Treball de Fi de Grau

Títol

Hoora

Autoria

David Casas Vilasís & Albert Marimón Plaza

Professorat tutor

Marcel Josephus Antonius Maas

Grau

Comunicació Audiovisual	
Periodisme	
Publicitat i Relacions Públiques	
Comunicació Interactiva	X
Comunicació de les Organitzacions	

Tipus de TFG

Projecte	X
Recerca	

Data

Del 26 al 29 de maig de 2026	X
24 de juliol de 2026	

Full resum del TFG

Títol del Treball Fi de Grau:

Català:	MEMÒRIA DEL PROJECTE: Hoora
Castellà:	MEMORIA DEL PROYECTO: Hoora
Anglès:	PROJECT REPORT: Hoora

Autoria:	David Casas Vilasís & Albert Marimón Plaza
Professorat tutor:	Marcel Josephus Antonius Maas

Curs:	2025/26	Grau:	Comunicació Audiovisual	
			Periodisme	
			Publicitat i Relacions Públiques	
			Comunicació Interactiva	X
			Comunicació de les Organitzacions	

Paraules clau (mínim 3)

Català:	**Intel·ligència Artificial (IA), Gestió del temps (*Time-tracking*), Visualització de dades (*Dashboards*), Metodologies Àgils (*Agile*), Disseny d'Experiència d'Usuari (UX/UI), Productivitat empresarial, Arquitectura de programari, Desenvolupament multiplataforma, Seguretat de dades, Programari com a Servei (SaaS)**
Castellà:	**Inteligencia Artificial (IA), Gestión del tiempo (Time-tracking), Visualización de datos (Dashboards), Metodologías Ágiles (Agile), Diseño de Experiencia de Usuario (UX/UI), Productividad empresarial, Arquitectura de software, Desarrollo multiplataforma, Seguridad de datos, Software como Servicio (SaaS).**

Anglès:	**Artificial Intelligence (AI), Time-tracking, Data visualization (Dashboards), Agile methodologies, User Experience/User Interface (UX/UI), Business productivity, Software architecture, Cross-platform development, Data security, Software as a Service (SaaS).**

Resum del Treball Fi de Grau (extensió màxima 100 paraules)

Català:	Aquest projecte consisteix en el disseny i desenvolupament d'una aplicació web de gestió de recursos i temps optimitzada per a 119 mitjana. El sistema permet el registre precís d'hores per projecte i tasca, visualització de dades mitjançant dashboards interactius i un mòdul d'assistència basat en IA per a l'optimització de la planificació. A causa de la sensibilitat de la informació operativa de l'empresa, el projecte es desenvolupa sota un acord de confidencialitat (NDA)
Castellà:	Este proyecto consiste en el diseño y desarrollo de una aplicación web de gestión de recursos y tiempo optimizada para 119 media. El sistema permite el registro preciso de horas por proyecto y tarea, visualización de datos mediante dashboards interactivos y un módulo de asistencia basado en IA para la optimización de la planificación. Debido a la sensibilidad de la información operativa de la empresa, el proyecto se desarrolla bajo un acuerdo de confidencialidad (NDA).
Anglès:	This project involves the design and development of a web-based resource and time management application optimized for 119 media. The system allows for the accurate recording of hours per project and task, data visualization through interactive dashboards, and an AI-powered assistance module for optimizing planning. Due to the sensitivity of the company's operational information, the project is being developed under a non-disclosure agreement (NDA).

Índice

1. Introducción 7
2. Justificación 8
   2.1. Relevancia Práctica y Empresarial 9
   2.2. Aportación Tecnológica e Innovación 10
   2.3. Interés Personal y Académico 11
   2.4. Escalabilidad y Proyección Futura 12
3. Objetivos 15
   3.1. Objetivo General 15
   3.2. Objetivos Específicos 15
4. Marco Conceptual 18
   4.1. Evolución Histórica de los Métodos de Registro de Horas y Control de Proyectos 18
   4.2. Histórico del Diseño de Aplicaciones y Evolución de las Interfaces de Usuario (UX/UI) 19
   4.3. Benchmark de Aplicaciones Existentes y Oportunidades de Innovación 21
   4.4. Evolución de las Arquitecturas Técnicas de Desarrollo Web y de Escritorio 22
5. Metodología y Planificación 25
   5.1. Marco de trabajo Agile y Sprints 25
   5.2. Gestión del proyecto con ClickUp 27
   5.3. Proceso de diseño y desarrollo 30
6. Análisis de Requisitos y Diseño 32
   6.1. Requisitos Funcionales 32
   6.2. Requisitos No Funcionales 34
   6.3. Casos de Uso del Sistema 36
   6.4. Sistema de Alertas y UX Predictiva 39
   6.5. Arquitectura de Roles y Permisos 39
   6.6. Modelo de Datos 41
   6.7. Entorno de Datos (Contexto 119.media) 43
   6.8. Diseño de Privacidad y Confidencialidad 44
7. Implementación Técnica (Avance) 46

7.1. Interfaz de Usuario (UI) y Tematización 46
7.2. Dashboards y Visualización de Datos 47
7.3. Funcionalidad de Registro (Temporizador) 47
7.4. Stack tecnológico y arquitectura del cliente 48
7.5. Backend Supabase, multi-tenant y RLS 51
7.6. Sistema de alertas, recordatorios y panel de bandeja (Tray) 53
7.7. Autenticación y almacenamiento seguro 54
7.8. Internacionalización, paleta de comandos y feedback al usuario 56
7.9. Empaquetado, distribución y updater nativo 57
7.10. Plan de Testeo y Validación Técnica 60
7.11. Endurecimiento de Seguridad y Configuración Nativa 61
7.12. Organización en Monorepo y Flujo de Desarrollo 62
8. Modelo de Negocio y Escalabilidad 64
8.1. Modelo de Suscripción y Pricing 65
8.2. Servicios de Mantenimiento, Soporte y Vías de Ingreso Añadidas 70
8.3. Personalización y "Whitelabel" 72
8.4. Enfoque hacia el Mercado B2B 74
8.5. Análisis competitivo y matriz de diferenciación 75
8.6. Cliente ideal (ICP) y segmentación de mercado 76
8.7. Estrategia de adquisición y canales de comercialización (Go-to-Market) 77
8.8. Métricas SaaS clave y unit economics 78
8.9. Proyección financiera y simulación de recuperación de la inversión 79
8.10. Análisis de riesgos del modelo y planes de mitigación 80
8.11. Roadmap comercial frente al roadmap de producto 81
9. Conclusiones y proyección futura 82
9.1. Conclusiones del Proyecto 82
9.2. Proyección Futura y Escalabilidad 83
10. Glosario de términos 87
11. Referencias 91
12. índice de figuras 94
13. Anexos 96

Anexo A. Declaración de uso de herramientas de inteligencia artificial generativa 96
Anexo B. Repositorio de código fuente 100
Anexo C. Material audiovisual y capturas de la aplicación 100
Anexo D. Sitio web institucional del proyecto 100
Anexo E. Presupuesto comercial del desarrollo 101
Anexo F. Acuerdo de confidencialidad (NDA) con 119.media 109

1. Introducción

El presente Trabajo de Fin de Grado aborda el diseño y desarrollo de una solución tecnológica avanzada para la gestión del tiempo y la optimización de recursos humanos en entornos corporativos de alta demanda. El proyecto se centra en la creación de una aplicación web personalizada para la empresa 119.media, diseñada para transformar la manera en que el equipo registra, visualiza y analiza su actividad diaria.

En el ecosistema empresarial actual, la gestión eficiente del tiempo no es solo una medida de productividad, sino un activo estratégico. Sin embargo, muchas organizaciones enfrentan dificultades debido a sistemas de registro arcaicos o excesivamente complejos que desmotivan al usuario y generan datos imprecisos. Esta propuesta nace con el objetivo de resolver dicha problemática mediante una interfaz UX/UI simplificada, un sistema de alertas inteligentes y la integración de un motor de Inteligencia Artificial (IA) que convierte el registro de horas en *insights* accionables.

Debido a que el software maneja información sensible sobre la estructura operativa, proyectos estratégicos (como *Winter25* o *Pla Estratègic*) y el rendimiento del personal de 119.media, el desarrollo de esta memoria se rige bajo un Acuerdo de Confidencialidad (NDA). Por consiguiente, aunque los procesos y arquitecturas se describen con rigor académico, ciertos datos específicos han sido anonimizados para proteger la propiedad intelectual y la privacidad de la organización.

El documento se estructura en tres fases principales: un análisis de requisitos y modelado de la arquitectura; el diseño de una experiencia de usuario optimizada para entornos *cross-platform*; y la implementación de un prototipo funcional que integra dashboards analíticos y asistencia mediante IA.

2. Justificación

La elección de este tema responde a una necesidad crítica detectada en el sector de la comunicación y los servicios digitales: la brecha entre la recolección de datos temporales y su utilidad real para la toma de decisiones. En el ecosistema de las agencias interactivas y consultoras tecnológicas, el tiempo constituye el recurso más valioso y, simultáneamente, el más complejo de cuantificar de manera exacta. Históricamente, las organizaciones han implementado mecanismos de control horario motivadas por imperativos legales o por un paradigma industrial heredado que equiparaba la presencia física con el rendimiento real. Sin embargo, en entornos de alta demanda creativa donde los flujos de trabajo son intrínsecamente dinámicos y fragmentados, los sistemas tradicionales han devenido en estructuras obsoletas que generan datos imprecisos.

Esta desconexión crónica da lugar a una gestión puramente reactiva. Los directores de proyectos e investigadores operativos identifican las desviaciones presupuestarias, el colapso de los equipos o la falta de rentabilidad de una cuenta a posteriori, es decir, cuando el proyecto ya ha concluido y el margen de maniobra es inexistente. Cuando los trabajadores se ven obligados a reconstruir su jornada laboral de forma retrospectiva al final de una semana, introducen sesgos cognitivos severos como la falacia de la planificación y el sesgo de disponibilidad, distribuyendo el tiempo en bloques homogéneos basados en porcentajes ideales que ocultan las verdaderas fricciones del día a día, tales como reuniones imprevistas, corrección de errores críticos o tareas de gestión interna.

La justificación fundamental de este trabajo radica en estrechar esta brecha mediante el diseño de una solución tecnológica avanzada. Al proponer un sistema centrado en la usabilidad que captura el tiempo con granularidad atómica basada en minutos reales y mediante mecanismos proactivos, como alertas contextuales y paneles de acceso rápido en la bandeja del sistema, se transforma el registro de tiempo de una carga burocrática pasiva a un flujo continuo de información de alta pureza. Este enfoque permite que los datos brutos dejen de ser un mero archivo

histórico y se conviertan, mediante módulos de analítica interactiva e inteligencia artificial, en un activo estratégico fundamental para el control de costes exacto, la asignación eficiente de cargas de trabajo y la toma de decisiones predictivas en tiempo real.

