
Anteproyecto

JANOO

“...Jano quedó definido por su aspecto básico: un dios de transición, que con una cara mira al pasado o inicio y con la otra mira al futuro o al final...” Abel G.M.



Imagen de *L'antiquité expliquée et représentée en figures* por [Bernard de Montfaucon](#). <https://es.wikipedia.org/wiki/Jano>

1. Introducción2. ObjetivosObjetivo GeneralObjetivos EspecíficosPropuesta de Valor / DiferenciaciónHistorias de Usuario3. Estructura del Proyecto3.1 Backend: REST Server con Clean Architecture3.2 Frontend: Aplicación en Angular4. Plan de MVP y CronogramaPlanificaciónCronograma y Gestión del tiempo5. AlcanceAlcanceLimitacionesRequerimientos6. Recursos e infraestructura: recursos materiales y humanosEstudio de MercadoFilosofía, Visión y RSCResponsabilidad Social Corporativa (RSC)Modelo Financiero InicialModelo Financiero FuturoPlan de MarketingObligaciones legales, fiscales y Prevención de riesgos laboralesPlan de Prevención de Riesgos LaboralesFinanciamiento7. Recursos e infraestructura: recursos en el área de desarrolloHardware y Software:8. Despliegue9. Análisis de Riesgos10. Bibliografía11. Contribución de la Inteligencia Artificial al proyecto12. Conclusiones y Observaciones13. Líneas futurasPlan de calidad y control del proyecto

1. Introducción

La gestión de horarios, días festivos y vacaciones en el entorno empresarial actual requiere soluciones robustas y eficientes. La presente propuesta tiene como objetivo desarrollar una aplicación web capaz de gestionar el fichaje de empleados de forma precisa y segura.

Este proyecto nace con el propósito de cubrir la necesidad de una herramienta de gestión del tiempo basada en las mejores prácticas del desarrollo de software, que garantice escalabilidad y accesibilidad. De este modo, se busca ofrecer una solución comercializable a organizaciones que demandan una gestión avanzada del tiempo laboral de sus empleados.

Motivación Personal

Este proyecto tiene un significado personal especialmente relevante, ya que aborda una problemática que he comentado en múltiples ocasiones con mis compañeros en Plexus. En la actualidad, la herramienta de gestión de fichajes utilizada por la empresa no está optimizada, lo que genera ineficiencias dentro de un equipo dedicado precisamente al desarrollo de software. Como organización que comercializa soluciones tecnológicas, resulta incoherente operar con un sistema que no se ajusta a las buenas prácticas del sector, y que además presenta errores en la gestión del tiempo y los recursos humanos.

Personalmente busco avanzar desde un rol de maquetación web hacia un perfil full-stack, saliendo de mi zona de confort para desarrollar soluciones tecnológicas completas y escalables. El proyecto Janoo representa la oportunidad perfecta para fusionar sus competencias en RRHH, marketing y desarrollo en un producto con alto impacto empresarial.

Más allá de ser una necesidad interna, esta iniciativa representa una excelente oportunidad para diseñar y construir una herramienta escalable, adaptable y con alto potencial de comercialización. A medida que se superen los hitos iniciales del MVP, será posible incorporar módulos adicionales como la imputación de horas a proyectos de desarrollo, integración con herramientas de gestión, entre otras funcionalidades clave. Esto no solo contribuirá a optimizar la eficiencia operativa dentro de la propia empresa, sino que también permitirá posicionar la aplicación como una solución competitiva y personalizable para otras organizaciones con necesidades similares en la gestión del tiempo laboral.

Contexto

Presentación de la Idea

Desarrollar una aplicación web accesible que permita el registro de fichajes de entrada y salida de los empleados, visualice días festivos y vacaciones asignadas, y muestre una vista clara de la jornada laboral y las horas trabajadas.

Problema - Necesidad

La herramienta actual de gestión de fichajes es ineficiente y no está optimizada, generando errores en la asignación de tiempo, acumulación incorrecta de horas y dificultando el control de recursos en un equipo de desarrollo de software.

Perfil de la promotora

Sólida formación superior en Ciencias Empresariales (especialidad en Recursos Humanos), Marketing y Publicidad, y Desarrollo de Aplicaciones Web. Este perfil multidisciplinar combina experiencia en gestión de personas, estrategias de mercado y competencias técnicas, creando un valioso tandem entre áreas que normalmente están desconectadas.

Experiencia profesional relevante

- Maquetadora web: diseño e implementación de interfaces orientadas a la usabilidad en proyectos corporativos (Banco Santander).
- Responsable de departamentos: liderazgo de equipos y procesos en Marketing, Atención al Cliente y Producción, aportando una visión estratégica y operativa.
- Administrativa: coordinación de tareas administrativas, garantizando organización y cumplimiento de procedimientos internos.

Habilidades técnicas

- Angular
- Symfony
- Adobe Illustrator & InDesign

Habilidades personales

- Flexibilidad y adaptabilidad
 - Curiosidad y aprendizaje continua
 - Orientación a procesos claros combinada con capacidad de innovar.
-

2. Objetivos

Objetivo General

Desarrollar una aplicación web accesible que permita el registro de fichajes (entrada y salida), la visualización de días festivos y vacaciones, y la administración de usuarios a través de un sistema seguro de autenticación, todo ello con una arquitectura escalable y fácil de mantener.

Objetivos Específicos

- Registro de fichajes: Permitir que los empleados registren sus entradas y salidas de forma intuitiva.
- Visualización de información laboral: Mostrar de forma clara los días festivos, vacaciones asignadas y horas trabajadas.
- Sistema de autenticación: Implementar un mecanismo de seguridad (JWT y Bcrypt) que garantice que solo empleados autorizados accedan a sus datos.
- Arquitectura escalable: Diseñar la solución para facilitar futuras ampliaciones (nuevos módulos, integraciones, etc.).
- Compatibilidad multiplataforma: Asegurar que la aplicación funcione correctamente en diversos dispositivos.

Propuesta de Valor / Diferenciación

Janoo combina funcionalidad interna y potencial comercial: ofrece un panel personalizable para empleados, cálculo automático de horas y gestión de permisos, con una arquitectura escalable que permitirá añadir módulos (imputación de horas a

proyectos, integración con herramientas de gestión, etc.), diferenciándose de las soluciones existentes por su flexibilidad y enfoque en la experiencia de usuario (UX).

Historias de Usuario

Sabemos que la gestión eficiente del tiempo y la administración de recursos humanos se han convertido en elementos clave para el éxito organizacional. Este proyecto tiene como objetivo desarrollar una aplicación web fundamentada en las mejores prácticas del desarrollo de software y en un enfoque ágil, que no solo facilita la adaptación a cambios en los requerimientos, sino que también permite entregar valor de manera continua y temprana a nuestros usuarios.

Para lograr este objetivo, se ha optado por trabajar con metodologías ágiles basadas en historias de usuario, un mecanismo efectivo para captar y comunicar de forma sencilla las necesidades reales del cliente. Cada historia de usuario refleja un fragmento funcional del sistema, expresado en términos claros que responden al "quién", "qué" y "por qué", y permite construir soluciones de alto valor.

- **Historia 1: Registro de entrada y salida**

Como empleado. Quiero registrar mis horas de entrada y salida a través de un botón en mi panel personal para que se contabilice mi jornada laboral de forma precisa

Criterios de aceptación:

1. La interfaz muestra botones claramente identificados para "Check-in" y "Check-out".
2. Al pulsar uno de estos botones, se genera un registro en la base de datos con la fecha y hora actuales.
3. Se muestra un mensaje de confirmación tras el registro exitoso.

- **Historia 2: Visualización del estado actual de fichaje**

Como empleado. Quiero visualizar en mi panel los registros del día (entrada y, si procede, salida) para que pueda confirmar que mi jornada se está registrando correctamente

Criterios de aceptación:

1. El panel muestra claramente la hora de entrada y, de haberla, la de salida.
2. Si falta el registro de salida, se destaca para que el usuario lo complete.
3. La información se actualiza en tiempo real tras cada acción.

- **Historia 3: Calendario de días festivos y vacaciones (Futuro)**

Como empleado. Quiero consultar un calendario que indique los días festivos y mis vacaciones aprobadas para que pueda planificar mis actividades y gestionar mi tiempo eficientemente

Criterios de aceptación:

1. El calendario es accesible desde el panel de control.
2. Se distinguen visualmente los días festivos (por ejemplo, con un color específico o ícono) y los días de vacaciones personales.
3. La información se actualiza automáticamente en función de las aprobaciones.

- **Historia 4: Reporte de asistencia para administración (Futuro)**

Como administrador o responsable de RRHH. Quiero generar un reporte que resuma las horas trabajadas, días festivos y vacaciones de los empleados para que se pueda realizar la gestión de nómina y evaluaciones de desempeño

Criterios de aceptación:

1. La herramienta permite filtrar por empleado, departamento y rango de fechas.
2. Se muestran totales resumidos y desgloses por período.
3. Sistema de autenticación
4. Valorar que el reporte se pueda exportar en formatos estándar¹.

¹ Esta posibilidad aún la estoy evaluando.

- **Historia 5: Registro de nuevo usuario**

Como nuevo usuario. Quiero registrarme proporcionando mis datos personales y de contacto para que se cree mi cuenta de forma segura en el sistema

Criterios de aceptación:

1. El formulario de registro valida campos obligatorios (correo, contraseña, etc.).
2. La contraseña se cifra utilizando Bcrypt.
3. Se confirma el registro mediante un mensaje o redirección a la pantalla de login.

- **Historia 6: Inicio de sesión seguro**

Como usuario registrado. Quiero iniciar sesión introduciendo mi correo y contraseña para que pueda acceder a mi panel personal a través de un mecanismo seguro basado en JWT

Criterios de aceptación:

1. Al introducir credenciales válidas, se genera un token JWT que se almacena en el localStorage.
2. Se muestran mensajes de error en caso de credenciales incorrectas.

- **Historia 7: Interfaz responsive y accesible**

Como usuario. Quiero que la aplicación se adapte correctamente a distintos dispositivos para que pueda utilizar el sistema de manera cómoda desde cualquier dispositivo

Criterios de aceptación:

1. La interfaz desarrollada con Angular y Angular Material se adapta a diferentes resoluciones.
2. Se realizan pruebas de usabilidad en varios navegadores y dispositivos.
3. Se cumplen las pautas de accesibilidad básicas (WCAG 2.1).

