypeScript**, se ha desplegado en la plataforma **Vercel**, que permite:

- Despliegue automático mediante conexión con GitHub.
- Acceso público e inmediato al proyecto desde cualquier dispositivo.
- Optimización automática del rendimiento y CDN global.

Cada vez que se realiza un push a la rama principal del repositorio, **Vercel realiza una nueva build y publica automáticamente los cambios**, asegurando una integración continua efectiva y sin intervención manual.

🛠️ Despliegue del backend

El backend, desarrollado con **Node.js + Express**, ha sido **dockerizado** para garantizar su portabilidad y facilitar su despliegue en diferentes entornos.

Actualmente se encuentra desplegado en **Render**, una plataforma en la nube que permite ejecutar contenedores Docker y expone la API REST utilizada por el frontend. Esta API se encuentra conectada con una base de datos en la nube mediante **MongoDB Atlas**.

El backend cuenta con:

- Endpoint `/api` accesible desde producción.
- Soporte para rutas protegidas mediante JWT.
- Conexión cifrada con MongoDB Atlas.
- Variables de entorno configuradas desde el panel de Render.

⚙️ Integración y despliegue continuo (CI/CD)

Para garantizar un flujo de trabajo ágil y profesional, se han implementado **GitHub Actions** que automatizan tareas como:

- Build del frontend y backend en cada push.
- Validación del código.
- Publicación en Vercel (frontend) y Render (backend, vía Docker Hub en el futuro).
- Verificación de despliegue exitoso.

Estado actual del despliegue

Componente	Plataforma	Estado	URL
Frontend	Vercel	Desplegado	(tu-url.vercel.app)

Backend	Render	Desplegado	(tu-api.onrender.com)

Se ha verificado el funcionamiento conjunto entre frontend y backend en entorno de producción.

9. Bibliografía

- **Vue.js** — Framework progresivo de JavaScript utilizado para desarrollar la interfaz de usuario como SPA (Single Page Application).
  https://vuejs.org

- **Vuetify** — Biblioteca de componentes UI basada en Material Design, empleada para el diseño visual y responsivo del frontend.
  https://vuetifyjs.com

- **Node.js** — Entorno de ejecución en JavaScript para el desarrollo del servidor backend.
  https://nodejs.org

- **MongoDB** — Base de datos NoSQL orientada a documentos, utilizada para almacenar la información de usuarios, fichajes y vacaciones.
  https://www.mongodb.com

- **Vercel** — Plataforma en la nube utilizada para el despliegue automatizado del frontend con integración continua.
  https://vercel.com

- **Docker** — Plataforma de contenedores que permite empaquetar el backend y facilitar su ejecución en diferentes entornos.
  https://www.docker.com

- **GitHub Actions** — Herramienta de automatización de flujos de trabajo para CI/CD, utilizada en este proyecto para despliegue y control de versiones.
  https://docs.github.com/actions

10. Observaciones

El sistema **FichApp** ha sido diseñado con un enfoque **modular y escalable**, lo que permitirá su evolución futura sin comprometer la estabilidad de las funcionalidades existentes.

Si bien el alcance actual cubre las necesidades básicas de gestión de fichajes, jornadas y vacaciones, se contempla una ampliación progresiva de características, en función de los requerimientos de los usuarios y del contexto empresarial.

Entre las posibles mejoras futuras se encuentran:

- **Exportación de informes en PDF**: generación de resúmenes de jornada, fichajes o vacaciones para facilitar su impresión o archivo.
- **Fichaje manual con validación administrativa**: opción para registrar fichajes no realizados a tiempo, siempre sujetos a revisión y aprobación por parte del responsable.

- **Panel de estadísticas**: visualización gráfica de datos como tiempos medios de trabajo, incidencias, fichajes por día, etc.
- **Integración con Google Calendar o notificaciones por email**: automatización de recordatorios o eventos relevantes para el usuario.
- **Gestión avanzada de incidencias**: categorización, seguimiento por estado y comentarios entre empleado y administrador.

Estas funcionalidades no forman parte del alcance inicial, pero la arquitectura del sistema las contempla y permite su integración en fases posteriores.

11. Uso de herramientas de IA en el proyecto

¿Qué herramienta de IA generativa usaste (nombre y versión)?
Durante el desarrollo del presente anteproyecto se ha utilizado la herramienta de inteligencia artificial generativa **ChatGPT (modelo GPT-4, de OpenAI)**.

¿Para qué usaste la herramienta?

Se ha empleado como apoyo para estructurar ideas, mejorar la redacción y revisar el estilo del contenido, así como para resolver dudas relacionadas con la redacción técnica y la organización del documento.

¿Qué contenidos fueron generados por el agente? Etiquétalos.

Todos los contenidos han sido desarrollados con ayuda y consulta de la IA, que ha sido utilizada como una herramienta de apoyo y referencia.

¿Cómo has utilizado o cambiado la salida de la IA generativa?

El contenido propuesto por la herramienta se utilizó como base de trabajo para el desarrollo del anteproyecto. A partir de los textos sugeridos, se realizaron ajustes de estilo, cambios en la redacción, incorporación de detalles personales del proyecto y reorganización de párrafos para que el resultado final se adaptase plenamente al objetivo deseado.

El uso de la IA ha sido, por tanto, un recurso para **pulir, estructurar y reforzar** las ideas propias, pero en ningún caso una sustitución del trabajo individual.