2.1. Relevancia Práctica y Empresarial

Para 119.media, la implementación de esta herramienta supone una evolución fundamental desde una gestión reactiva, basada en la corrección de desviaciones a posteriori, hacia una gestión proactiva y predictiva en tiempo real. En el sector de la comunicación y los servicios digitales, donde los márgenes de beneficio dependen directamente de la asignación eficiente de las horas por trabajador, disponer de métricas precisas es un factor de competitividad crítico. La capacidad de registrar horas por minutos, evitando el sesgo generalizado de los repartos porcentuales abstractos, permite capturar la actividad del equipo con una granularidad atómica. Al vincular de forma directa cada micro-registro a áreas y proyectos específicos de la empresa, como IT, RRHH, Web o proyectos estratégicos, la organización adquiere la capacidad de realizar un control de costes exacto y una auditoría de rendimiento financiero por cada proyecto.

La verdadera justificación práctica de este proyecto radica en la transformación cultural y operativa de la propia herramienta de control. Tradicionalmente, los sistemas de time-tracking han sido percibidos por las plantillas como mecanismos invasivos de vigilancia o como una tediosa carga administrativa que interrumpe el estado de enfoque (flow state) de los perfiles creativos y técnicos. Esta propuesta rompe ese paradigma al diseñarse no como un elemento de fiscalización, sino como una ayuda directa para el trabajador en su jornada diaria.

Para lograrlo, la relevancia del sistema se sustenta en mecánicas de interacción diseñadas específicamente para minimizar la fricción del usuario. El despliegue de ventanas emergentes (pop-ups) de validación predictiva a horas configurables y el desarrollo de un panel de acceso rápido anclado a la bandeja del sistema (Tray

Panel) permiten al empleado interactuar con el cronómetro de forma casi invisible, sin necesidad de mantener la aplicación principal abierta en primer plano. Además, la futura sincronización e integración planificada con los ecosistemas de Google facilita la automatización de flujos de información y la unificación de calendarios. Esto no solo garantiza la recolección de datos de alta pureza para los Managers y Superadmins, sino que reduce la carga cognitiva del User Basic, fomentando la adopción orgánica del hábito del registro diario y optimizando globalmente el uso del talento dentro de 119.media.

2.2. Aportación Tecnológica e Innovación

La principal innovación de este trabajo es el uso de la Inteligencia Artificial no como un mero añadido cosmético o una función secundaria, sino como un módulo de asistencia nuclear e integrado de forma nativa en la arquitectura del sistema. Mientras que las herramientas convencionales de time-tracking disponibles en el mercado operan de manera estática y puramente descriptiva, limitándose a almacenar filas de datos brutos y generar gráficos del pasado, esta propuesta evoluciona hacia la analítica predictiva y prescriptiva. El agente de IA procesa de los vectores históricos de las series temporales de registros horarios para identificar patrones de comportamiento organizacional complejos, detectar cuellos de botella operativos antes de que provoquen retrasos críticos en las entregas y ofrecer un sistema de feedback y recomendaciones totalmente personalizado según los requisitos del proyecto y de la empresa.

Esta integración representa una aportación técnica significativa tanto al campo de la comunicación interactiva como al de la ingeniería del software y la gestión de proyectos. Desde una perspectiva arquitectónica, el verdadero hito tecnológico radica en haber implementado nuevas tecnologías en una aplicación de control de proyectos. La aplicación rompe con la dependencia histórica de frameworks pesados de escritorio basados en Chromium y Node.js, implementando una solución híbrida multiplataforma nativa construida sobre Tauri v2 con un núcleo de rendimiento y seguridad de memoria programado en Rust. Este binario ligero interactúa de forma segura con un backend en la nube basado en Supabase, donde

la lógica de negocio multi-inquilino (multi-tenant) y la privacidad absoluta de los datos confidenciales requeridos por el acuerdo de NDA quedan blindadas físicamente a nivel de base de datos mediante políticas de Seguridad a Nivel de Fila (Row Level Security - RLS).

Al conectar este entorno de datos estructurado y ultra seguro con la capacidad analítica de los modelos de lenguaje a través de llamadas avanzadas a APIs de IA, el sistema demuestra de manera empírica cómo el análisis de datos de alta precisión puede ponerse al servicio del factor humano. En lugar de utilizar las métricas para ejercer una presión punitiva, el software utiliza los algoritmos para mejorar el bienestar laboral y la salud organizativa. Al cruzar los tiempos reales de ejecución con las asignaciones de tareas, la asistencia por IA es capaz de alertar a los administradores y directores de proyectos (Managers) sobre saturaciones crónicas de horas en determinados perfiles, permitiendo una reasignación inteligente de recursos que previene activamente la sobrecarga de tareas y el síndrome de desgaste ocupacional (burnout), marcando un nuevo estándar en el diseño de herramientas de productividad empresarial.

2.3. Interés Personal y Académico

Como futuros profesionales del ámbito interactivo y la ingeniería de software aplicada, este proyecto representa un hito fundamental en nuestra trayectoria formativa, ya que nos permite aplicar de forma integral, transversal y simultánea competencias avanzadas en desarrollo full-stack, diseño de interfaces y análisis de datos. La naturaleza multidisciplinar del Trabajo de Fin de Grado (TFG) exige la convergencia de dos perfiles especializados que deben coordinarse de manera matemática: por un lado, la dirección de David en el diseño de la experiencia de usuario (UX/UI), la maquetación responsiva del front-end y la lógica de visualización interactiva; por otro lado, el liderazgo de Albert en la arquitectura del back-end, la optimización de la infraestructura en la nube y el despliegue del motor analítico de Inteligencia Artificial. Esta división de responsabilidades simula fielmente la estructura y las metodologías de trabajo de las células de desarrollo de alta eficiencia en la industria tecnológica actual.

La oportunidad de implementar esta solución en un entorno empresarial real y bajo un estricto Acuerdo de Confidencialidad (NDA) añade una capa insustituible de profesionalismo y rigor metodológico al TFG. Lejos de limitarse a un ejercicio académico teórico o a un entorno de simulación controlado con datos ficticios (mock data), el proyecto nos obliga a enfrentarnos a las restricciones, presiones y desafíos técnicos reales que rigen el mercado del software corporativo moderno. El manejo de información sensible sobre la estructura organizativa y las métricas de costes de 119.media nos ha empujado a priorizar la seguridad de la información como un requisito de diseño no funcional de carácter crítico.

Académicamente, esto se traduce en la necesidad de diseñar e implementar un sistema de gobierno de datos robusto, basado en una arquitectura multi-tenant y un esquema de gestión de permisos por roles estrictamente segmentado (Superadmin, Admin, Manager y User Basic). La correcta articulación de este modelo asegura que cada perfil interactúe exclusivamente con las capas de datos asociadas a su nivel de responsabilidad corporativa. Asimismo, el desarrollo nos ha planteado retos significativos de escalabilidad técnica, tales como la migración automatizada de datos, el endurecimiento de la seguridad de los tokens de sesión en el almacenamiento local nativo (Keychain en macOS) y la automatización del empaquetado y distribución multiplataforma firmado mediante flujos de integración continua en GitHub Actions. En definitiva, este TFG no solo consolida los conocimientos adquiridos a lo largo del grado, sino que nos dota de una experiencia práctica de primer nivel, alineada con las demandas y los estándares de calidad más exigentes del sector interactivo actual.

2.4. Escalabilidad y Proyección Futura

Un aspecto diferenciador y de alto valor estratégico de esta propuesta es su intrínseco potencial de escalabilidad, el cual trasciende la mera resolución de un problema local. Aunque el software se concibe y desarrolla inicialmente como una solución personalizada para cubrir las necesidades operativas, los flujos de trabajo y las carencias métricas específicas de 119.media, toda su infraestructura y

arquitectura técnica subyacente han sido diseñadas desde los cimientos bajo las directrices de un modelo de Software as a Service (SaaS). Esta visión a largo plazo implica que el sistema no se comporta como un bloque monolítico rígido, sino como una plataforma modular, parametrizable y altamente adaptable.

La adopción de este enfoque arquitectónico modular, gestionado de manera eficiente mediante un monorepo distribuido en pnpm workspaces (separando limpiamente la aplicación nativa, la landing web y las librerías compartidas de utilidad y diseño), garantiza que la lógica de negocio nuclear pueda ser desacoplada del contexto exclusivo de la empresa inicial. De este modo, la estructura de base de datos relacional, la matriz jerárquica de permisos por roles (Superadmin, Admin, Manager y User Basic) y, fundamentalmente, el motor de análisis prescriptivo alimentado por Inteligencia Artificial pueden replicarse y adaptarse con mínima fricción a otras agencias de comunicación, estudios creativos, productoras audiovisuales o consultoras tecnológicas que compartan problemáticas similares en la gestión del talento y la rentabilidad temporal de sus proyectos.

Esta flexibilidad de diseño abre la puerta a que, en una fase post-TFG, el proyecto pueda evolucionar de forma natural hacia una herramienta comercializable a gran escala en el mercado B2B. Al estar sustentada sobre una base de datos PostgreSQL en Supabase dotada de una arquitectura multi-tenant nativa, la plataforma es capaz de gestionar múltiples organizaciones independientes (tenants) de manera simultánea sobre la misma infraestructura física. El aislamiento absoluto, la integridad y la privacidad de los datos de cada cliente quedan garantizados de forma matemática en el motor de la base de datos a través de las políticas de Seguridad a Nivel de Fila (Row Level Security - RLS), asegurando que ninguna empresa pueda interceptar o visualizar los costes, proyectos o registros de otra.

Asimismo, la proyección futura de este software destaca por su adaptabilidad a nivel de infraestructura de redes y soberanía de datos. La separación limpia entre la capa de presentación construida en React y TypeScript, y los servicios del servidor,

permite que el modelo de negocio no se limite a una nube pública administrada de forma centralizada. El sistema está preparado para ofrecer un modelo de despliegue híbrido o local (Self-Hosted). Esto permitirá que aquellas corporaciones que operen bajo regulaciones legales extremas o estrictos acuerdos de confidencialidad puedan instalar el ecosistema directamente en sus propios servidores privados, reduciendo costes de suscripción y reteniendo el control absoluto de su información, lo que posiciona a esta herramienta como una alternativa altamente competitiva, escalable y robusta frente a las soluciones genéricas del mercado global.

3. Objetivos

El propósito central de este TFG es la convergencia entre el desarrollo de software robusto y el análisis inteligente de datos para optimizar la gestión del talento en entornos creativos.

3.1. Objetivo General

Diseñar y desarrollar un prototipo funcional de aplicación web (MVP) orientado al contexto real de la empresa 119.media, que permita el registro sistemático de la actividad del equipo, la visualización del uso del tiempo y la generación de recomendaciones mediante inteligencia artificial para mejorar la planificación operativa.

Este objetivo general no se concibe como una meta aislada, sino como el punto de convergencia entre las tres dimensiones que estructuran el presente trabajo: la investigación del estado del arte, el diseño centrado en el usuario y la ingeniería de software aplicada. En coherencia con el marco metodológico híbrido descrito en el capítulo de Metodología y Planificación, cada objetivo específico se ha definido para ser verificable de forma empírica sobre el prototipo funcional (MVP), de manera que su cumplimiento pueda contrastarse contra requisitos concretos y, posteriormente, contra las decisiones de implementación técnica adoptadas. De este modo, los objetivos operan como criterios de aceptación de alto nivel que recorren transversalmente las fases de análisis, diseño y desarrollo, garantizando que ninguna funcionalidad construida quede desvinculada de una necesidad real y previamente justificada de 119.media.