3. Estructura del Proyecto

Capas de arquitectura de la aplicación y tecnologías asociadas		
Capa	Tecnologías	Funcionalidades
Frontend	Angular, TypeScript, HTML5, SCSS	Desarrollo de la interfaz de usuario (UI), fichaje, visualización de jornadas y vacaciones
Backend	Node.js, Express.js	REST Server clean architecture y autenticación JWT
Base de datos	MongoDB/Mongoose	Almacenamiento de fichajes, horarios, empleados, días festivos y vacaciones
Despliegue	Docker, Mongo Atlas DB, Vercel y Render	Despliegue escalable y gestión de la infraestructura

La solución se compone de dos módulos principales:

- **Backend (REST Server)** con Clean Architecture: Provee una API REST para la gestión de autenticación, usuarios, fichajes y otros recursos.
<https://github.com/lenvigo/janoo-back>
- **Frontend (Angular)**: Aplicación que consume la API y proporciona una interfaz moderna, responsive y profesional. <https://github.com/lenvigo/janoo-front-app>

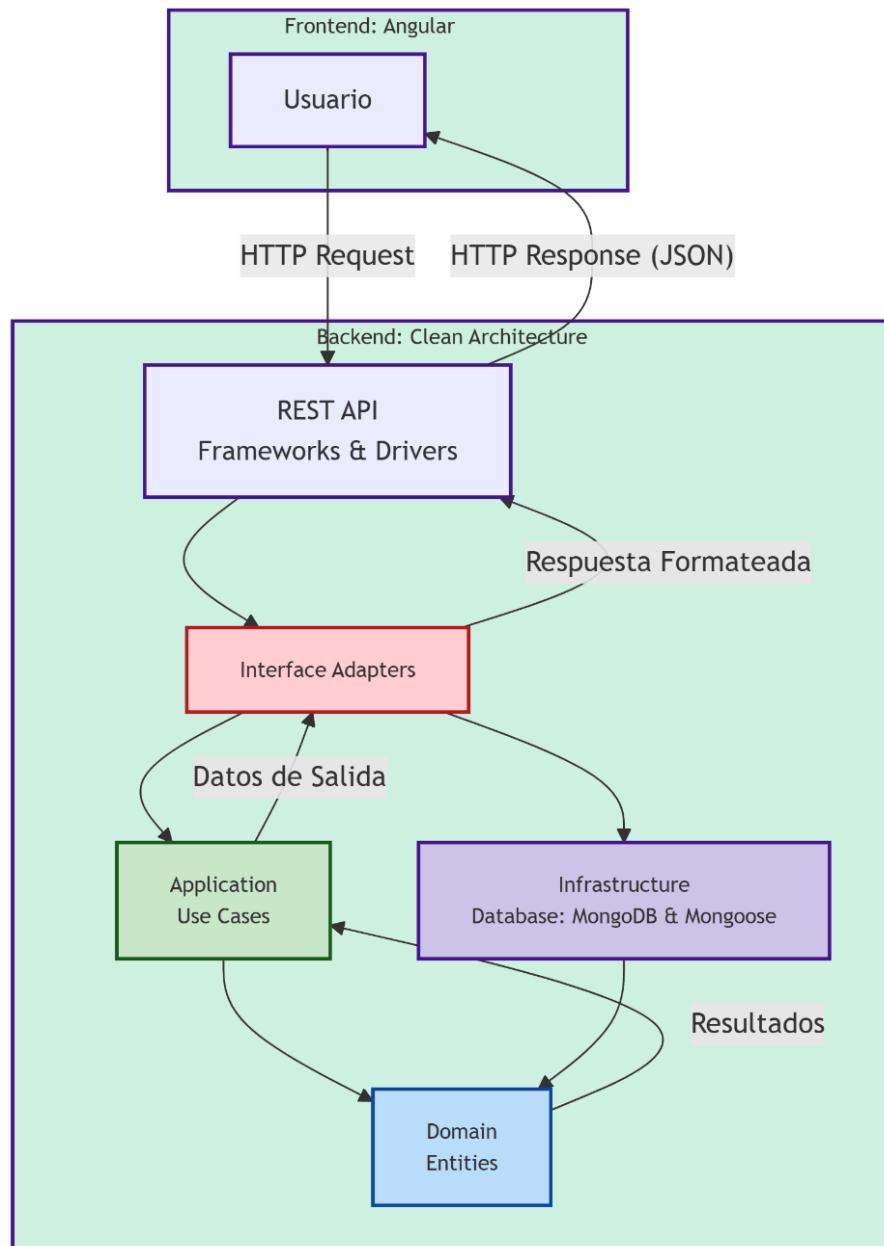


Figura 2. Flujo de interacción Frontend – Backend según el enfoque Clean Architecture

3.1 Backend: REST Server con Clean Architecture

```
src/
  └── presentation/          # Capa de presentación
      ├── auth/              # Autenticación
      ├── checkin/            # Control de asistencia
      ├── incident/            # Gestión de incidencias
      ├── schedule/            # Programación
      ├── user/                # Gestión de usuarios
      ├── vacation/            # Gestión de vacaciones
      ├── middlewares/         # Middlewares personalizados
      ├── routes.ts             # Definición de rutas
      └── server.ts             # Configuración del servidor

  └── domain/                # Capa de dominio
      ├── datasources/         # Fuentes de datos
      ├── dtos/                 # Data Transfer Objects
      ├── entities/             # Entidades del dominio
      ├── errors/               # Errores personalizados
      ├── repositories/          # Interfaces de repositorios
      ├── use-cases/             # Casos de uso
      └── index.ts               # Exportaciones del dominio

  └── infrastructure/          # Capa de infraestructura
      ├── datasources/          # Implementaciones de fuentes de datos
      ├── repositories/          # Implementaciones de repositorios
      ├── mappers/               # Mapeadores de datos
      └── index.ts               # Exportaciones de infraestructura

  └── data/                  # Capa de datos

  └── config/                # Configuración general

  └── types/                 # Tipos y interfaces

  └── __tests__/              # Pruebas unitarias

  └── __integration__/         # Pruebas de integración

  └── app.ts                  # Punto de entrada
```

Enfoque Clean Architecture:

El backend se organiza en capas que separan la lógica de negocio de la infraestructura técnica. Esto se logra aplicando los siguientes patrones y principios:

- *Inversión de Dependencias:*

Las capas del sistema dependen de abstracciones (interfaces) en lugar de implementaciones concretas. Por ejemplo, la capa del dominio define contratos para el acceso a datos sin conocer la tecnología subyacente.

- *Repository Pattern:*

Se define un contrato para acceder a los datos, implementado en la capa de infraestructura. Esto facilita la realización de pruebas unitarias y el eventual cambio de tecnología sin modificar la lógica de negocio.

- *Interfaz Adapter Pattern:*

Los adaptadores (controllers y gateways) se encargan de traducir peticiones HTTP a llamadas a la lógica de negocio del dominio, manteniendo a este último aislado de detalles externos.

- *Domain-Driven Design (DDD):*

Se centra en el dominio del negocio, definiendo entidades, objetos de valor y casos de uso, lo que permite modelar fielmente la lógica y reglas de negocio de la aplicación.

Capas principales del backend:

1. Dominio (Domain):

- Contiene las entidades (por ejemplo, Usuario, Fichaje) y las reglas de negocio (casos de uso) sin dependencia de frameworks o bases de datos.

2. Aplicación (Application):

- Coordina los casos de uso y define las interfaces de repositorios y servicios que serán implementados más adelante.

3. Adaptadores de Interfaces (Interface Adapters):

- Implementa los controladores que reciben peticiones y los gateways que adaptan datos entre formatos externos y las entidades del dominio.

4. Infraestructura (Infrastructure):

- Implementa las conexiones a sistemas externos (MongoDB con Mongoose), los repositorios concretos y otros servicios técnicos.

5. Presentación (Presentation):

- Configura el REST server (Express.js) y define los endpoints (por ejemplo, `/register`, `/login`, `/users`).

Tecnologías del backend:

- Servidor: Node.js y Express.js.
- Persistencia: MongoDB administrada con Mongoose.
- Seguridad: JWT para autenticación y Bcrypt para el cifrado de contraseñas.

Para el backend de este proyecto se eligieron Node.js, Express.js, MongoDB administrada con Mongoose junto con JWT y Bcrypt por las siguientes razones²:

- Servidor: Node.js y Express.js

Se eligieron por su capacidad para ejecutar JavaScript en el servidor de manera asíncrona y eficiente, lo que resulta ideal para aplicaciones en tiempo real y de alta concurrencia. Express.js, al ser un framework minimalista y flexible, facilita la creación de APIs REST mediante la utilización de middleware para manejar peticiones, errores y otras funcionalidades de forma modular. Además, ofrecen una curva de aprendizaje relativamente rápida y se integran de forma natural con el ecosistema JavaScript, lo cual es ventajoso para equipos de desarrollo que ya trabajan en el frontend con Angular.

- Persistencia: MongoDB administrada con Mongoose

MongoDB es una base de datos NoSQL que ofrece alta escalabilidad y flexibilidad en el manejo de datos, lo que permite adaptar los esquemas sin la rigidez de las bases de datos relacionales. Mongoose se utiliza como una capa de modelado de datos, lo que facilita la definición, validación y transformación de los datos al interactuar con MongoDB. Esta capacidad de adaptación es especialmente adecuada para aplicaciones que

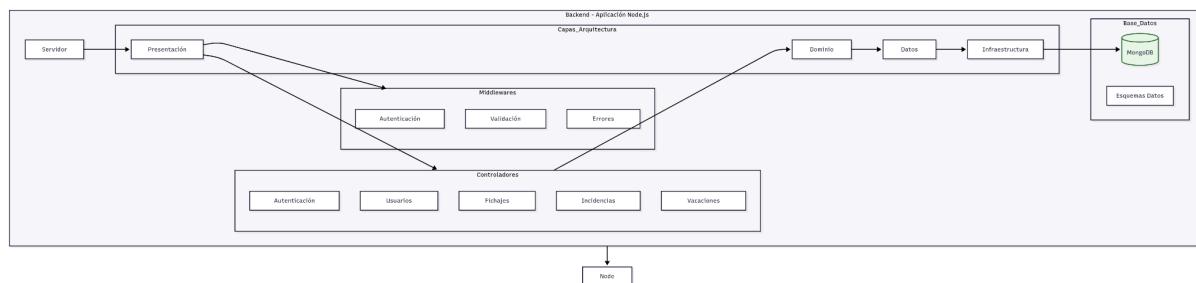
² Generado por ChatGPT

pueden tener estructuras de datos variables como en sistemas de fichaje, donde se registran diversos tipos de eventos y datos de usuarios.

- Seguridad: JWT para autenticación y Bcrypt para el cifrado de contraseñas

JWT (JSON Web Tokens) proporciona un método seguro y escalable para manejar la autenticación sin necesidad de mantener un estado de sesión en el servidor. Esto facilita el desarrollo de aplicaciones distribuidas y mejora el rendimiento al reducir la sobrecarga del servidor. Bcrypt es un algoritmo de cifrado robusto que permite almacenar contraseñas de manera segura, minimizando el riesgo ante ataques de fuerza bruta y asegurando la integridad de los datos de los usuarios.

Estas tecnologías han sido seleccionadas considerando el enfoque de Clean Architecture, que exige una clara separación de responsabilidades y permite desacoplar la lógica de negocio de la infraestructura técnica. Además, son herramientas ampliamente utilizadas y respaldadas por comunidades activas, lo que garantiza su estabilidad, escalabilidad y facilidad de mantenimiento en el contexto del proyecto.



3.2 Frontend: Aplicación en Angular

```

src/
├── app/
|   ├── auth/           # Módulo de autenticación
|   |   ├── login/      # Componente de inicio de sesión
|   |   └── register/   # Componente de registro
|   |
|   ├── core/          # Servicios y utilidades core
|   |   ├── guards/     # Guards de autenticación
|   |   ├── interceptors/ # Interceptores HTTP
|   |   ├── models/      # Interfaces y tipos
|   |   └── services/    # Servicios principales
|   |
|   └── features/       # Módulos de funcionalidad

```

```
| |   ├── checkins/      # Gestión de fichajes
| |   ├── incidents/    # Gestión de incidencias
| |   ├── users/        # Gestión de usuarios
| |   └── vacations/    # Gestión de vacaciones
|
| |
|   ├── shared/         # Componentes y utilidades compartidas
|   └── components/     # Componentes reutilizables
|
| |
|   ├── app.component.* # Componente raíz
|   ├── app.module.ts    # Módulo principal
|   ├── app-routing.module.ts # Configuración de rutas
|   └── app.config.ts    # Configuración de la aplicación
|
| ├── assets/           # Recursos estáticos
| ├── environments/     # Configuraciones por entorno
| └── styles/            # Estilos globales
```

Para el frontend, se aplican varios patrones de diseño que aseguran modularidad y facilidad de mantenimiento:

- *Arquitectura Modular:*

Se organiza la aplicación en diferentes módulos:

- Core Module: Para servicios singleton (autenticación, interceptores, manejo global de errores).
- Shared Module: Para componentes, directivas y pipes reutilizables.

- *Dependency Injection (DI):*

Angular inyecta dependencias de forma nativa, permitiendo reutilizar servicios y mantenerlos como singeltons, lo cual se alinea al patrón DI.

- *Container/Presentational Components:*

Se separa la lógica (container components) de la presentación (presentational components), favoreciendo la claridad y reutilización.

- *Observer Pattern:*

Utilizando RxJS para la gestión reactiva, los componentes pueden observar y reaccionar a los cambios en el estado, ofreciendo una experiencia interactiva y fluida.

Tecnologías del frontend:

- Lenguajes y Frameworks: Angular, TypeScript, HTML5, SCSS.
- Librerías de UI: Angular Material.
- Ruteo: Angular Router.
- Seguridad y Gestión de Sesiones: Angular Guards y almacenamiento de JWT en el localStorage.

Para el frontend de este proyecto se eligieron Angular, TypeScript, HTML5, SCSS, Angular Material, Angular Router y Angular Guards (junto con el almacenamiento de JWT en el localStorage) por las siguientes razones³:

- Angular y Angular CLI:

Angular es un framework robusto y escalable para construir aplicaciones de una sola página (SPA). Su arquitectura modular, basada en componentes, permite organizar el código de forma clara y reutilizable, lo que es ideal para proyectos con crecimiento potencial, como este. Además, Angular CLI facilita la generación de código, la configuración inicial y la integración de buenas prácticas de desarrollo.

- TypeScript:

La elección de TypeScript se debe a que añade tipado estático al código JavaScript, lo que reduce errores en tiempo de compilación y mejora la mantenibilidad del proyecto. Esto resulta especialmente útil en aplicaciones complejas, pues facilita el trabajo en equipo y el escalado del código.

- HTML5 y SCSS:

Utilizar HTML5 garantiza que la aplicación cumpla con los estándares modernos para la estructura y semántica del contenido. SCSS permite escribir estilos más organizados y mantenibles con la posibilidad de usar variables, anidamientos y mixins, lo que acelera el desarrollo de interfaces atractivas y responsivas.

- Angular Material:

³ Generado por ChatGPT

Se seleccionó Angular Material porque ofrece una amplia gama de componentes UI preconstruidos que siguen las guías de diseño Material Design. Esto no solo acelera el desarrollo, sino que asegura una interfaz consistente, moderna y accesible, elementos esenciales para una buena experiencia de usuario.

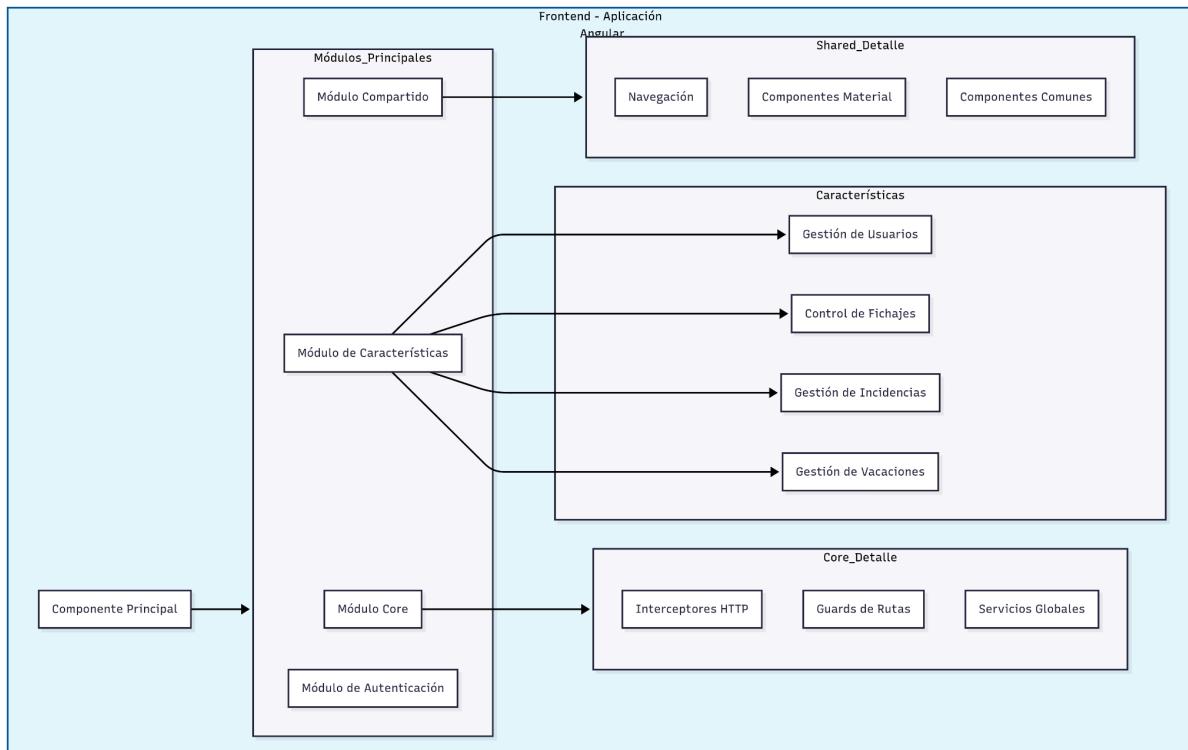
- Angular Router:

Angular Router permite la gestión eficiente de la navegación interna de la aplicación, facilitando la implementación de rutas y el lazy loading de módulos. Esto es clave para mantener la aplicación ligera y garantizar tiempos de carga óptimos, aspectos importantes en un entorno empresarial.

- Angular Guards y gestión de sesiones con JWT en localStorage:

La implementación de Angular Guards ayuda a proteger las rutas y asegurar que solo los usuarios autenticados puedan acceder a áreas sensibles de la aplicación. El uso de JWT almacenado en el localStorage, junto con los guards, proporciona una solución segura y sencilla para la gestión de sesiones, evitando accesos no autorizados y manteniendo la integridad de la información de los usuarios.

Estas tecnologías fueron escogidas sobre otras alternativas por su madurez, la amplia comunidad de soporte y su capacidad para integrarse de manera efectiva en una solución escalable, segura y profesional como la que se plantea en este proyecto. Cada una de estas elecciones responde a la necesidad de garantizar un desarrollo ágil, un mantenimiento sencillo y una experiencia de usuario de alta calidad, elementos clave para la gestión avanzada del tiempo laboral en el entorno empresarial.



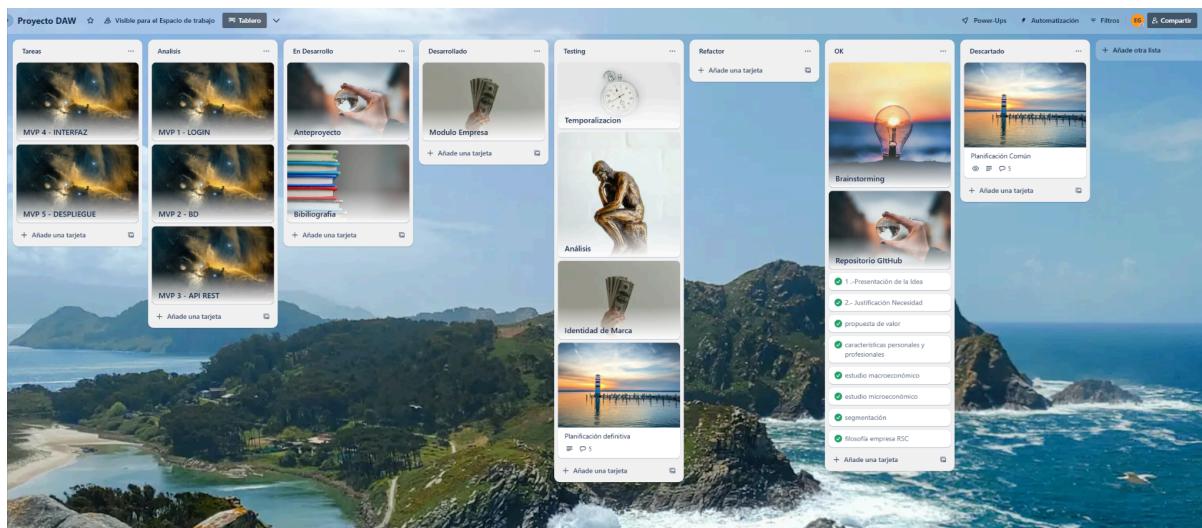
4. Plan de MVP y Cronograma

Planificación

El proyecto estará dividido en pequeñas tareas dentro de cada MVP para facilitar su desarrollo y seguimiento. Se utilizará una herramienta como Trello para gestionar el avance del proyecto y la asignación de tareas. Trabajaré de manera iterativa, con entregas parciales que se ajustarán a las necesidades de entrega.



<https://trello.com/invite/b/67d7f1d69b48dc409709deac/ATTlf279e1bd33704a696de5bc815e00a440D2669D40/proyecto-daw>



Dedicación semanal prevista: 5-15 horas/semana

Estimación horas de trabajo⁴:

- Análisis y diseño: 30–40 horas.
- Desarrollo del Backend (REST server con Clean Architecture):
 - Implementación del módulo de autenticación, conexión a la base de datos y endpoints CRUD: 120–150 horas.
- Desarrollo del Frontend (Angular):
 - Creación de la interfaz de usuario, componentes (login, registro, dashboard) e integración con la API: 80–100 horas.
- Integración, pruebas y despliegue: 30–40 horas.