3.2. Objetivos Específicos

Para asegurar el cumplimiento del objetivo general, se han definido las siguientes metas tácticas divididas por áreas de actuación:

Investigación y Definición:

- Identificar y documentar los requisitos funcionales específicos de 119.media, definiendo casos de uso, flujos de trabajo y necesidades de reporte.
- Establecer un marco de seguridad y confidencialidad mediante un NDA para el manejo de datos internos de la organización.

- Realizar un *benchmark* de herramientas de *time-tracking* existentes para detectar oportunidades de mejora en la experiencia de usuario.

Diseño y Experiencia de Usuario (UX/UI):

- Diseñar una interfaz *cross-platform* que garantice un registro de horas ágil y minimalista, optimizando la carga administrativa del trabajador.
- Crear un sistema de navegación basado en roles (Superadmin, Admin, Manager, User) que asegure que cada perfil acceda solo a la información pertinente.
- Prototipar visualizaciones de datos (dashboards) claras para la consulta de horas por proyecto, persona y periodo.

Desarrollo Técnico e Implementación:

- Implementar un motor de registro de horas preciso, basado en minutos y no en porcentajes, para evitar distorsiones en la métrica de productividad.
- Desarrollar un sistema de alertas y *pop-ups* de validación personalizables que fomenten el hábito del registro diario.
- Integrar un módulo de Inteligencia Artificial capaz de analizar patrones de comportamiento y generar informes con recomendaciones automáticas sobre reasignación de recursos.

Validación y Documentación:

- Validar el prototipo funcional con usuarios de la empresa para iterar mejoras basadas en el feedback directo.
- Elaborar una memoria técnica exhaustiva que documente la arquitectura del sistema, el modelo de datos y las conclusiones del proceso de desarrollo.

Trazabilidad de Objetivos, Requisitos e Implementación

Con el fin de asegurar la coherencia integral del proyecto y facilitar la defensa académica del mismo, se establece una matriz de trazabilidad que vincula cada objetivo específico con el requisito funcional o no funcional que lo materializa y con la subsección de la implementación técnica donde dicho requisito se resuelve. Esta trazabilidad bidireccional, alineada con las buenas prácticas de la ingeniería de requisitos, demuestra que el alcance del MVP responde de forma disciplinada a las metas planteadas y permite auditar el grado de avance real del sistema.

4. Marco Conceptual

El marco teórico de este proyecto documenta los antecedentes científicos, metodológicos y tecnológicos que han regido el control del tiempo laboral y la gestión de proyectos. Esta investigación fundamenta el paso desde los sistemas mecánicos analógicos de control industrial hasta las modernas plataformas de software como servicio (SaaS) potenciadas por Inteligencia Artificial, contextualizando cómo la evolución del diseño de interfaces (UX/UI) y las arquitecturas de datos han dado forma al Producto Mínimo Viable (MVP) desarrollado para 119.media.

4.1. Evolución Histórica de los Métodos de Registro de Horas y Control de Proyectos

La cuantificación del tiempo de trabajo ha sido un pilar central en la estructuración de la economía industrial y del conocimiento. Comprender cómo los métodos analíticos han evolucionado permite identificar los fallos sistémicos nuestro software busca solucionar en el entorno digital contemporáneo.

Del Reloj de Fichar Mecánico a la Hoja de Cálculo Digital

El registro de horas sistemático nació durante la Segunda Revolución Industrial con las patentes del reloj de fichar mecánico de Willard Bundy (1888). Este enfoque, puramente taylorista, consideraba el tiempo como una métrica homogénea de presencia física en una línea de ensamblaje (Taylor, 1911). La introducción de tarjetas de cartón perforadas garantizaba un control de costes directo pero unidimensional: el trabajador vendía bloques masivos de horas, sin distinción de la naturaleza de la tarea.

Con la llegada de la informática de oficina en la década de 1980, el paradigma físico se trasladó a las hojas de cálculo digitales como Microsoft Excel. Las empresas comenzaron a usar plantillas de cuadrículas de tiempo (*timesheets*) herramienta que sigue vigente en organizaciones contemporáneas, incluyendo el escenario de estudio de este proyecto. Si bien esto permitió digitalizar los registros y realizar sumatorios básicos, introdujo problemas críticos vigentes en el sector actual:

- Asincronía e Inexactitud: El registro pasó a ser reactivo (el empleado rellena la hoja al final del día, de la semana o del mes), dependiendo enteramente de la memoria del usuario.
- Falta de granularidad: Las hojas de cálculo son propensas a errores de formato y no permiten vincular el tiempo de forma dinámica a elementos dinámicos o relacionales como micro-tareas o costes cambiantes en tiempo real.

Modelos Históricos de Gestión de Proyectos: El Gráfico de Gantt y el Método del Camino Crítico

A nivel analítico, la necesidad de estructurar el uso del tiempo dio origen a las primeras herramientas visuales de gestión. Henry Gantt desarrolló en 1910 el **Gráfico de Gantt**, un sistema de barras horizontales que vinculaba actividades específicas con duraciones de tiempo prefijadas (Gantt, 1919).

Posteriormente, en la década de 1950, la corporación DuPont y el ejército de los Estados Unidos desarrollaron de forma independiente el **Método del Camino Crítico (CPM)** y la **Técnica de Revisión y Evaluación de Proyectos (PERT)**. Estos modelos matemáticos permitían identificar la secuencia de tareas indispensables que determinaban la duración mínima de un proyecto (Project Management Institute, 2017).

Sin embargo, tanto Gantt como PERT/CPM fueron diseñados para industrias pesadas o infraestructuras rígidas. Al aplicarse a agencias de comunicación, comunicación interactiva, producción audiovisual y desarrollo de software actuales, estos sistemas estáticos fracasan si no están nutridos por un software de cronometraje en tiempo real capaz de reajustar los caminos críticos basándose en minutos reales consumidos y no en planificaciones estáticas ideales.

4.2. Histórico del Diseño de Aplicaciones y Evolución de las Interfaces de Usuario (UX/UI)

El éxito o fracaso de una herramienta de gestión del tiempo está íntimamente ligado a la evolución de los paradigmas de diseño interactivo. El software corporativo ha pasado de dar prioridad a la base de datos a dar prioridad a la cognición del usuario.

De las Interfaces Basadas en Formularios Dense al Diseño Centrado en el Usuario (UCD)

Las primeras aplicaciones de *time-tracking* basadas en la web (finales de los 90 y principios de los 2000) replicaban la estructura de los formularios físicos. El usuario se enfrentaba a interfaces densas repletas de menús desplegables anidados (*dropdowns*) donde debía seleccionar cliente, proyecto, subproyecto, código de facturación e introducir las horas manualmente mediante teclado. Este paradigma violaba principios básicos de la **usabilidad** de Jakob Nielsen (1994), incrementando drásticamente el tiempo de interacción y provocando el rechazo sistémico del empleado.

El cambio hacia el Diseño Centrado en el Usuario (*User-Centered Design*) redefinió las herramientas de productividad (Norman, 2013). La literatura demostró que para recolectar datos limpios es necesario minimizar el número de clics y la carga cognitiva del operario. Se popularizó el uso de **temporizadores de un solo clic** (*action-driven UI*), donde la interfaz de usuario pasa a un segundo plano y se integra en el flujo diario mediante barras de herramientas accesibles o paneles reducidos.

Evolución Teórica de los Dashboards Analíticos

La visualización de datos de rendimiento corporativo evolucionó en paralelo. Inicialmente, los cuadros de mando eran informes tabulares estáticos que requerían exportación y procesamiento secundario (Few, 2006). Con la maduración de las librerías gráficas web y los estudios de percepción visual de la información de Edward Tufte (2001), los tableros analíticos se convirtieron en paneles interactivos en tiempo real.

La teoría moderna del diseño de dashboards establece que los indicadores clave de rendimiento (KPIs) deben distribuirse en capas de lectura jerárquica: una vista macro para toma de decisiones directivas (Superadmin/Manager) y una vista micro orientada a la autogestión de la productividad del propio trabajador (User Basic), evitando la saturación de información (information overload).

4.3. Benchmark de Aplicaciones Existentes y Oportunidades de Innovación

Para validar la necesidad científica y comercial del MVP diseñado para *119.media*, es imprescindible mapear y analizar críticamente las soluciones de software dominantes en el mercado global actual de gestión del tiempo y recursos.

Análisis Crítico de Competidores Líderes

El mercado de plataformas de control de tiempos está altamente maduro, categorizándose en tres grandes enfoques arquitectónicos y de producto:

ENFOQUE DE SISTEMA		
Estáticos Toggle / Harvest (Paso manual, Baja analítica)	Integrados ClickUp / Jira (Fricción de uso por complejidad)	Proactivos MVP 119.media (Registro rápido, Análisis IA, RLS)

Figura 4.1 Tabla de enfoques de sistemas de gestión de proyectos.

Toggle Track / Harvest (Herramientas Especializadas Estáticas):

- *Ventajas:* Pioneros en la simplificación del temporizador en entornos de navegador web y extensiones. Muy eficientes para el registro rápido de horas individuales.
- *Desventajas:* Operan de forma aislada. La analítica que ofrecen es descriptiva y pasada, careciendo de lógicas profundas de prevención de sobrecarga o inteligencia prescriptiva. Sus modelos SaaS exigen el almacenamiento obligatorio en nubes públicas propietarias, lo que entra en conflicto directo con normativas corporativas estrictas de acuerdos de confidencialidad (NDA).

ClickUp / Jira (Sistemas Integrados de Gestión de Proyectos):

- *Ventajas:* Centralizan la tarea, el documento y la estimación temporal en un ecosistema robusto que mapea flujos de trabajo metodológicos complejos.

- *Desventajas:* La fricción del usuario se eleva exponencialmente. El registro de horas suele estar enterrado bajo múltiples capas de menús dentro de la tarea, lo que incentiva el abandono del hábito diario por parte del trabajador. Al ser plataformas masivas, su rendimiento técnico en equipos de escritorio locales puede ser pesado e ineficiente debido a arquitecturas basadas en contenedores pesados.

La Brecha de Innovación Identificada

El estudio comparativo de este estado del arte revela una brecha tecnológica clara: **las herramientas del mercado son reactivas y estáticas**. Capturan el dato que introduce el usuario, pero no devuelven valor inmediato ni al empleado ni al administrador de forma automatizada.

La oportunidad de innovación del proyecto radica en la fusión de tres elementos que la literatura actual reclama pero las herramientas genéricas omiten de forma conjunta: la agilidad de una app de escritorio nativa superligera (*Tauri v2*), la protección jurídica del dato mediante despliegues multi-inquilino segregados (*RLS en PostgreSQL*), y la transformación del registro de tiempo bruto en consejos organizacionales predictivos mediante un módulo de Inteligencia Artificial enfocado en el bienestar y la rentabilidad empresarial.

4.4. Evolución de las Arquitecturas Técnicas de Desarrollo Web y de Escritorio

El cómo hemos llegado a la solución técnica seleccionada (Tauri v2, React, Supabase) responde a una sucesión histórica de superación de limitaciones de ingeniería de software.