Esto nos da un rango aproximado de 260 a 330 horas en total.

Se han definido **cinco MVP⁵** para estructurar el desarrollo del proyecto y facilitar el seguimiento de tareas:

MVP 1: Módulo de Login y Autenticación

- Objetivo: Permitir el registro e inicio de sesión mediante JWT y cifrado de contraseñas con Bcrypt.
- Tecnologías: Angular, Node.js, Express.js, MongoDB, JWT, Bcrypt.

⁴ Estimación calculada por IA ([ChatGPT 03-mini-high](#)).

⁵ Estructurado en 5 productos mínimos viables con ayuda de ChatGPT.

- Tareas:
 - Crear el modelo de usuario en MongoDB.
 - Configurar Express.js para gestionar autenticación.
 - Crear rutas API (`/register`, `/login`, `/me`).
 - Implementar JWT y cifrado con Bcrypt.
 - Desarrollar formularios en Angular para login y registro.
 - Configurar Angular Guards para proteger rutas.
- Resultado: Un usuario se registra, inicia sesión y su sesión se mantiene activa mediante JWT.

MVP 2: Creación de Base de Datos con Usuarios de Prueba

- Objetivo: Contar con una base de datos inicial poblada con usuarios ficticios para las pruebas.
- Tecnologías: MongoDB, Mongoose, Faker.js, Postman.
- Tareas:
 - Configurar la conexión a MongoDB desde Express.js.
 - Definir el esquema de usuario en Mongoose (nombre, email, contraseña, rol).
 - Crear un script usando Faker.js para poblar la base de datos.
 - Probar consultas básicas y gestionar errores.
- Resultado: Base de datos funcional con mínimo 10 usuarios de prueba y API REST para consulta de usuarios en `/api/users`.

MVP 3: Creación de la API REST Base

- Objetivo: Definir la estructura del backend y exponer los endpoints básicos.
- Tecnologías: Node.js, Express.js, MongoDB, Postman.
- Tareas:
 - Crear middleware para logs y errores (morgan, cors, dotenv).
 - Definir rutas base (`/users`, `/auth`, etc.).
 - Implementar operaciones CRUD para usuarios y fichajes.
 - Probar y documentar endpoints con Postman.

- Resultado: API funcional con rutas `/users` y `/auth`, y operaciones seguras con JWT.

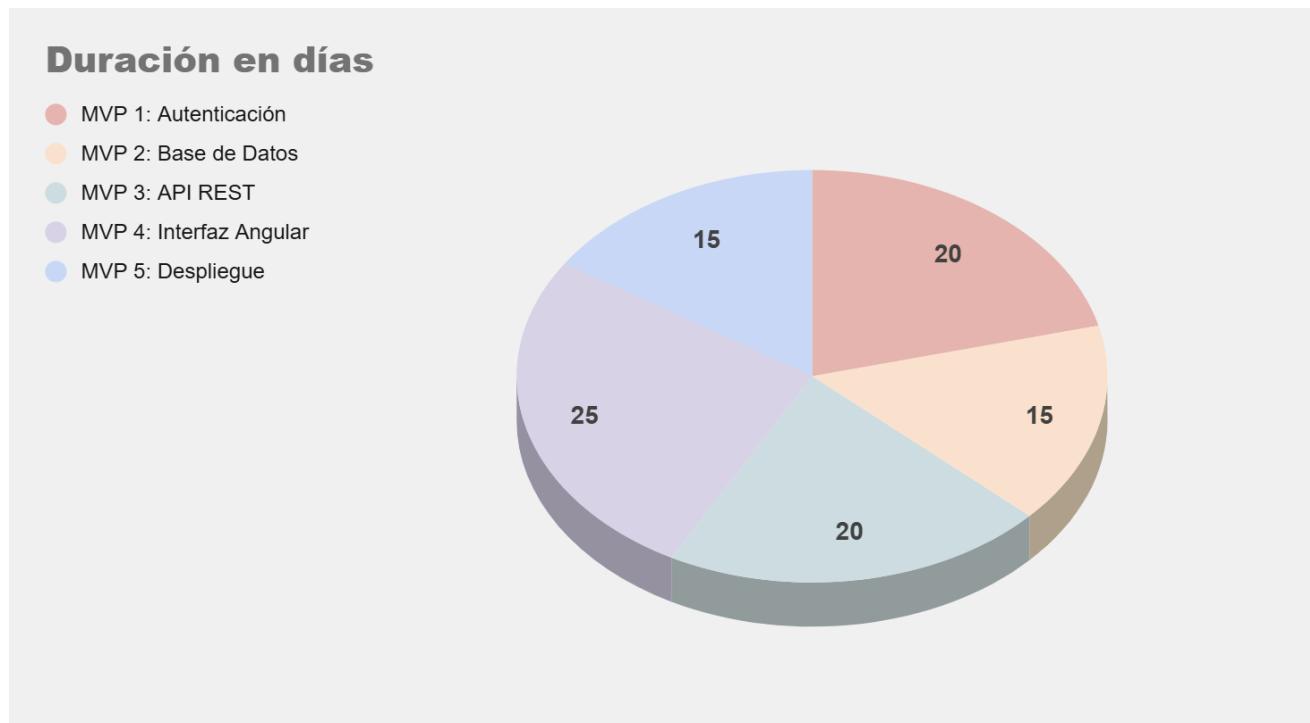
MVP 4: Creación de la Interfaz Base en Angular

- Objetivo: Desarrollar la estructura inicial de Angular con navegación, login y registro.
- Tecnologías: Angular, Angular Router, Angular Material.
- Tareas:
 - Inicializar el proyecto Angular.
 - Configurar rutas (`/login`, `/register`, `/dashboard`).
 - Desarrollar componentes para login y registro.
 - Conectar con la API REST para la autenticación.
 - Gestionar el token JWT y proteger rutas con Angular Guards.
- Resultado: Interfaz funcional que permite autenticación y acceso a un panel (dashboard) protegido.

MVP 5: Despliegue en Entorno de Desarrollo

- Objetivo: Validar el sistema en un entorno realista.
- Tecnologías: Docker, Vercel/Render, MongoDB Atlas.
- Tareas:
 - Configurar Dockerfile y docker-compose para backend y base de datos.
 - Desplegar el backend en Render.
 - Configurar y conectar con MongoDB Atlas.
 - Desplegar el frontend en Vercel.
 - Realizar pruebas completas del flujo de usuario.
- Resultado: Sistema completamente desplegado con backend, frontend y base de datos remota operativos.

Cronograma y Gestión del tiempo



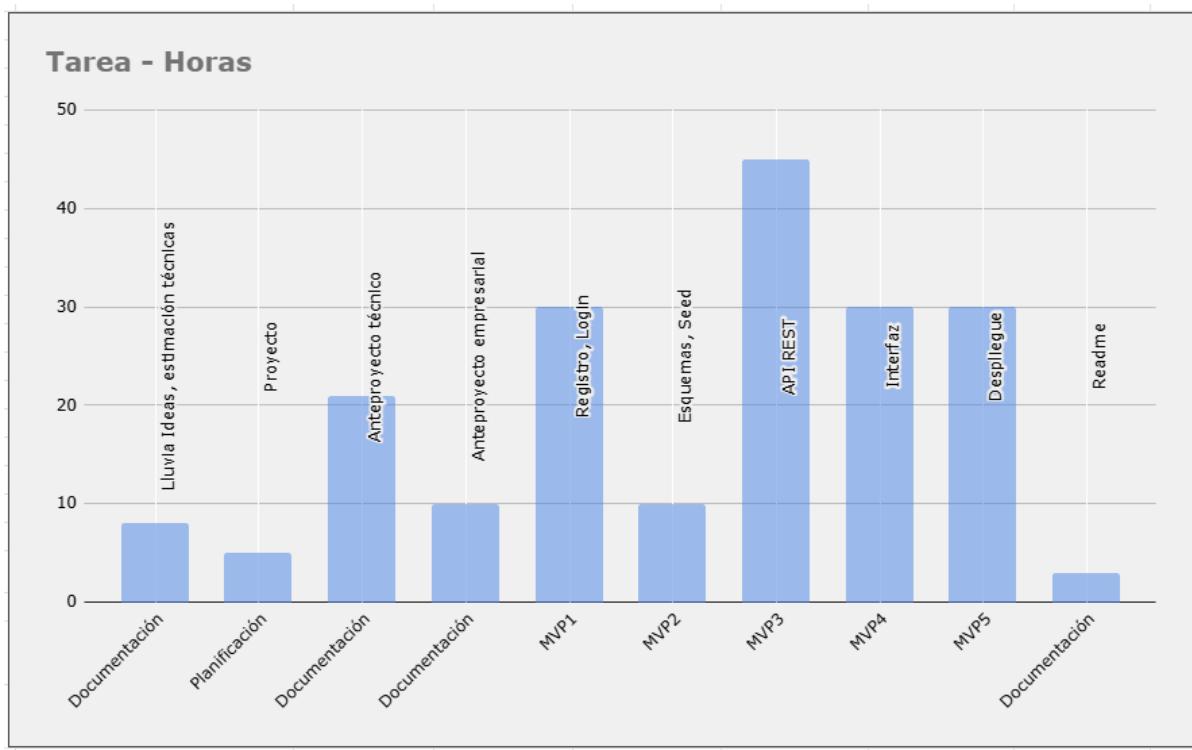
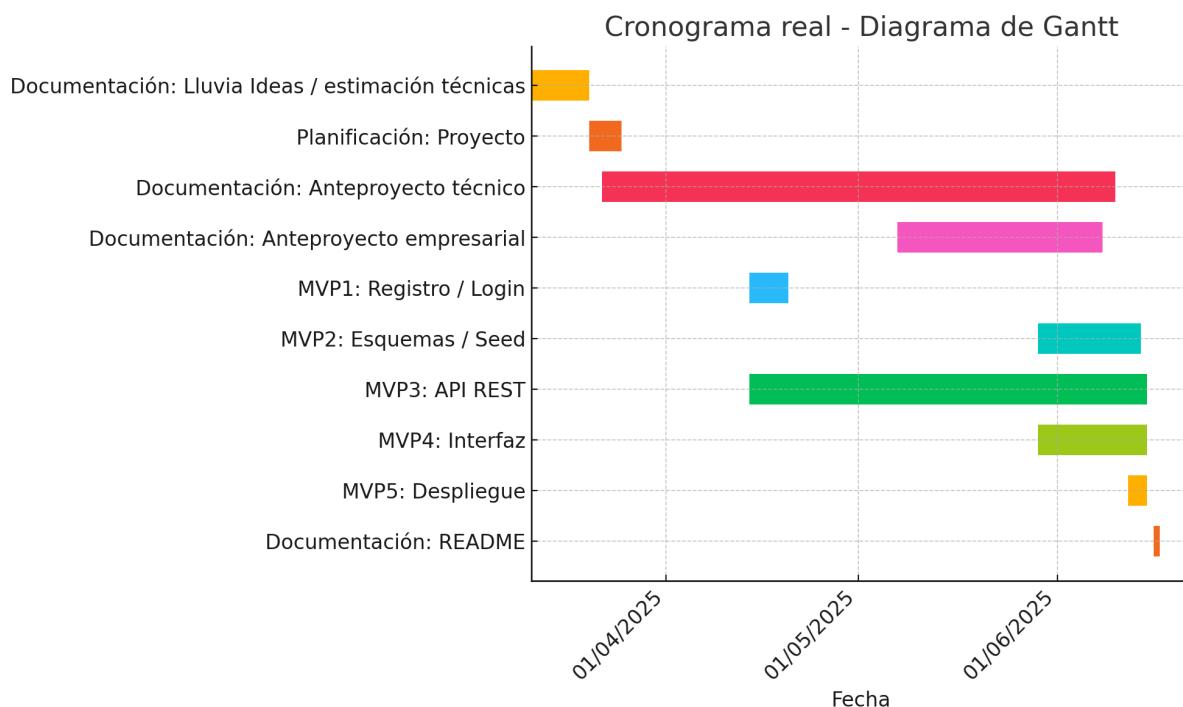
MVP	01/05/2025	21/05/2025	05/06/2025	25/06/2025	20/07/2025
MVP 1: Autenticación					
MVP 2: Base de Datos					
MVP 3: API REST					
MVP 4: Interfaz Angular					
MVP 5: Despliegue					

Estimación de esfuerzo por fase (MVP)

MVP	Alcance principal	Horas estimadas [†]	Semanas del cronograma*	h/semana necesarias para cumplir el plan
MVP 1 – Autenticación	Registro, login JWT, guards	50 – 60 h (≈ 55 h)	3	~18 h/sem
MVP 2 – Base de datos de prueba	Esquema Usuario, seed Faker	25 – 35 h (≈ 30 h)	2,5	~12 h/sem
MVP 3 – API REST	CRUD usuarios + fichajes, middleware	80 – 100 h (≈ 90 h)	3	~30 h/sem
MVP 4 – Interfaz Angular	Routing, componentes, dashboard	70 – 90 h (≈ 80 h)	3,5	~23 h/sem
MVP 5 – Despliegue	Docker, CI/CD, pruebas end-to-end	35 – 45 h (≈ 40 h)	2	~20 h/sem

Cronograma real

Tarea	Alcance	Horas	Fecha Inicio	Fecha Fin
Documentación	Lluvia Ideas, estimación técnicas	8	11/03/2025	20/03/2025
Planificación	Proyecto	5	20/03/2025	25/03/2025
Documentación	Anteproyecto técnico	21	22/03/2025	10/06/2025
Documentación	Anteproyecto empresarial	10	07/05/2025	08/06/2025
MVP1	Registro, Login	30	14/04/25	20/04/2025
MVP2	Esquemas, Seed	10	29/05/2025	14/06/2025
MVP3	API REST	45	14/04/25	15/06/2025
MVP4	Interfaz	40	29/05/2025	19/06/2025
MVP5	Despliegue	30	12/06/2025	15/06/2025
Documentación	Readme	4	16/06/2025	20/06/2025



5. Alcance y limitaciones.

Alcance

- Implementación de todos los módulos definidos en cada MVP.
- Creación de la solución completa para el registro de fichajes, autenticación y administración de usuarios.
- Despliegue en un entorno realista para validar el flujo de usuario completo.

Limitaciones

- La primera versión se centrará en funcionalidades esenciales; integraciones con sistemas de nómina/ERP se programarán en fases futuras.
- Se limitará el alcance a la gestión de fichajes y autenticación en la primera iteración.

Requerimientos

- Funcionales: Registro de usuarios, autenticación, administración de datos, interfaz de usuario para login y rutas protegidas.
 - No funcionales: Seguridad (JWT, Bcrypt), escalabilidad, accesibilidad y compatibilidad con múltiples dispositivos.
-

6. Recursos e infraestructura: recursos materiales y humanos

Estudio de Mercado

Macroentorno

Factor	Análise	Fonte
Económico	El mercado español de software de control horario creció un 15% anual debido a la obligatoriedad de registro de jornada y a la digitalización post-COVID.	softwaredoit.es
Político-Legal	El Real Decreto-ley 8/2019 obliga al registro diario de la jornada en todas las empresas españolas desde mayo de 2019.	boe.es
Social	En Galicia, el 65 % de las PYMEs reconoce dificultades para gestionar manualmente los horarios, buscando herramientas digitales para aumentar la satisfacción y la retención del talento.	sbssoftware.es
Tecnológico	Alta adopción de soluciones SaaS en Recursos Humanos; frameworks como Angular y Symfony son estándar para desarrollar aplicaciones escalables.	taclia.com

Microentorno

Competencia				
Competidor	Funcionalidades clave	Prezo / Monetización	Diferenciación principal	Fuente
SBS Presencia (Galicia)	Control horario, informes, gestión de permisos	Desde €3/mes/usuario	Enfocado a la administración pública y PYMES gallegas	sbssoftware.es
Evolk Galicia	Control horario web, portal empleado, alertas	€4,50/mes/usuario	Desarrollado en Galicia con soporte local	evolgali.cia.es
HRLOG (España)	Fichaje, vacaciones, informes en tiempo real	Freemium (plan pro €5/mes)	Interface intuitiva y rápido onboarding	hrlog.es
Sesame HR (Valencia)	Control horario, gestión de turnos, RSC analytics	Desde €3,75/mes/usuario	Integración IA y foco en escalabilidad	es.wikipedia.org
Connecteam (Multinacional)	Fichaje móvil, proyectos, comunicación interna	Desde \$2,50/mes/usuario	Alto crecimiento, solución global	es.wikipedia.org

Cliente Potencial (Target)		
Atributo	Descripción	Fonte
Sector	Empresas tecnológicas y PYMEs industriales gallegas	sbssoftware.es
Tamaño	10–200 empleados	Estimación basada en competidores
Cargo decisor	RRHH, CFO, CEO	
Necesidades	Cumplir norma de registro, reducir errores, mejorar productividad	

Cliente Potencial (Target)		
Hábitos de compra	Descubrimiento vía buscadores, comparativas online, demos gratuitas	softwaredoit.es

Segmentación de Mercado		
Segmento	Características	Tamaño estimado
PYMEs gallegas	10–50 empleados, limitada digitalización	~12.000 empresas en Galicia
Empresas tecnológicas españolas	50–200 empleados, alta adopción SaaS	~2.500 empresas
Grandes corporaciones multinacionales	>200 empleados, busca integración global	~300 empresas

Análisis DAFO



Filosofía, Visión y RSC

Misión

Facilitar el fichaje y la gestión del tiempo laboral mediante una herramienta digital intuitiva, personalizable y accesible (WCAG), tanto para los empleados que fichan como para los administradores.

Visión (3–5 años)

Crear una solución eficaz, positiva y clara para el control del tiempo trabajado, fomentando la conciliación laboral gracias a herramientas intuitivas, agradables y fáciles de usar. Convertir a Janoo en la plataforma de referencia para PYMEs en Galicia y España, con el objetivo de extender su adopción internacionalmente.

Valores corporativos

- Honestidad
- Transparencia
- Diversidad e inclusión
- Responsabilidad social y ambiental

Responsabilidad Social Corporativa (RSC)		
ODS	Iniciativa	Indicador de éxito
ODS 3: Salud y bienestar	Incorporar recordatorios de pausas activas y fomentar hábitos saludables en el panel empleado	80% de usuarios usan recordatorios al menos 3 veces por semana
ODS 5: Igualdad de género	Garantizar diseño y contenidos no discriminatorios, promoviendo igualdad	Auditoría anual de contenidos con 0 incidencias de sesgo

Responsabilidad Social Corporativa (RSC)		
ODS 8: Trabajo decente y crecimiento económico	Ofrecer versión gratuita básica para pequeñas empresas y tarifas reducidas para emprendedores	50 % de clientes son PYMEs con menos de 50 empleados en el primer año
ODS 12: Consumo responsable	Eliminar registros en papel sustituyéndolos por panel digital accesible	Reducción del 100 % del uso de papel en el registro horario de empresas usuarias
ODS 17: Alianzas para los objetivos	Garantizar accesibilidad total siguiendo los estándares WCAG 2.1	Cumplimiento del 100 % de los criterios WCAG verificado por auditoría externa anual

Modelo Financiero Inicial

Inversión inicial

La inversión inicial es prácticamente nula. El desarrollo lo lleva a cabo una sola persona desde su despacho en casa, utilizando un equipo informático ya amortizado y herramientas de código abierto o gratuitas. Gracias a ello no es necesario desembolsar partidas adicionales en hardware, licencias ni infraestructura, pues todos los recursos requeridos están cubiertos por medios propios.

Costes de funcionamiento mensual

Los gastos mensuales se reducen a servicios domésticos proporcionales al uso profesional (electricidad e internet) y a un hosting básico para la aplicación en fase beta. Estas partidas suman aproximadamente 70 – 150 € al mes. No se incluye retribución del promotor ni servidores de alto rendimiento porque, al tratarse de un proyecto de carácter educativo en pleno desarrollo, la prioridad es probar la viabilidad funcional antes de escalar recursos.

Estimación de Ingresos Futuros para Comercialización

Para un escenario futuro, donde la aplicación se convierte en un producto vendible, se puede proyectar un modelo que incluya:

- Modelo de monetización:
 1. Suscripción Premium: Ejemplo: 10 €/mes/usuario para funciones avanzadas.
 2. Versión Freemium: Funciones básicas gratuitas; ingresos a través de actualización a premium.
 3. Servicios Adicionales: Integración con otros sistemas o módulos extras (por transacción o tarifa mensual).
- Estimación de Volúmenes:

Suponiendo que en el primer año de comercialización se logra llegar a 200 usuarios activos y con la actualización al modelo premium, se espera una tasa de conversión del 20–30%, lo que daría entre 40 y 60 usuarios premium. Ingreso mensual estimado:

- 40 usuarios premium x 10 € = 400 € mensuales (escenario conservador)
 - 60 usuarios premium x 10 € = 600 € mensuales (escenario optimista)
- Proyección de crecimiento:

Se prevé que con la incorporación de módulos adicionales (ej.: imputación de horas a proyectos, integración con ERP, etc.), el precio y el número de usuarios puedan incrementarse gradualmente.

Factores a Considerar para el Precio Premium de 10 €/mes

- Funciones Avanzadas y Personalización:

Janoo se diseñaría desde el principio con una interfaz intuitiva, personalizable y cumpliendo las pautas WCAG, lo que puede traducirse en una experiencia de usuario superior y adaptada a las necesidades de cada empresa.

La implementación de módulos adicionales (como integración con sistemas ERP, imputación de horas a proyectos, análisis de datos avanzados) podría justificar un precio premium, ya que estas funcionalidades aportan un ahorro de tiempo y una mayor eficiencia en la gestión.