El Ecosistema Web y el Tránsito Hacia las Aplicaciones Nativas Híbridas

En la década de 2000, las aplicaciones de gestión interna se dividían estrictamente en aplicaciones web accesibles, pero con capacidades limitadas en el sistema operativo, o aplicaciones nativas monolíticas (C++, Java) de alto rendimiento, pero con costes de desarrollo y mantenimiento prohibitivos para múltiples plataformas (Windows, macOS, Linux).

La aparición del motor V8 de JavaScript y de arquitecturas como Electron en 2013 eliminó esta frontera, permitiendo usar lenguajes web (HTML, CSS, JS) para construir software de escritorio empaquetado. No obstante, como detalla la literatura técnica contemporánea, el coste informático de Electron (duplicar instancias del navegador Chromium en memoria RAM por cada app abierta) generó una crisis de optimización en los ordenadores corporativos de los usuarios (Flaki, 2021).

Tauri v2 surge históricamente para corregir esta ineficiencia, reemplazando la base de Chromium por los motores de renderizados nativos del sistema e implementando la lógica de control del ciclo de vida en un lenguaje seguro y eficiente a nivel de memoria y CPU como Rust (Tauri Core Team, 2024).

El Paradigma "Backend-as-a-Service" (BaaS) y las Bases de Datos Relacionales

A nivel de infraestructura de datos, el desarrollo tradicional requería la programación y mantenimiento de servidores de aplicaciones dedicados (módulos Express, Spring, etc.) para conectar el cliente con la base de datos. El nacimiento del paradigma *Backend-as-a-Service* (BaaS) a través de entornos gestionados como Supabase democratizó el despliegue de MVPs altamente complejos y seguros.

Al estructurar el sistema sobre PostgreSQL, el backend del proyecto hereda más de 30 años de evolución en consistencia y teoría de bases de datos relacionales, cumpliendo estrictamente con las *propiedades ACID* (acrónimo en inglés de atomicidad, consistencia, aislamiento y durabilidad). Esto posibilita la inyección de lógicas críticas de control a nivel físico de tabla (como las restricciones de no negatividad en la duración de las horas y el aislamiento multi-inquilino mediante *Row Level Security*), garantizando que las auditorías operativas exigidas bajo contratos NDA de confidencialidad empresarial se ejecuten bajo los más altos estándares de la ingeniería informática actual (Silberschatz et al., 2020).

Síntesis Histórica-Conceptual de la Investigación

Para estructurar visualmente cómo vuestro TFG aprovecha las bases de la investigación previa para solucionar las problemáticas actuales, se establece el siguiente cuadro analítico:

Época / Sistema Histórico	Paradigma de Control	Limitación del Enfoque Antiguo	Solución Innovadora en el MVP
Reloj de fichar mecánico (1888) / Taylorismo industrial.	Presencia física atada al espacio de trabajo.	Unidimensional, rígido, nula visibilidad del rendimiento de tareas del sector creativo.	Temporizador atómico por minutos vinculado directamente a flujos específicos (IT, RRHH, Web).
Hojas de Cálculo Digitales (1980) / Gestión de oficina clásica.	Registro manual reactivo en celdas asincrónicas.	Propensión a fallos tipográficos, pérdida de trazabilidad, dependencia de la memoria.	UX Predictiva con pop-ups y Quick-entry disparados al cambio preciso de minuto desde el Tray nativo.
SaaS Monolíticos y Electron (2013) / Primeras apps web híbridas.	Centralización total del dato, consumo masivo de recursos web.	Lentitud en equipos, binarios pesados (>100MB), riesgo legal de fugas de datos bajo NDA.	Tauri v2 (Rust) de alto rendimiento con aislamiento estricto multi-tenant mediante PostgreSQL RLS.
Dashboards e Informes Estáticos (2000s) / Analítica descriptiva.	Gráficos e informes del pasado que requieren lectura humana exhaustiva.	Incapacidad de predecir el colapso de un equipo o el desvío de costes de forma proactiva.	Módulo de Asistencia con IA enfocado en analítica prescriptiva y prevención del síndrome de burnout.

Figura 4.2 Tabla de histórico de sistemas de gestión de proyectos.

5. Metodología y Planificación

La ejecución y articulación de este proyecto se fundamenta conceptual y operativamente en metodologías de trabajo dinámicas, iterativas y de alta flexibilidad, diseñadas específicamente para permitir una adaptación constante, fluida y rigurosa a los requisitos cambiantes y las necesidades operativas reales de 119.media. Lejos de adoptar un enfoque predictivo tradicional o en cascada (Waterfall), el cual resultaría ineficaz y excesivamente rígido ante la naturaleza volátil de un ecosistema digital e interactivo, se ha optado de forma estratégica por la implementación de un marco metodológico híbrido. Este enfoque combina las virtudes del Design Thinking para la fase inicial de descubrimiento, empatía e ideación conceptual, con los principios y ceremonias del marco de trabajo Agile (metodologías ágiles) para la fase de desarrollo, implementación técnica y despliegue del Producto Mínimo Viable (MVP).

Esta hibridación metodológica garantiza un equilibrio perfecto entre la exploración creativa centrada en el usuario y la disciplina de ingeniería de software necesaria para construir un sistema seguro, multi-inquilino (multi-tenant) y escalable. Mientras que el Design Thinking aporta las herramientas analíticas para entender las verdaderas fricciones de los trabajadores a la hora de registrar sus horas y modelar la interfaz cross-platform sin añadir carga cognitiva, las metodologías ágiles estructuran el backlog técnico en bloques manejables de código, priorizando la entrega continua de valor y permitiendo al equipo de desarrollo pivotar o refactorizar la arquitectura en función del feedback directo e inmediato recibido por parte de los administradores de la organización.

5.1. Marco de trabajo Agile y Sprints

El ciclo de desarrollo técnico e implementación del software se organiza de forma rigurosa mediante el marco de trabajo *Agile*, desglosando la totalidad del proyecto en bloques temporales fijos denominados *sprints* de corta duración, planificados metodológicamente para abarcar un intervalo de aproximadamente 3 semanas cada uno. Esta cadencia de trabajo cíclica y predecible responde a los estándares de la

ingeniería del software moderna (Schwaber & Sutherland, 2020), facilitando de manera sistemática la entrega progresiva y continua de funcionalidades operativas (incrementos de software) y dotando al equipo de una altísima capacidad para pivotar, mitigar riesgos y reajustar los requisitos técnicos según el *feedback* directo e inmediato recolectado tras cada fase de validación con el representante de la organización.

Este marco ágil se articula en torno a dos pilares operativos fundamentales que garantizan la eficiencia del proyecto:

- **Iteración constante y gestión del *backlog*:** Cada *sprint* culmina de forma preceptiva con una reunión de revisión de objetivos alcanzados (*Sprint Review*), un proceso analítico donde se evalúa el comportamiento del prototipo funcional frente a las necesidades reales de *119.media*. Esta evaluación permite reordenar, refinar y ajustar dinámicamente el *backlog* de tareas pendientes. Si durante una iteración se detecta una fricción en el registro horario o una ineficiencia en las políticas de seguridad a nivel de fila (RLS), la metodología permite priorizar dicha corrección en el siguiente ciclo, asegurando que el desarrollo avance alineado con las prioridades estratégicas de la empresa.

- **Gestión y segregación de roles especializados:** A diferencia de las estructuras de desarrollo monolíticas, la flexibilidad de las metodologías ágiles permite una división clara, especializada y complementaria de las responsabilidades de ingeniería dentro del equipo del TFG. Bajo este esquema, **David** asume el rol de liderazgo enfocado al diseño interactivo (*UX/UI*), la arquitectura del cliente (*Front-end* en React y TypeScript), la gestión del diseño responsivo multiplataforma (Tauri v2) y la optimización de los flujos de visualización de los *dashboards*. Por su parte, **Albert** lidera de forma síncrona el desarrollo del *Back-end* sobre PostgreSQL (Supabase), la configuración de la infraestructura multi-inquilino (*multi-tenant*), el endurecimiento de la seguridad mediante políticas RLS en base de datos, y el diseño y entrenamiento de prompts para el módulo de análisis prescriptivo mediante Inteligencia Artificial. Esta especialización de roles, coordinada bajo un mismo repositorio de código y sincronizada de forma continua, optimiza la

velocidad de desarrollo del MVP y garantiza la cohesión de todo el ecosistema tecnológico.

5.2. Gestión del proyecto con ClickUp

Para la planificación detallada y el seguimiento pormenorizado de los hitos y entregables que componen este Trabajo de Fin de Grado, se ha utilizado ClickUp como plataforma de software e infraestructura centralizada para la gestión de proyectos. En el ámbito de la ingeniería de software y el desarrollo ágil, la elección de un entorno de gestión robusto es una decisión crítica; contar con una herramienta estática o descentralizada incrementaría el riesgo de desviaciones temporales o pérdida de trazabilidad en los requisitos corporativos. ClickUp actúa como estructura de control, permitiendo orquestar los flujos de trabajo mediante la centralización de datos en tiempo real y ofreciendo ventajas tácticas en las siguientes dimensiones cardinales:

- **Visualización dinámica del *Timeline* (Gantt y Calendario):** La plataforma permite modelar y monitorizar el ciclo de vida completo del desarrollo a través de un cronograma interactivo. Esto abarca desde la fase de conceptualización inicial e investigación del estado del arte (*Kick-off*), pasando por los sucesivos *sprints* de programación, hasta el proceso de redacción y entrega de la memoria académica final. Esta visualización macro garantiza que las dependencias entre tareas técnicas (como la necesidad de tener consolidadas las migraciones SQL en Supabase antes de implementar las vistas de los componentes en React) queden explícitamente reflejadas y controladas.

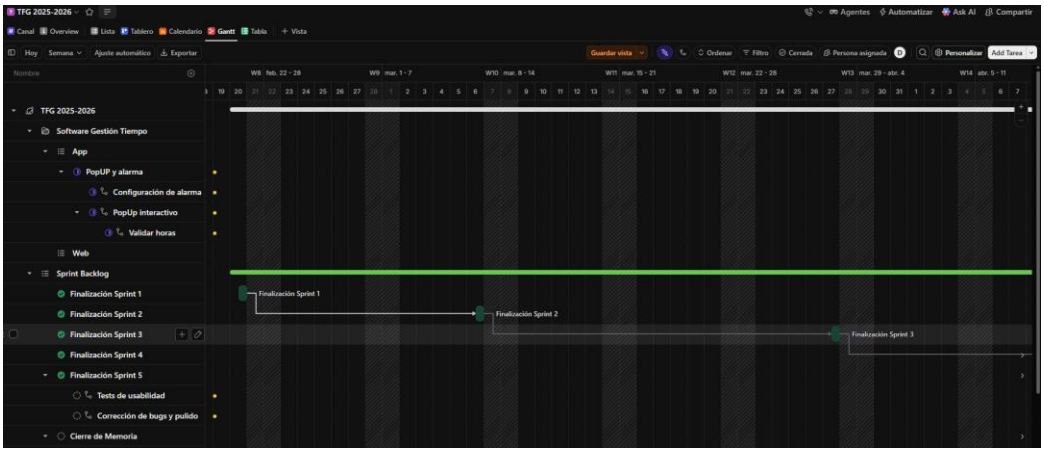


Figura 5.1 Imagen de referencia del proyecto de ClickUp Hoora.