- Soporte y Seguridad:

Si se ofrece un servicio de soporte técnico especializado y garantías en materia de seguridad (muy valorados en las aplicaciones de control horario, especialmente ante la obligación legal del RDL 8/2019), esto puede trasladarse en un coste superior.

Empresas que manejan datos sensibles y requieren un alto nivel de seguridad están dispuestas a pagar un poco más por un servicio robusto y fiable.

- Segmentación del Mercado:

En el caso de dirigirse a PYMEs que valoren la eficiencia y la posibilidad de personalizar sus herramientas de gestión del tiempo, un precio de 10 €/mes/usuario podría percibirse como una inversión en productividad.

Si el producto ofrece un claro diferencial en términos de usabilidad y escalabilidad, el segmento premium puede estar dispuesto a pagar un precio superior.

- Estrategia Freemium:

Se puede optar por un modelo freemium en el que la versión básica sea gratuita, permitiendo que las empresas prueben la herramienta sin compromiso. Una vez que los usuarios evaluen la funcionalidad, la actualización a una versión premium a 10 €/mes se justificaría por las características avanzadas y la integración de módulos adicionales.

De esta forma, el precio se adapta gradualmente a la percepción de valor del cliente.

Conclusión

Aunque 10 €/mes/usuario premium podría ser más alto que el coste de la competencia, este precio se puede justificar si se alcanza un valor añadido notable a través de:

1. Mayor personalización y usabilidad (cumplimiento de estándares WCAG, panel intuitivo, etc.).
2. Funcionalidades avanzadas que permitan una gestión integral del tiempo y recursos, optimizando la productividad en las empresas.

3. Soporte técnico y seguridad reforzada, imprescindibles en aplicaciones con implicaciones legales.

Posibilidad de realizar pruebas de mercado (un piloto o focus groups) para validar si los potenciales clientes perciben esta tarifa como competitiva en función del valor que aportan las funciones adicionales. Si la percepción es que es demasiado alto, se pueden ajustar las tarifas o establecer niveles de suscripción (por ejemplo, una versión premium a 7–8 €/mes con opción a módulos adicionales a medida).

Esta estrategia permite posicionar Janoo de manera competitiva, garantizando al mismo tiempo que el precio refleje la calidad y el valor añadido que aportará el producto en el futuro.

Modelo Financiero Futuro

Suponiendo que en una fase de comercialización se decida expandir el proyecto, se estima la contratación de 2 a 3 empleados en áreas clave (Desarrollo, Marketing y Soporte). A continuación, se incluye un ejemplo para tres puestos, uno para cada área:

A. Estimación de Salarios y Cargas Sociales

- Desarrollador/a
 - Salario neto estimado: 2.000 €/mes
 - Cargas sociales (aprox. 30%): 600 €/mes
 - Costo total: 2.600 €/mes
- Personal de Marketing y Soporte
 - Salario neto estimado: 1.500 €/mes (pueden diferir según experiencia y función específica)
 - Cargas sociales (aprox. 30%): 450 €/mes
 - Costo total: 1.950 €/mes

B. Coste de Material Informático para Nuevos Empleados

Costo estimado de cada equipo: 1.200 €

Amortización mensual: 1.200 € / 36 ≈ 33 €/mes por empleado

Coste total de amortización sería de: 33 € × 3 ≈ 100 €/mes (aproximadamente)

Resumen de Costes Futuros Mensuales (para 3 empleados)				
Concepto	Detalle	Costo unitario (€)	Cantidad	Total (€/mes)
Desarrollador/a	Salario + cargas	2.600	2	5.600
Marketing y Soporte	Salario + cargas	1.950	1	1.950
Material informático	Amortización (por equipo)	33	3	99

Estos cálculos ayudarán a definir un precio de suscripción que cubra los costes y permita la rentabilidad a largo plazo. Por ejemplo, si se considera la necesidad de repartir estos costes entre el número de usuarios premium, el precio medio por usuario deberá ajustarse para que el ingreso total mensual supere estos costes operativos, considerando también los márgenes de beneficio y reinversión.

Plan de Marketing

Estrategia de Lanzamiento (Fase Educativa y de Piloto)

- *Canales de Adquisición:*
 - Redes Sociales: LinkedIn, Twitter y foros de desarrolladores y recursos humanos.
 - Marketing de Contenidos: Creación de un blog y tutoriales en vídeo sobre gestión de tiempo laboral.
 - Comunidades y Grupos: Participar en eventos y comunidades online relacionadas con el control horario y RRHH.

- *Presupuesto Inicial (Fase Educativa):*
 - Inversión mínima (prácticamente cero, enfocándose en herramientas gratuitas y marketing orgánico).

- Estrategia basada en networking y difusión en medios especializados.
- *Cronograma:*
 - Mes 1 – 3: Desarrollo y lanzamiento en versión piloto (beta cerrada)
 - Mes 4 – 6: Recogida de feedback, optimización y difusión en redes sociales.
 - Mes 7 – 12: Estrategia de captación de usuarios mediante colaboraciones y contenidos especializados.

Estrategia para Comercialización (Futuro)

- *Canales de Adquisición Remunerados:*
 - SEM (Google Ads, LinkedIn Ads) con un presupuesto inicial de unos 200 – 300 €/mes.
 - Colaboraciones con influencers y expertos en RRHH.
 - Participación en ferias y eventos sectoriales.
- *Indicadores Clave:*
 - CTR, tasa de conversión, coste por adquisición y fidelización de usuarios.

Obligaciones legales, fiscales y Prevención de riesgos laborales

Marco normativo de referencia

- Fiscalidad: Ley 58/2003, General Tributaria; Ley 37/1992 del IVA; Ley 27/2014, del Impuesto sobre Sociedades; Real Decreto 1065/2007 (obligaciones censales, modelos 036/037, 303, 111, 200).

- Laboral: Estatuto de los Trabajadores (RDL 2/2015); Real Decreto-ley 8/2019 (registro diario de jornada); Real Decreto 901/2020 (planes de igualdad cuando proceda).
- Seguridad Social: Real Decreto 84/1996 (afiliación y cotización); alta en RETA o encuadramiento de plantilla en Régimen General.
- Prevención de Riesgos Laborales (PRL): Ley 31/1995; Real Decreto 488/1997 (pantallas de visualización); Ley 10/2021 de trabajo a distancia.
- Protección de datos: Reglamento (UE) 2016/679 (GDPR) y Ley Orgánica 3/2018 (LOPD-GDD).
- Propiedad intelectual e industrial: Ley 21/2014 (reforma del TRLPI), Ley 17/2001 (marcas), Real Decreto 281/2003 (Registro de Software).

Obligaciones fiscales y laborales

Obligaciones comunes (autónomo y SL)

- Alta censal ante la AEAT mediante los modelos 036/037 e inscripción en el epígrafe IAE 765/763 antes del inicio de la actividad.
- Cotización a la Seguridad Social: ingreso mensual de la cuota (RETA) o liquidaciones RLC/RNT en el Régimen General.
- Declaraciones trimestrales de IVA (modelo 303) cuando se realicen operaciones sujetas.
- Ingreso de retenciones de IRPF (modelo 111) cuando existan nóminas o pagos a profesionales.
- Registro de jornada diario cuando se contrate personal (RD-ley 8/2019).
- Cumplimiento del RGPD-LOPD: registro de actividades de tratamiento y cláusulas informativas en web, contratos y facturas.

Obligaciones exclusivas del autónomo

- Alta en el Régimen Especial de Trabajadores Autónomos (RETA) y domiciliación de la cuota.
- Pago fraccionado del IRPF propio mediante modelo 130, con periodicidad trimestral.

- Comunicación de variaciones de datos o bases de cotización ante la TGSS.

Obligaciones exclusivas de la sociedad limitada

- Otorgamiento de la escritura de constitución, inscripción en el Registro Mercantil y alta del Código de Cuenta de Cotización (CCC).
- Presentación anual del Impuesto sobre Sociedades (modelo 200).
- Formulación, aprobación y depósito de las cuentas anuales en el Registro Mercantil dentro de los seis meses posteriores al cierre.
- Legalización de libros oficiales (contables y societarios) y libro-registro de socios por sede electrónica del RM cada ejercicio.

Permisos y autorizaciones específicas.

- Registro de marca y dominio (OEPM + ESNIC) antes del lanzamiento público.
- Inscripción de ficheros de datos o registro interno de actividades de tratamiento según GDPR.
- Licencias de software de terceros (MIT, GPL, comerciales) documentadas y compatibles.
- Registro de la propiedad intelectual del código fuente (opcional pero recomendable para acreditación de autoría).
- Declaración responsable de cumplimiento de la LSSI-CE para servicios de la sociedad de la información.

Plan de Prevención de Riesgos Laborales

Objetivo: garantizar la seguridad y salud tanto en oficina como en teletrabajo.

a) Evaluación de riesgos

- Ergonomía: posturas, altura de pantalla, iluminación, pausas activas.
- Riesgos eléctricos: revisión periódica de fuentes de alimentación y SAI.
- Riesgos psicosociales: carga mental y desconexión digital.

- Ciberseguridad (extendido a PRL por posible daño patrimonial): antivirus, MFA y copias de seguridad.

b) Medidas preventivas

- Puestos certificados ergonómicamente (sillas clase A, reposapiés, monitores ≥24").
- Formación inicial y anual en PRL + ciberhigiene (phishing, contraseñas, RGPD).
- Implantación de política de pausas: 5 min cada 55 min de pantalla.
- Reconocimientos médicos voluntarios y adaptaciones post-lesión.

c) Organización y recursos

- Servicio de Prevención Ajeno (SPA) contratado: Prevengo S.L.
- Delegado/a de Prevención: fundadora.
- Presupuesto anual PRL: 1500 € (equipos ergonómicos y formación).

d) Seguimiento

- Auditoría interna semestral.
- Indicadores: porcentaje de incidencias ergonómicas resueltas, tiempo medio de corrección.

Calendario de cumplimiento de las obligaciones legales		
Mes	Acción clave	KPI/Evidencia
M-1	Alta censal, RETA/SL, plan PRL inicial	Altas registradas
M-1	Registro de marca y dominio	Nº registros
M	Registro horario operativo	100 % fichajes diarios
M+1	Evaluación de riesgos teletrabajo	Informe firmado
Trimestral	IVA 303 + IRPF 111	Modelos presentados
Semestral	Auditoría PRL interna	Informe auditoría
Anual	IS 200 + cuentas anuales	Justificante AEAT + Registro

Financiamiento

Fase Actual

- *Capital propio:*
 - Se utiliza el equipamiento ya adquirido y recursos personales, sin inversión extra.
 - Costes mensuales mínimos (70 – 150 €).

Fase de Comercialización (Futura)

- *Necesidad de financiación externa:*
 - Se estima que, para expandir el equipo y potenciar el marketing, se podría necesitar una inversión adicional de entre 15.000 y 30.000 € durante el primer año de comercialización.
- *Fuentes previstas:*
 - Préstamos Bancarios: Con condiciones favorables para emprendedores.
 - Subvenciones: Convocatorias públicas a nivel autonómico o nacional, como el Kit Digital o ayudas para el emprendimiento tecnológico. *Las ayudas y subvenciones detectadas se detallan en 3.9.3.
 - Inversión Privada: Posibles inversores ángeles interesados en tecnología educativa y soluciones de RRHH.

Ayudas y subvenciones públicas para la digitalización

Programas nacionales

- Kit Digital / Acelera Pyme (Red.es)
Qué financia: bonos de 2 000 € a 29 000 € para soluciones SaaS (control horario incluida).
Quién puede solicitar: autónomos y PYMEs ≤ 50 trabajadores con test de «Madurez digital».
Estado 2025: tercera convocatoria segmento III (0–2 emp.) abierta hasta

31 dic 2025 o agotarse fondos.

Pasos:

1. registrarse en Acelera Pyme,
2. test diagnóstico,
3. escoger agente digitalizador y firmar "Acuerdo de prestación".

- ENISA Emprendedoras Digitales 2025

Tipo: préstamo participativo 25 000 € – 1,5 M€; hasta 9 años.

Requisitos: empresa < 3 años, capital social ≥ 50 % mujeres.

Interés: euríbor + 1,5 %–3 % + tramo variable según beneficios.

Ventaja: no exige garantías personales.

- CDTI-NEOTEC 2025

Subvención: hasta 400 000 € (70 % del presupuesto) para empresas de base tecnológica < 3 años.

Elegibilidad: plan de negocio tecnológico, TRL ≥ 6.

Plazo: convocatoria prevista julio-septiembre 2025.

Programas autonómicos (Galicia)

- IGAPE – "Dixitalización Industria 4.0"

Ayuda: 40 %–50 % a fondo perdido, tope 150 000 €.

Para: PYMEs con sede en Galicia que implanten software de gestión y automatización.

Plazo 2025: abierto del 15 jun al 30 sep 2025.

- GAIN – Conecta HUBs

Objeto: proyectos colaborativos en IA, big data y cloud; rango 50 000 € – 250 000 €, subvención 60 %.

Condición: consorcio de al menos 2 empresas gallegas + un organismo de investigación.

Periodo: pre-propuestas hasta 5 oct 2025.

Programas europeos relevantes

- EIC Accelerator (Horizon Europe)

Financiación mixta: subvención hasta 2,5 M€ + equity hasta 15 M€.

Enfoque: deep-tech con alta escalabilidad (control horario con IA/predicción).

Proceso: presentación "short application" (pitch-deck + vídeo) → invitación a "full proposal" → entrevista en Bruselas.

Possible estrategia a seguir⁶:

1. Fase 0 – diagnóstico
Obtener certificado Pyme y test de madurez digital (Acelera Pyme).
2. Fase 1 – corto plazo (0-6 meses)
Tramitar Kit Digital segmento III para financiar la primera licencia SaaS y campaña piloto.
Preparar memoria para IGAPE «Digitalización 4.0» cubriendo backend cloud y seguridad.
3. Fase 2 – medio plazo (6-18 meses)
Constituir SL o reforzar capital para optar a ENISA Emprendedoras.
Presentar solicitud NEOTEC si se crea spin-off tecnológica.
4. Fase 3 – largo plazo (≥ 18 meses)
Explorar EIC Accelerator para expansión UE; requiere TRL ≥ 8 y primeras ventas.

Documentación básica común

- Plan de negocio validado (resumen ejecutivo + proyecciones financieras a 3 años).
- Memoria técnica: arquitectura SaaS, TRL*, impacto y escalabilidad.
- Certificados de estar al corriente con AEAT y TGSS.
- Informe de auto-evaluación digital (Acelera Pyme) y de huella de carbono si procede.

Technology Readiness Level*		
Nivel	Descripción breve	Situación típica
TRL 1	Principios básicos observados	Idea en papel
TRL 2	Formulación del concepto	Hipótesis y bocetos
TRL 3	Prueba de concepto experimental	Prototipo de laboratorio inicial
TRL 4	Validación de componente en laboratorio	Módulo aislado funcionando
TRL 5	Validación en entorno relevante	Demo parcial en entorno simulado

⁶ Estrategia propuesta por ChatGPT

Technology Readiness Level*		
TRL 6	Prototipo en entorno relevante	Sistema piloto casi completo
TRL 7	Demostración en entorno operativo	Piloto real con usuarios limitados
TRL 8	Sistema completo y calificado	Producto final listo para producción inicial
TRL 9	Sistema probado en operación real	Producto comercial con clientes

*NOTA : TRL son las siglas de Technology Readiness Level (Nivel de Madurez Tecnológica).

Es una escala estándar de 9 peldaños que usa la Comisión Europea para medir lo "maduro" que está un producto o tecnología.

7. Recursos e infraestructura: recursos en el área de desarrollo

Hardware y Software:

- Ordenador con entorno de desarrollo (Visual Studio Code u otro IDE).
 - Finalidad: Es la herramienta base para la codificación, depuración, pruebas y mantenimiento de la aplicación.
 - Licencia/Coste: Visual Studio Code es de código abierto y gratuito; otros IDEs pueden tener versiones gratuitas, gratuitas para estudiantes o de pago.
 - Fase de uso: Se utilizará durante todo el ciclo del proyecto.

- Herramientas:

- Node.js
 - Finalidad: Plataforma para desarrollar el backend (REST server) y ejecutar JavaScript en el servidor.

- Licencia/Coste: Gratuito, con licencia MIT.
- Fase de uso: Se emplea durante las fases de desarrollo y pruebas del REST server.
- Angular CLI
 - Finalidad: Herramienta para inicializar, desarrollar y mantener la aplicación frontend en Angular.
 - Licencia/Coste: Gratuito, con licencia MIT.
 - Fase de uso: Se utiliza desde el inicio del desarrollo frontend, así como durante las actualizaciones y despliegue.
- MongoDB (usando Mongoose)
 - Finalidad: Sistema de base de datos NoSQL para almacenar fichajes, horarios, empleados, días festivos y vacaciones.
 - Licencia/Coste: La edición Community es gratuita; MongoDB Atlas ofrece planes gratuitos para entornos de desarrollo y planes de pago para producción.
 - Fase de uso: Principalmente en fases de desarrollo y pruebas, y en producción mediante MongoDB Atlas para asegurar escalabilidad y alta disponibilidad.
- Git y GitHub.
 - Finalidad: Control de versiones y alojamiento de repositorios de código, facilitando la colaboración y el seguimiento de cambios.
 - Licencia/Coste: Git es software libre; GitHub ofrece opciones gratuitas (para proyectos públicos y ciertos privados) y planes de pago si se requieren funciones avanzadas.
 - Fase de uso: Se emplean durante todo el proyecto, desde el desarrollo hasta la integración y despliegue.

Plataformas de Despliegue:

- Docker
 - Finalidad: Contenerización de la aplicación para garantizar entornos de ejecución consistentes y facilitar el despliegue.

- Licencia/Coste: Gratuito y de código abierto bajo la licencia Apache 2.0.
- Fase de uso: Se utiliza en la fase de integración y despliegue para empaquetar y ejecutar el backend (y, opcionalmente, partes del frontend).
- Render / Vercel
 - Finalidad: Plataformas en la nube para el despliegue y alojamiento de la aplicación, tanto del backend como del frontend.
 - Licencia/Coste: Ambas plataformas disponen de planes gratuitos para aplicaciones de desarrollo y prueba; se pueden migrar a planes de pago conforme se requiera mayor escalabilidad y rendimiento en producción.
 - Fase de uso: Se usan en la fase de despliegue y validación en entornos realistas (producción).
- MongoDB Atlas.
 - Finalidad: Servicio en la nube para gestionar la base de datos MongoDB con alta disponibilidad y escalabilidad.
 - Licencia/Coste: Ofrece opciones gratuitas para desarrollo y planes de pago para entornos de producción con mayor demanda de recursos.
 - Fase de uso: Se implementa en la fase de producción para garantizar que la base de datos soporte el crecimiento y la carga de la aplicación.

Herramientas de Prueba y Gestión:

- Postman
 - Finalidad: Herramienta para el testeo y validación de la API REST, permitiendo enviar y recibir peticiones HTTP.
 - Licencia/Coste: Cuenta con una versión gratuita; versiones de equipo están disponibles con coste adicional según necesidades.
 - Fase de uso: Se utiliza durante la fase de desarrollo y pruebas para validar el correcto funcionamiento del REST server.
- Trello
 - Finalidad: Gestión de proyectos y seguimiento de tareas, facilitando la asignación y control del avance del proyecto.

- Licencia/Coste: Ofrece una versión gratuita con funcionalidad completa para proyectos pequeños y medianos; existen planes de pago para funciones avanzadas.
 - Fase de uso: Se utiliza en todo el ciclo del proyecto para planificar, monitorizar y coordinar las tareas.
-
- Otras herramientas colaborativas (Google Drive, Slack,..)
 - Finalidad: Facilitan la comunicación, coordinación y colaboración entre los miembros del equipo.
 - Licencia/Coste: Varias de estas herramientas ofrecen planes gratuitos, con opciones de pago que incluyen características adicionales.
 - Fase de uso: Se aplican a lo largo de todo el proyecto, especialmente en las fases de coordinación, desarrollo y seguimiento.

8. Despliegue

El despliegue de la aplicación es un elemento clave para garantizar su disponibilidad, escalabilidad y rendimiento en entornos reales.

Aspectos destacados del despliegue:

- **Contenerización con Docker:**

Se configurará un Dockerfile y un archivo docker-compose para empaquetar el backend (REST server) y la base de datos en contenedores. Esto permitirá garantizar la consistencia entre los entornos de desarrollo y producción sin necesidad de implementar funcionalidades avanzadas adicionales.

- **Plataformas en la Nube (Render / Vercel y MongoDB Atlas):**

Frontend: Se desplegará en Vercel, una plataforma optimizada para aplicaciones frontend que permite una entrega continua, escalabilidad automática y excelente integración con entornos de desarrollo modernos. <https://janoo.vercel.app>

Backend: Se desplegará en Render, lo que proporciona una solución sencilla para alojar el servidor Node.js, con capacidades de escalado automático y reinicios automáticos ante fallos. <https://janoo.onrender.com>

Base de Datos: Se utilizará MongoDB Atlas, una solución en la nube gestionada para bases de datos MongoDB, que permite alta disponibilidad, seguridad y backups automáticos sin necesidad de mantenimiento manual.

- **Pruebas de Integración:**

Antes del despliegue final, se llevarán a cabo pruebas de integración para asegurar que el flujo de autenticación y de registro de fichajes funcione correctamente en un entorno simulado de producción. Esta verificación se realizará mediante herramientas básicas de testeo, como Postman, para validar los endpoints del API.

Esta estrategia de despliegue se centra en cumplir de forma efectiva y realista los objetivos planteados para el MVP 5, asegurando que la aplicación se pueda poner en producción de manera estable y sin sobrecargar el proceso con mejoras o funcionalidades adicionales que excedan el alcance inmediato del proyecto

9. Análisis de Riesgos⁷

Integración y Compatibilidad

- Riesgo: Desacuerdos entre el frontend y el backend.
- Mitigación: Pruebas de integración continuas y documentación detallada de la API REST.