- **Desglose granular y asignación de tareas:** Fieles a los principios de desglosar problemas complejos en unidades de código manejables y atómicas, los requisitos funcionales imprescindibles definidos para *119.media* se ramifican estructuralmente dentro de ClickUp en subtareas específicas con responsables asignados de forma unívoca. Elementos de alta complejidad técnica —tales como la matriz de permisos por roles para la navegación protegida (ProtectedRoute), la integración del plugin de notificaciones nativas de Tauri v2 en Rust, o el diseño de prompts para el motor de Inteligencia Artificial— son descompuestos en tareas de desarrollo con criterios de aceptación claros y estimaciones de tiempo realistas.

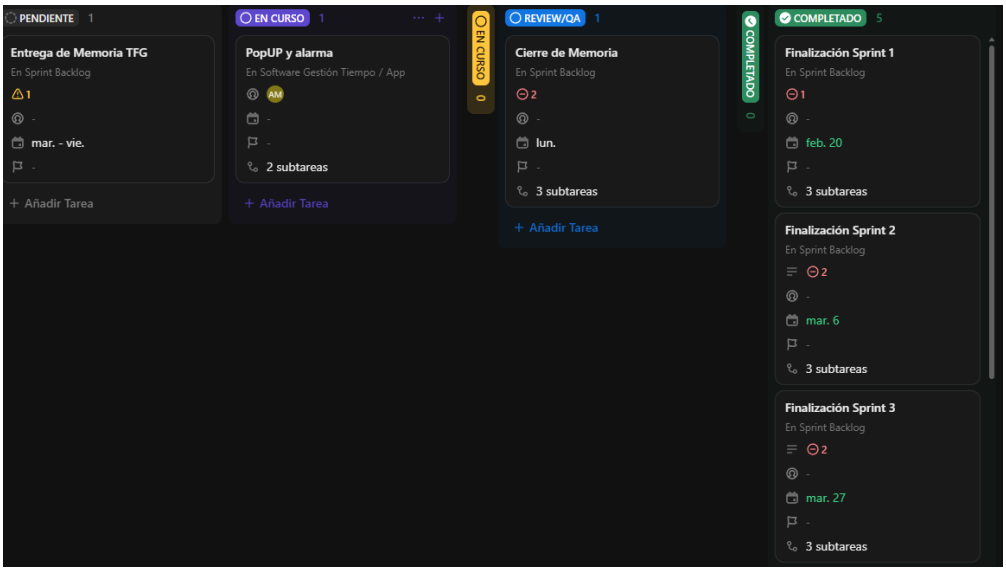


Figura 5.2 Imagen de referencia del proyecto de ClickUp Hoora.

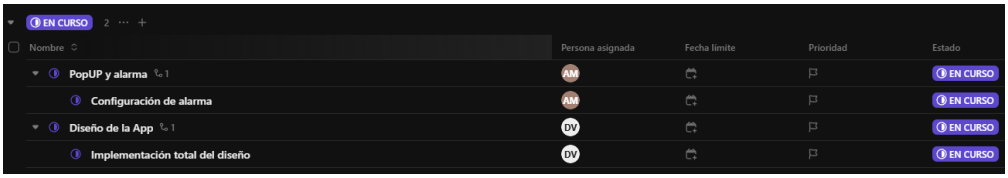


Figura 5.3 Imagen de referencia del proyecto de ClickUp Hoora.

- **Monitorización de hitos académicos y control de plazos (UAB):** La gestión temporal mediante ClickUp no solo se enfoca en el desarrollo técnico del software, sino que está calibrada para asegurar el estricto cumplimiento del calendario institucional fijado por la Universitat Autònoma de Barcelona (UAB). A través del establecimiento de hitos o *milestones* inamovibles dentro del tablero, el equipo puede evaluar en tiempo real la velocidad de trabajo durante los *sprints*, garantizando que los entregables se completen y validen con suficiente margen respecto a los plazos establecidos por la coordinación universitaria.

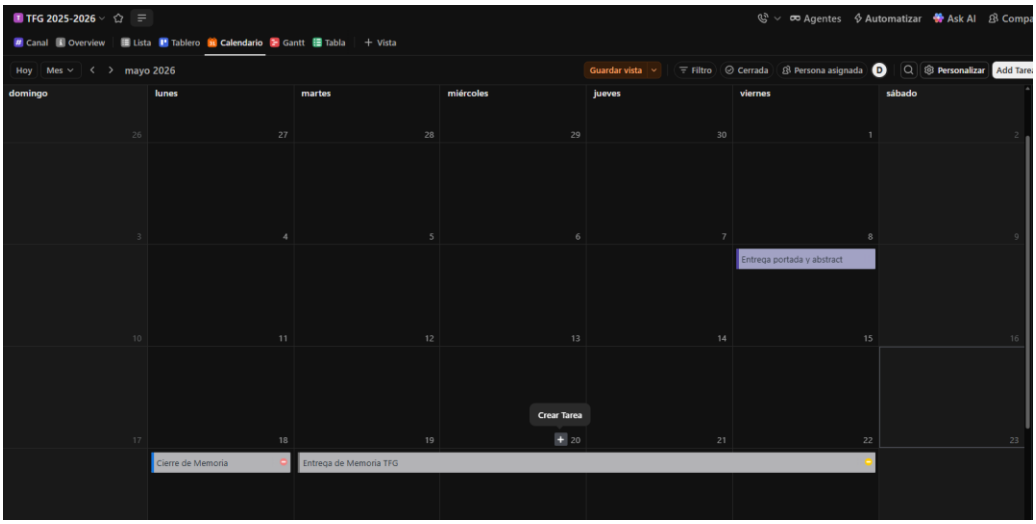


Figura 5.4 Imagen de referencia del proyecto de ClickUp Hoora.

- **Centralización documental y trazabilidad de requisitos:** Uno de los mayores desafíos en el desarrollo de software bajo acuerdos de confidencialidad (NDA) es evitar el aislamiento entre las especificaciones iniciales y la ejecución técnica. ClickUp funciona como un repositorio documental integrado que vincula directamente las actas de reunión y las historias de usuario (*User Stories*) pactadas con *119.media* con las tareas de programación en el *back-end* o *front-end*. Esto asegura una trazabilidad

bidireccional impecable: cada línea de código escrita o cada migración ejecutada en Supabase responde directamente a un requisito funcional previamente documentado y justificado en la plataforma de gestión.

5.3. Proceso de diseño y desarrollo

Tomando como base la hibridación metodológica del proyecto, el flujo de trabajo se estructura de manera secuencial e iterativa siguiendo las fases del Design Thinking (Gothelf & Seiden, 2016). Este enfoque garantiza que cada decisión técnica y arquitectónica esté supeditada a resolver una necesidad real del usuario, evitando el desarrollo de funcionalidades superfluas y optimizando el esfuerzo de ingeniería. El proceso se divide metodológicamente en tres grandes etapas interconectadas:

- Descubrimiento y Definición (Empatizar y Definir): Esta primera fase se centra en la inmersión profunda dentro del contexto operativo real de 119.media. A través de sesiones de consultoría y toma de requerimientos con los responsables de la organización (Lluís Domingo, Director creativo de 119.media), se analizaron las fricciones sistémicas del control horario y la gestión de proyectos. Toda la información cualitativa recopilada se tradujo formalmente en Historias de Usuario (User Stories), definiendo de forma unívoca el comportamiento esperado del sistema para cada perfil (Superadmin, Admin, Manager y User Basic). Esta definición acotó el alcance del Producto Mínimo Viable (MVP) y sentó las bases para el modelado de las tablas de datos en Supabase y las restricciones del control de accesos.

- Ideación y Prototipado (Idear y Prototipar): Con los requisitos funcionales claramente definidos, el proceso se trasladó al plano del diseño interactivo y la arquitectura de la información. Utilizando Figma como entorno de diseño colaborativo, se transformaron las abstracciones lógicas de las User Stories en esquemas visuales de baja fidelidad (wireframes) para validar la distribución de los componentes esenciales, como el temporizador por minutos o los formularios de quick-entry. Posteriormente, estos esquemas evolucionaron hacia un prototipo interactivo navegable de alta fidelidad. Este

prototipo permitió validar la navegación estructural, la cohesión estética de la paleta cromática (optimizada para Light Mode y Dark Mode mediante tokens de diseño) y la usabilidad predictiva de los pop-ups de validación antes de proceder a la escritura de una sola línea de código en el front-end.

- Implementación y Test (Evolución Ágil y Validación): La última fase representa la convergencia técnica del proyecto, donde las maquetas de Figma se materializan en código de producción a través del desarrollo full-stack. El motor configurado en una base de datos PostgreSQL, las funciones edge de invitación y el aislamiento de datos mediante políticas Row Level Security (RLS), los componentes codificados en React 18, TypeScript y Tailwind CSS, empaquetando todo el conjunto sobre la capa nativa y ligera de Tauri v2. Una vez construido el binario funcional del MVP, el proceso no concluyó de forma lineal, sino que se activó un Plan de Testeo y Validación Técnica. Se realizaron pruebas de funcionalidad, verificación de roles y tests de usabilidad.

Durante el desarrollo del presente proyecto se han incorporado de manera sistemática diversas herramientas de productividad asistida por inteligencia artificial generativa, principalmente para acelerar tareas repetitivas de redacción, generación de scaffolding de código y producción de figuras paramétricas. La utilización de estas herramientas responde a una lógica de eficiencia metodológica equivalente a la de los entornos de desarrollo integrado (IDE) o los gestores de paquetes: instrumentos que aceleran la ejecución pero no sustituyen la responsabilidad intelectual de los autores. El detalle exhaustivo de las herramientas utilizadas, el ámbito específico de aplicación, la delimitación de las áreas no asistidas y las garantías de supervisión humana se recogen íntegramente en el Anexo A del presente documento.

6. Análisis de Requisitos y Diseño

Este apartado detalla la arquitectura lógica y las especificaciones técnicas que definen el Producto Mínimo Viable (MVP). El diseño se ha centrado en resolver la necesidad de una visión global de costes temporales y carga por persona en 119.media.

6.1. Requisitos Funcionales

El núcleo de la aplicación se basa en la agilidad y la precisión del dato recogido:

- Registro de Proyectos y Tareas: Capacidad para dar de alta proyectos y desglosarlos en tareas específicas.
- Gestión de Tiempos: El sistema permite registrar horas por tarea, priorizando que el cómputo se realice en minutos y no en porcentajes para garantizar la exactitud de los informes.
- Visualización y Dashboards: Implementación de informes que muestran horas por proyecto, persona y periodo, permitiendo identificar proyectos que exceden el tiempo previsto.
- Asistente con IA: Módulo de análisis que ofrece recomendaciones automáticas sobre reasignación de recursos y detección de patrones de productividad.
- Compatibilidad: Desarrollo orientado a ser *cross-platform* para permitir el acceso desde distintos dispositivos.

Más allá de su enumeración, estos requisitos funcionales imprescindibles constituyen el contrato operativo entre la herramienta y el equipo de 119.media. Cada uno de ellos se ha derivado directamente de las fricciones detectadas durante la fase de descubrimiento y se ha priorizado siguiendo el método MoSCoW (Must, Should, Could, Won't), de modo que el esfuerzo de ingeniería se concentre primero en las capacidades nucleares que aportan valor inmediato —el registro atómico del tiempo y su visualización— y reserve para iteraciones posteriores aquellas funciones de alto valor pero menor criticidad, como la analítica prescriptiva mediante Inteligencia Artificial. La siguiente matriz formaliza este catálogo, asignando a cada

requisito un identificador único, una prioridad y el rol o actor que lo consume, lo que habilita la trazabilidad descrita en el capítulo de Objetivos.

ID	Requisito funcional	Prioridad	Actor principal
RF-01	Alta y gestión de proyectos y desglose en tareas específicas	Must	Manager / Admin
RF-02	Registro de tiempo medido en minutos y vinculado a proyecto y tarea	Must	Todos los roles
RF-03	Dashboards de horas por proyecto, persona y periodo	Must	Manager / Admin / Superadmin
RF-04	Asistente de IA con recomendaciones de reasignación de recursos	Should	Manager / Admin
RF-05	Alertas y pop-ups de validación de horas configurables	Must	Todos los roles
RF-06	Navegación y visibilidad de datos segmentada por rol	Must	Todos los roles
RF-07	Panel de bandeja (Tray) con formulario de quick-entry	Must	Todos los roles
RF-08	Invitación y gestión de miembros del workspace	Must	Superadmin / Admin
RF-09	Internacionalización ES/EN y cambio de tema Light/Dark	Could	Todos los roles

Figura 6.1 Tabla de requisitos de funcionalidad.

La priorización mediante MoSCoW no es un mero formalismo de gestión: condiciona el orden de implementación y, por tanto, el riesgo del proyecto. Los requisitos marcados como Must; registro atómico del tiempo, dashboards, alertas, navegación por rol e invitaciones. Constituyen el corazón del Producto Mínimo Viable y deben estar plenamente operativos para que la herramienta aporte valor desde el primer día. Los requisitos Should, como la analítica prescriptiva mediante Inteligencia Artificial, aportan la ventaja competitiva diferencial, pero pueden incorporarse en iteraciones posteriores sin comprometer la operatividad básica. Finalmente, los requisitos Could, como la internacionalización completa o la conmutación de tema, mejoran la experiencia sin ser críticos. Esta gradación permite asignar el esfuerzo de ingeniería de forma proporcional al valor entregado y mitiga el riesgo de dispersión en funcionalidades accesorias.

6.2. Requisitos No Funcionales

Si los requisitos funcionales definen qué debe hacer el sistema, los requisitos no funcionales (RNF) determinan con qué nivel de calidad debe hacerlo, y son precisamente estos atributos los que diferencian un prototipo académico de una herramienta apta para operar en un entorno corporativo regido por un acuerdo de confidencialidad. Para su clasificación se ha adoptado como referencia el modelo de calidad de producto de la norma ISO/IEC 25010, que estructura la calidad del software en características medibles. En el contexto de 119.media, la seguridad y la confidencialidad se erigen como el atributo crítico de carácter no negociable, dado que la herramienta manipula la estructura de costes de personal y la planificación de proyectos estratégicos.

El rendimiento constituye el segundo pilar: la elección de un cliente nativo ligero compilado sobre Tauri v2 y Rust responde al requisito de no penalizar las estaciones de trabajo de los perfiles creativos y técnicos, evitando el consumo masivo de memoria característico de las alternativas basadas en Chromium. La usabilidad, entendida como la minimización de la carga cognitiva durante el registro, condiciona transversalmente el diseño de la interfaz y del panel de bandeja. A estos se suman la portabilidad multiplataforma, la mantenibilidad garantizada por la organización en monorepo, la internacionalización nativa y la fiabilidad respaldada por las propiedades ACID de PostgreSQL. La siguiente tabla sintetiza los RNF prioritarios junto con el criterio objetivo que permite verificar su cumplimiento.

Categoría (ISO/IEC 25010)	Requisito no funcional	Criterio de verificación
Seguridad	Aislamiento multi-tenant y confidencialidad de datos NDA	Políticas RLS activas en todas las tablas; tokens de sesión en almacén seguro nativo (Keychain)
Eficiencia de rendimiento	Cliente de escritorio nativo ligero	Consumo de recursos muy por debajo de una alternativa Electron; arranque y respuesta inmediatos
Usabilidad	Carga cognitiva mínima en el registro de horas	Registro completado en pocos clics; quick-entry desde el Tray sin abrir la ventana principal
Portabilidad	Ejecución multiplataforma	Binario firmado por sistema operativo (macOS; Windows aplazado) desde una misma base de código
Mantenibilidad	Arquitectura modular y desacoplada	Monorepo pnpm con separación de app, landing y librerías compartidas
Compatibilidad / i18n	Soporte multilingüe	Catálogos completos ES/EN intercambiables sin recargar la aplicación
Fiabilidad	Integridad de los datos registrados	Restricción de no negatividad en la duración a nivel de base de datos; transacciones ACID

Figura 6.2 Tabla de requisitos de funcionalidad.

El requisito de seguridad merece una consideración especial por su carácter transversal. No se satisface con una única medida, sino con una defensa en profundidad que combina el aislamiento a nivel de fila en la base de datos, el almacenamiento criptográfico de los tokens de sesión en el llavero del sistema operativo, la delegación de las operaciones privilegiadas en funciones de servidor y el endurecimiento de la configuración del cliente nativo. Cada una de estas capas es necesaria, pero ninguna es suficiente por sí sola; su combinación es la que permite afirmar que el sistema cumple los compromisos del acuerdo de confidencialidad incluso ante el fallo de alguna de ellas.

El requisito de rendimiento, por su parte, se verifica de forma comparativa frente al paradigma dominante. Mientras que una aplicación equivalente basada en Electron embarca una instancia completa de navegador por proceso, con el consiguiente coste de memoria, el cliente compilado sobre Tauri v2 delega el renderizado en el motor web nativo del sistema y reserva la lógica de control para un binario en Rust de huella reducida. Esta diferencia, lejos de ser anecdótica, sostiene tanto la promesa de no penalizar las estaciones de trabajo como la viabilidad económica del modelo de negocio, al trasladar el grueso del cómputo a la máquina del usuario y minimizar el coste de infraestructura en la nube.

6.3. Casos de Uso del Sistema

El comportamiento esperado del sistema se ha modelado mediante casos de uso, técnica que describe las interacciones entre los actores y la aplicación orientadas a la consecución de un objetivo concreto. Los actores se corresponden con la matriz de cuatro roles de la organización (Superadmin, Admin, Manager y User Basic), de modo que cada caso de uso queda acotado a las capacidades que su nivel de responsabilidad le confiere. Este modelado cumple una doble función: por un lado, traduce las Historias de Usuario pactadas con 119.media en especificaciones verificables; por otro, sirve de puente directo hacia el diseño de la navegación protegida por rol y el modelo de datos relacional descritos en las subsecciones siguientes.

El diagrama de casos de uso ofrece una visión global de las relaciones actor–funcionalidad, evidenciando cómo las capacidades de los perfiles superiores incorporan y extienden las de los inferiores. A continuación, se enumeran los casos de uso principales que articulan el Producto Mínimo Viable.

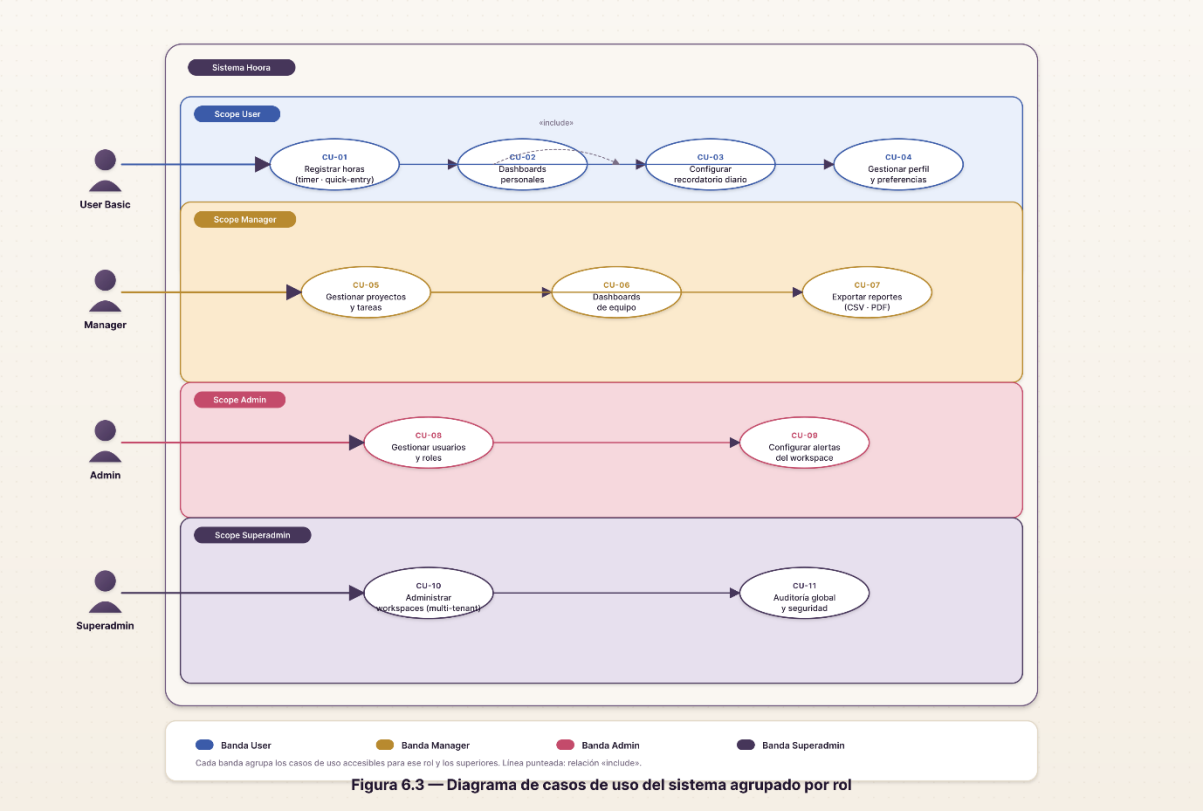


Figura 6.3 — Diagrama de casos de uso del sistema agrupado por rol

Figura 6.3 Diagrama de casos de uso del sistema agrupado por rol.

ID	Actor principal	Descripción del caso de uso
CU-01	User Basic	Registrar horas iniciando/pausando el temporizador o introduciendo minutos sobre un proyecto y tarea
CU-02	User Basic	Validar o corregir el registro a través del pop-up disparado por la alerta
CU-03	User Basic	Configurar el recordatorio diario fijando la hora del aviso en Ajustes
CU-04	Manager	Consultar dashboards de horas por proyecto, persona y periodo
CU-05	Manager	Crear proyectos y desglosarlos en tareas asignables
CU-06	Admin / Superadmin	Invitar a un nuevo miembro al workspace mediante la Edge Function
CU-07	Superadmin	Gestionar los usuarios del sistema y sus roles efectivos
CU-08	Todos los roles	Alternar tema (Light/Dark) e idioma (ES/EN) de la interfaz

Figura 6.4 Tabla de actores y caso de uso.