Seguridad

- Riesgo: Vulnerabilidades en autenticación y manejo de datos sensibles.
- Mitigación: Implementación robusta de JWT, cifrado con Bcrypt y validación exhaustiva de entradas.

⁷ Generado por chat GPT

Despliegue y Escalabilidad

- Riesgo: Fallas durante la puesta en producción.
 - Mitigación: Uso de Docker, pruebas en entornos similares a producción y monitorización continua.
-

10. Bibliografía⁸

- [1] ANGULAR, "Angular," [Online]. Available: <https://angular.io>.
- [2] ANGULAR MATERIAL, "Angular Material," [Online]. Available: <https://material.angular.io>.
- [3] NODE.JS, "Node.js Documentation," [Online]. Available: <https://nodejs.org/en/docs/>.
- [4] EXPRESS, "Express Documentation," [Online]. Available: <https://expressjs.com>.
- [5] MONGODB, "MongoDB Manual," [Online]. Available: <https://www.mongodb.com/docs/manual/>.
- [6] MONGOOSE, "Mongoose Documentation," [Online]. Available: <https://mongoosejs.com/docs/>.
- [7] JWT.IO, "JSON Web Tokens," [Online]. Available: <https://jwt.io>.
- [8] WCAG, "Web Content Accessibility Guidelines (WCAG) 2.0," W3C, 2008, [Online]. Available: <https://www.w3.org/TR/WCAG20>.
- [9] F. HERRERA, "NodeJS: RestServer con Clean Architecture," Online Course. [Online]. Available: <https://es.linkedin.com/learning/nodejs-restserver-con-clean-architecture>.
- [10] OPENAI, "ChatGPT," [Online]. Available: <https://chat.openai.com>.
- [11] STACK OVERFLOW, "Stack Overflow," [Online]. Available: <https://stackoverflow.com>.
- [12] GOOGLE, "Google Search," [Online]. Available: <https://www.google.com>.

⁸ Sistema IEEE.

Justificación de la elección de sistema de citación IEEE: La similitud de los sistemas de citación Harvard e IEEE con la ISO690. El estricto cumplimiento de la ISO 690 para recursos electrónicos requiere señalar las fechas de consulta lo que supondría una actualización recurrente de este apartado. Por ello, se opta por el sistema IEEE, que permite omitir las fechas de consulta sin perder rigor en la presentación de las referencias.

- [13] Gobierno de España, «Ley 31/1995, de 8 de noviembre, de Prevención de Riesgos Laborales», BOE, núm. 269, pp. 32590–32611, 10-nov-1995.
- [14] Comisión Europea, Technology Readiness Levels (TRL) – H2020 General Annex G, Bruselas, 2018. [Online]. Available:
https://ec.europa.eu/research/participants/data/ref/h2020/other/wp/2018-2020/annexes/h2020-wp1820-annex-g-trl_en.pdf
- [15] ISO, ISO 9001:2015 Quality Management Systems – Requirements. Ginebra: International Organization for Standardization, 2015.
- [16] CMMI Institute, CMMI Development v2.0, Model Overview. Pittsburgh, PA, USA, 2018.
- [17] Red.es, «Programa Kit Digital: Bases reguladoras y convocatorias», 2025. [Online]. Available: <https://www.acelerapyme.gob.es>
- [18] Instituto Galego de Promoción Económica (IGAPE), «Cheques Dixitalización Industria 4.0», Convocatoria 2025. [Online]. Available: <https://www.igape.gal>
- [19] Ministerio de Industria, Comercio y Turismo, «ENISA Emprendedoras Digitales», 2025. [Online]. Available: <https://www.enisa.es>
- [20] Ministerio de Ciencia e Innovación – CDTI, «Programa Neotec 2025», 2025. [Online]. Available: <https://www.cdti.es>
- [21] Softwaredoit, «Mercado español de software de control horario: crecimiento 15 % anual», Informe de Mercado, jul-2024. [Online]. Available:
<https://www.softwaredoit.es/blog/control-horario-mercado.html>
- [22] SBS Software, «Encuesta PYMEs gallegas sobre gestión de horarios», A Coruña, 2024. [Online]. Available:
<https://sbssoftware.es/estudio-pymes-galicia-registrar-jornada-laboral>
- [23] Taclia, «Tendencias SaaS RR.HH. 2024-2026», Barcelona, 2024. [En línea]. Disponible en: <https://taclia.com/es/blog/tendencias-saas-rrhh>
- [24] HRLOG, «Planes y precios HRLOG», 2025. [Online]. Available: <https://hrlog.es/precios>
- [25] Evolk Galicia, «Control horario y portal del empleado», 2025. [Online]. Available: <https://evolkgalicia.es>
- [26] Google Cloud, «SonarCloud Documentation», 2025. [Online]. Available: <https://docs.sonarcloud.io>

[29] GitHub, «GitHub Actions Documentation», 2025. [Online]. Available: <https://docs.github.com/actions>

[31] International Labour Organization, «Telework and Ergonomic Good Practices», Ginebra, 2022.

[32] G. Vermeer, «Estimating Defect Density in Agile Projects», IEEE Software, vol. 38, no. 2, pp. 82-88, mar-2021.

11. Contribución de la Inteligencia Artificial al proyecto

La inteligencia artificial representa una oportunidad estratégica para potenciar y optimizar los procesos de desarrollo. En este proyecto, la IA se integra de manera transversal mejorando la calidad y la eficiencia..

- Contribución en la redacción del anteproyecto:
 - ¿Qué herramienta de IA generativa usaste (nombre y versión)?
 - Chat GPT 4 → [Modelo GPT-4-turbo de OpenAI](#).
 - Chat GPT 3 → [03-mini-high](#)
 - ¿Para qué usaste la herramienta?
 - Chat GPT 3 → Estimación de horas de trabajo (etiquetado) y selección de las tecnologías idóneas para la idea definitiva.
 - Chat GPT 4 → Apoyo en la redacción del documento y en la comparativa de las ideas iniciales analizando pros y contras de la gestión del tiempo..
 - ¿Qué contenidos fueron generados por el agente? Etiquetalos.
Los contenidos generados 100% por la IA están etiquetados.
 - ¿Cómo has utilizado o cambiado la salida de la IA generativa?
Utilización de la salida como guía para la redacción del documento.

- Contribución en la Automatización y Asistencia en el Desarrollo:⁹

Herramientas basadas en IA, como asistentes de codificación (Copilot) y generación automática de documentación (ChatGPT), facilitan la redacción de código, la revisión de patrones de diseño y la optimización del proceso de testing. Permiten identificar errores de manera temprana y proponer mejoras, acelerando el ciclo de desarrollo y reduciendo la carga de tareas repetitivas.

12. Conclusiones y Observaciones¹⁰

La solución propuesta, basada en el desarrollo de un REST server mediante Clean Architecture y una aplicación frontend modular con Angular, permite una separación clara de responsabilidades y cumple con los estándares de calidad, escalabilidad y seguridad requeridos.

La propuesta está planificada en MVPs, lo que garantiza la posibilidad de implementación iterativa y la incorporación de futuras mejoras.

13. Líneas futuras

Plan de calidad y control del proyecto

Garantizar que el producto "Janoo" cumpla con los requisitos funcionales, legales y de experiencia de usuario mediante un sistema de mejora continua basado en evidencias y métricas objetivas que ayuden a su comercialización.

⁹ Generado por ChatGPT

¹⁰ Generado por ChatGPT

Indicadores de calidad (KPIs)			
Categoría	Indicador	Fórmula / Unidad	Valor objetivo
Código	Cobertura de tests unitarios	% líneas cubiertas	≥ 80 %
Código	Índice de "code smells" SonarQube	Nº/1 000 LOC	≤ 5
Código	Complejidad ciclomática media	Media por función	≤ 10
Procesos	Velocidad de sprint	Puntos completados / sprint	±10 % estim.
Procesos	% incidencias cerradas dentro de sprint	Cerradas / creadas	≥ 90 %
Producto	Bugs críticos en producción	Nº por versión	0
Producto	Performance back-end	peticiones/seg. @ P95	≥ 150
Producto	Accesibilidad (WCAG 2.1)	Puntos cumplidos	100 % AA
Cliente/ Usuario	NPS (Net Promoter Score) piloto	Escala -100...+100	≥ 40

Procedimiento de seguimiento

1. Integración continua (CI): pipeline GitHub Actions – ejecución de tests, lint y análisis SonarQube en cada *push*.
2. Revisión por pares (*pull request review*): mínimo 1 revisor externo al autor; checklist de estilo, seguridad y accesibilidad.

3. Sprint review & retro: revisión funcional quincenal con demo; recogida de métricas de velocidad y defectos.
4. Informe mensual de calidad: exporte automático de SonarQube y Test Coverage, consolidado por el QA lead y archivado en Confluence.
5. Auditoría semestral: simulación de hackeo (*penetration test*) + auditoría de accesibilidad externa (Ilunion Tec).

Gestión de incidencias y cambios

- Registro: todas las incidencias y *change requests* se crean en la columna «Backlog» de Jira.
- Clasificación: prioridad (bloqueante/alta/media/baja) y tipo (bug/feature/refactor/ops).
- Flujo Kanban:
 - *To Do* → *In Progress* → *Code Review* → *Testing* → *Done*.
- SLA internos:
 - Bloqueante: < 24 h.
 - Alta: < 48 h.
- Gestión de cambios mayores: evaluación de impacto (coste, riesgos, cronograma) + aprobación del *Product Owner*.

Herramientas y roles		
Rol	Responsabilidades clave	Herramientas
QA lead (promotora)	Mantener métricas, coordinar auditorías, aprobar <i>releases</i>	SonarQube, Jira, GitHub Actions
Dev	Escribir tests, solventar <i>code smells</i> , revisar <i>PRs</i>	Jest, Cypress, ESLint
PO / Cliente	Validar historias, priorizar backlog	Jira, Miro
Usuario beta	Probar versiones candidatas, cumplimentar encuestas	Encuesta SUS, Hotjar

Documentación y entregables

- **Plan de QA** (este capítulo) y checklist de revisión.
- **Registro de incidencias** exportable (.csv) y actas de *retrospective*.
- **Informes SonarQube** archivados en /docs/quality/YYYY-MM.
- **Informe de auditoría externa de accesibilidad.**

Cronograma QA (2025/2026)		
Fecha	Hito	Evidencia
17 jul 2025	CI inicial operativa	Pipeline verde en GitHub Actions
30 oct 2025	Cobertura tests ≥ 50 %	SonarQube report
20 ene 2026	Auditoría accesibilidad interna	Checklist AA firmado
30 feb 2026	Cobertura tests ≥ 80 %	SonarQube report
15 may 2026	Pen-test externo	Informe «OK» sin vulnerabilidades críticas
01 jun 2026	NPS piloto recopilado	Informe encuesta