El caso de uso nuclear es CU-01 (registrar horas). Su flujo principal arranca cuando el trabajador selecciona el proyecto activo y acciona el temporizador desde la cabecera o introduce directamente los minutos en el formulario de quick-entry del Tray; el sistema persiste el registro vinculándolo a su identidad, fecha, proyecto y tarea, y actualiza en tiempo real las métricas del dashboard. El flujo alternativo CU-02 se activa cuando, llegada la hora configurada, la aplicación emite una alerta y despliega un pop-up que permite validar o ajustar el tiempo sin abandonar la tarea en curso, reduciendo a segundos una operación que los sistemas tradicionales relegan al final de la jornada.

En el plano administrativo, CU-06 (invitar miembro) ilustra la segregación de privilegios: la invitación no se resuelve en el cliente, sino que se delega en una función de servidor que opera con clave de servicio para crear el invitado y registrarlo en el workspace sin exponer credenciales privilegiadas, garantizando que la incorporación de personal respete el aislamiento multi-inquilino. Esta separación entre lo que el usuario puede solicitar y lo que el servidor ejecuta con privilegios elevados es coherente con los requisitos no funcionales de seguridad y con la arquitectura de permisos detallada seguidamente.

El caso de uso CU-03 (configurar recordatorio) habilita la captura proactiva: el usuario fija en Ajustes la hora a la que desea recibir el aviso diario, valor que el sistema persiste en sus preferencias y honra incluso con la ventana principal oculta en la bandeja. CU-04 (consultar dashboard) es el principal punto de contacto del perfil Manager con la herramienta: le permite contrastar las horas reales consumidas frente a lo previsto, por proyecto, persona y periodo, y detectar desviaciones presupuestarias antes de que se conviertan en pérdidas.

CU-05 (crear proyecto y tareas) materializa la estructura sobre la que se imputan las horas: el Manager da de alta el proyecto, lo desglosa en tareas y asigna responsables, generando las entidades del modelo de datos que después consumen los registros de tiempo. CU-07 (gestionar usuarios y roles), reservado al Superadmin, cierra el ciclo administrativo al permitir alta, baja y reasignación de roles de los miembros del workspace. La articulación de estos casos de uso demuestra que cada acción de la interfaz responde a un actor, un objetivo y un conjunto de entidades de datos perfectamente delimitados.

6.4. Sistema de Alertas y UX Predictiva

Para fomentar el hábito del registro sin interrumpir el flujo de trabajo, se han diseñado mecánicas de interacción específicas:

- Alarmas Personalizadas: Cada usuario o el administrador puede configurar alertas a horas específicas para el registro.
- Validación mediante Pop-ups: Al activarse la alerta, el sistema muestra una ventana emergente que permite validar o introducir las horas directamente sobre el proyecto y tarea activos.
- Vista de Calendario: Integración de una interfaz de calendario para visualizar la distribución temporal de las tareas.

El diseño de estas mecánicas parte de una premisa extraída de la literatura de usabilidad: todo punto de fricción en el registro reduce la pureza del dato recogido. Por ello, la UX predictiva no se limita a recordar la tarea pendiente, sino que anticipa la acción más probable del usuario y la presenta preconfigurada en el pop-up, de manera que la validación se resuelva con una única interacción. La hora de disparo es parametrizable por el propio usuario o por el administrador, lo que permite alinear el aviso con las ventanas temporales donde la organización experimenta mayor fragmentación de tareas, respetando los ritmos de concentración profunda de los perfiles creativos.

La vista de calendario complementa este flujo aportando una representación temporal de la distribución de tareas, que facilita la detección visual de huecos sin registrar y la planificación de la jornada. En conjunto, alarmas, pop-ups de validación y calendario conforman un sistema de captura proactiva cuyo objetivo último es transformar el registro de horas de una carga administrativa pasiva en un hábito de baja fricción integrado de forma natural en el flujo de trabajo diario.

6.5. Arquitectura de Roles y Permisos

El acceso a la información está estrictamente segmentado para proteger la privacidad de los datos según el nivel de responsabilidad:

- Superadmin: Gestión integral de los 13 usuarios del sistema.

- Admin: Administración técnica de las funcionalidades de la aplicación.
- Manager: Capacidad de supervisión total de proyectos y creación de nuevos flujos de trabajo.
- User Basic: Acceso limitado a sus propios proyectos y gestión de sus registros horarios personales.

Esta jerarquía de cuatro niveles no es un mero esquema organizativo, sino que se proyecta técnicamente sobre dos capas complementarias: la navegación del cliente, mediante el componente de rutas protegidas (ProtectedRoute) que condiciona las vistas accesibles al rol efectivo, y el motor de la base de datos, donde las políticas de Seguridad a Nivel de Fila garantizan que la restricción de visibilidad no dependa exclusivamente de la interfaz. El rol efectivo de cada usuario se obtiene de la tabla de pertenencia al workspace, de modo que un mismo individuo puede ostentar roles distintos en organizaciones distintas. La siguiente matriz formaliza qué acciones tiene permitidas cada perfil sobre los recursos críticos del sistema.

Recurso / Acción	**Superadmin**	**Admin**	**Manager**	**User Basic**
Gestionar usuarios del sistema	Sí	Parcial	No	No
Invitar miembros al workspace	Sí	Sí	No	No
Crear y editar proyectos y tareas	Sí	Sí	Sí	No
Ver costes y horas de todo el equipo	Sí	Sí	Sí	No
Consultar dashboards de empresa	Sí	Sí	Sí	No
Registrar y editar horas propias	Sí	Sí	Sí	Sí
Configurar alertas y recordatorios propios	Sí	Sí	Sí	Sí

Figura 6.5 Tabla de recursos por usuario.

La doble capa de aplicación de los permisos (navegación en el cliente y políticas en la base de datos) responde a un principio de seguridad defensiva: la restricción visible en la interfaz mejora la experiencia al ocultar lo que el usuario no puede usar,

pero nunca debe ser la única barrera, pues el cliente es manipulable. Por ello, el componente de rutas protegidas determina qué vistas se renderizan según el rol efectivo, mientras que las políticas RLS garantizan que, aun cuando una petición llegase a la base de datos eludiendo la interfaz, jamás devolvería filas fuera del ámbito del usuario y su workspace. El rol efectivo no se asume del cliente, sino que se hidrata desde la tabla de pertenencia tras el inicio de sesión, fuente única de verdad para la autorización.

6.6. Modelo de Datos

El modelo de datos relacional es el cimiento sobre el que descansan tanto la integridad de los registros como el aislamiento multi-inquilino exigido por el NDA. Se ha versionado íntegramente mediante migraciones SQL, lo que asegura su reproducibilidad y trazabilidad a lo largo del ciclo de vida del proyecto. La entidad raíz es el workspace, que representa a cada organización o tenant; la tabla de pertenencia vincula a los usuarios con sus workspaces y el rol asociado, habilitando la arquitectura en la que múltiples empresas conviven sobre la misma infraestructura física sin riesgo de interferencia.

A partir de esa raíz se despliegan las entidades de dominio: los perfiles públicos derivados del sistema de autenticación, los proyectos y sus tareas, las tareas internas de empresa y, como núcleo analítico, los registros de tiempo, que almacenan las horas dedicadas por usuario, fecha, proyecto o tarea y duración. Sobre la duración se aplica una restricción de no negatividad a nivel de base de datos, garantía de que ningún dato corrupto contamine los informes. La distinción entre registros de tiempo de proyecto y registros de tareas internas permite separar la actividad facturable de la gestión interna sin perder granularidad. El diagrama entidad-relación siguiente representa estas tablas y sus cardinalidades.

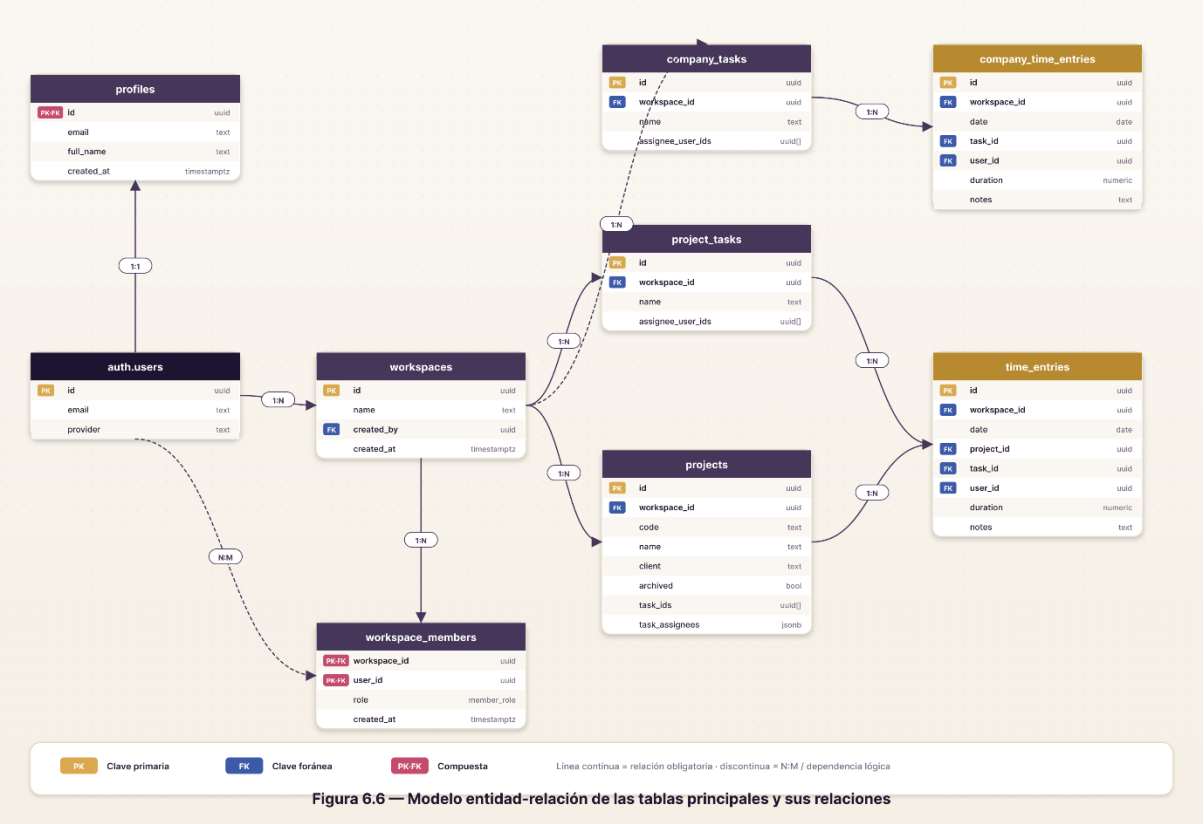


Figura 6.6 Modelo entidad-relación de las tablas principales y sus relaciones.

Tabla / Entidad	Propósito	RLS
workspaces	Organización o tenant; raíz del aislamiento multi-inquilino	Sí
workspace_members	Pertenencia usuario–workspace y rol efectivo asociado	Sí
profiles	Perfil público mostrado, derivado de auth.users	Sí
projects	Proyectos catalogados dentro de cada workspace	Sí
project_tasks	Tareas específicas asociadas a cada proyecto, con asignación de usuarios	Sí
company_tasks	Tareas internas de empresa (RRHH, reuniones, administración)	Sí
time_entries	Horas por usuario, fecha, proyecto/tarea y duración	Sí
company_time_entries	Horas dedicadas a tareas internas de empresa	Sí

Figura 6.7 Tabla de espacios y propósitos de cada espacio.

Para que las políticas RLS evaluasen la pertenencia sin incurrir en recursión infinita (un problema real que llegó a romper el inicio de sesión) se introdujeron funciones auxiliares de comprobación de membresía, versionadas en una migración específica. Este detalle, ampliado en el capítulo de implementación, ilustra cómo el diseño del modelo de datos y el de la seguridad se condicionan mutuamente y deben abordarse de forma conjunta.

La integridad de los datos se garantiza a nivel de motor mediante las propiedades ACID de PostgreSQL y mediante restricciones declarativas, como la no negatividad de la duración de los registros de tiempo, que impide la inserción de valores imposibles con independencia de la lógica del cliente. Las cardinalidades del modelo reflejan la realidad del dominio: un workspace agrupa a múltiples miembros y proyectos; un proyecto se desglosa en múltiples tareas; y cada registro de tiempo pertenece a un único usuario y a una única tarea o proyecto, lo que permite agregaciones precisas por cualquiera de esas dimensiones.

La separación entre registros de tiempo de proyecto y registros de tareas internas de empresa no es redundante, sino que refleja dos naturalezas distintas de actividad: la imputable a cuentas y proyectos (potencialmente facturable y sujeta a control presupuestario) y la de gestión interna, como reuniones, recursos humanos o administración. Mantenerlas en entidades diferenciadas permite calcular con exactitud la rentabilidad de cada proyecto sin que la actividad estructural de la organización distorsione la métrica.

6.7. Entorno de Datos (Contexto 119.media)

La plataforma está preparada para gestionar la estructura organizativa real de la empresa, incluyendo áreas como:

- Proyectos Estratégicos: Winter25, Pla estratègic.
- Gestión Interna: RRHH, Team Building, Reunió equip, Admin.
- Áreas Operativas: IT, Web, XXSS, Comercial, Direcció Projectes.

Esta taxonomía no está embebida de forma rígida en el código, sino que se modela como datos configurables del workspace, lo que constituye la base de la capacidad de personalización y marca blanca descrita en el modelo de negocio. En el escenario de estudio, las áreas y proyectos reflejan la estructura operativa real de 119.media, pero la misma arquitectura permite reconfigurar por completo la taxonomía para otras agencias o consultoras sin alterar el núcleo del sistema. Todos los nombres de proyectos y áreas empleados en esta memoria se presentan anonimizados o como ejemplos representativos, en cumplimiento del acuerdo de confidencialidad.

Esta capacidad de reconfiguración de la taxonomía sin tocar el código se apoya en que las áreas, los proyectos y las tareas internas se modelan como datos del workspace y no como constantes embebidas. En consecuencia, migrar la herramienta de una agencia de comunicación a un estudio de arquitectura o a una consultora tecnológica se reduce a poblar esas entidades con la estructura del nuevo cliente, preservando intacta la lógica de registro, agregación y seguridad. Es precisamente esta parametrización la que convierte un desarrollo a medida para 119.media en una plataforma replicable, fundamento de la proyección SaaS del proyecto.

6.8. Diseño de Privacidad y Confidencialidad

Dado que el proyecto se desarrolla bajo un NDA, el diseño contempla medidas de seguridad estrictas:

- Anonimización: Uso de dades ficticias o anonimizadas en la redacción de la memoria para evitar la publicación de información sensible.
- Acceso Restringido: Implementación de protocolos de seguridad para asegurar que solo los perfiles autorizados accedan a los datos de costes y carga de trabajo.

La privacidad no se aborda como una capa añadida a posteriori, sino como un requisito de diseño que permea todas las decisiones arquitectónicas. El aislamiento de los datos se garantiza matemáticamente en el motor de la base de datos a través

de las políticas RLS, de modo que ninguna consulta pueda devolver filas pertenecientes a un workspace ajeno, con independencia de posibles fallos en la capa de presentación. A ello se suma el endurecimiento del almacenamiento de los tokens de sesión en el llavero criptográfico del sistema operativo y la delegación de las operaciones privilegiadas, como las invitaciones, en funciones de servidor que nunca exponen claves sensibles al cliente.

En el plano documental, la redacción de esta memoria aplica una política estricta de anonimización de los datos reales de costes, rendimiento y proyectos estratégicos. Los ejemplos cuantitativos y los nombres de cuentas se sustituyen por valores ficticios o representativos que preservan el rigor expositivo sin comprometer la propiedad intelectual ni los secretos comerciales de la organización, honrando así el marco jurídico del NDA bajo el que se desarrolla el proyecto.

En síntesis, el análisis de requisitos y el diseño no se han abordado como una fase documental previa y desvinculada del código, sino como el armazón lógico que predetermina cada decisión de implementación. Los requisitos funcionales y no funcionales, los casos de uso, la matriz de permisos y el modelo de datos forman un sistema coherente en el que la confidencialidad exigida por el NDA, la usabilidad reclamada por los perfiles creativos y la precisión contable requerida por la dirección se traducen, sin pérdidas, en estructuras concretas de software. El capítulo siguiente expone cómo este diseño se ha materializado en un prototipo funcional verificable.

7. Implementación Técnica (Avance)

En esta fase del proyecto, se ha consolidado el desarrollo de la interfaz de usuario y la lógica de visualización de datos, resultando en un prototipo funcional que permite la gestión del tiempo en tiempo real. La implementación se divide en dos ejes principales: la arquitectura del *front-end* y el sistema de visualización de métricas.

7.1. Interfaz de Usuario (UI) y Tematización

Siguiendo los principios de diseño definidos en la fase de prototipado, se han implementado dos modos visuales para mejorar la adaptabilidad del software al entorno de trabajo de 119.media:

- Modo Oscuro (Dark Mode): Diseñado para reducir la fatiga visual en entornos de baja iluminación, utilizando una paleta de colores basada en tonos púrpura y carbonos profundos. Este modo resalta los indicadores de actividad mediante contrastes en tonos cálidos (naranjas y oros).
- Modo Claro (Light Mode): Orientado a la máxima legibilidad y claridad durante el día, empleando fondos neutros que facilitan la lectura de reportes densos y estructuras de proyectos complejos.
- Diseño Responsivo: Ambos temas mantienen una estructura de navegación lateral persistente que garantiza una experiencia de usuario (UX) coherente y rápida para el registro de horas.

Esta tematización no se ha resuelto mediante hojas de estilo duplicadas, sino a través de una capa de design tokens centralizada en el archivo tokens.css, compartida en el monorepo con el paquete de interfaz común. Al abstraer la paleta cromática, las tipografías y los espaciados en variables semánticas, el cambio entre Modo Claro y Modo Oscuro se reduce a la conmutación de un conjunto de tokens, sin tocar la lógica de los componentes. Esta decisión arquitectónica no solo garantiza la coherencia visual entre la aplicación de escritorio y la landing pública, sino que constituye el fundamento técnico de la capacidad de marca blanca descrita en el modelo de negocio: personalizar la identidad visual de un cliente corporativo equivale a redefinir ese archivo de tokens, sin alterar una sola línea del núcleo en React.

La estructura de navegación lateral persistente, materializada en componentes reutilizables (MainLayout, Sidebar y Topbar), mantiene siempre accesibles el temporizador y los accesos principales, minimizando el número de clics necesarios para registrar tiempo y reforzando el requisito no funcional de usabilidad.

7.2. Dashboards y Visualización de Datos

El panel de "Resumen" se ha configurado para ofrecer una visión global inmediata de los KPIs críticos detectados en la fase de requisitos:

- Métricas en Tiempo Real: Visualización de horas totales de la semana, proyectos activos y porcentaje de utilización del equipo.
- Distribución de Proyectos: Un gráfico de barras desglosa el uso del tiempo por áreas clave (ej. Hoora Core, Portal Cliente, Backoffice), permitiendo identificar desviaciones presupuestarias temporales.
- Actividad Reciente: Un flujo de eventos que registra las últimas acciones de los miembros del equipo, mejorando la transparencia operativa.

El diseño de estos cuadros de mando aplica la teoría de lectura jerárquica de la información: una capa macro orientada a la toma de decisiones directivas (visible para Manager, Admin y Superadmin) y una capa micro centrada en la autogestión de la productividad del propio trabajador (User Basic), evitando la saturación informativa. Los indicadores se alimentan en tiempo real de los registros persistidos, de modo que cualquier hora introducida desde la cabecera o desde el panel de bandeja se refleja de inmediato en las métricas, cerrando el bucle entre captura y análisis sin pasos intermedios de exportación.

7.3. Funcionalidad de Registro (Temporizador)

Se ha implementado el motor de registro de horas preciso, permitiendo que el usuario inicie, pause o detenga el cronómetro directamente desde la cabecera de la aplicación. Este sistema se vincula automáticamente al proyecto seleccionado, cumpliendo con el requisito de registrar tiempos en minutos para una mayor precisión contable.

El estado del cronómetro se gestiona mediante un contexto de React especializado (TimeTrackingContext), que mantiene sincronizada la sesión de registro activa en todas las vistas de la aplicación, incluida la cabecera y el panel de bandeja. La elección de medir el tiempo en minutos reales, en lugar de en repartos porcentuales, materializa el requisito RF-02 y erradica el sesgo de planificación descrito en el marco conceptual: el dato registrado refleja la fricción real de la jornada y no una estimación idealizada a posteriori.

El control del temporizador desde la cabecera convive con el del panel de bandeja sin duplicar estado: ambos puntos de entrada operan sobre el mismo contexto, de modo que iniciar el cronómetro en uno se refleja instantáneamente en el otro. Esta unicidad de la fuente de verdad del tiempo activo evita uno de los errores más frecuentes en las herramientas de time-tracking, el doble conteo derivado de sesiones de registro paralelas e inconsistentes.

7.4. Stack tecnológico y arquitectura del cliente

La aplicación se ha implementado como una solución de escritorio nativa multi-plataforma construida sobre Tauri v2, lo que permite distribuir un binario firmado por sistema operativo consumiendo recursos muy por debajo de una alternativa basada en Electron. El front-end se ha desarrollado con React 18 y TypeScript, empaquetado mediante Vite, y los estilos se han construido con Tailwind CSS apoyados en una capa de design tokens (tokens.css) que centraliza la paleta cromática, tipografías y espaciados compartidos con el paquete de UI común.

Las piezas clave del stack son las siguientes:

- Capa nativa: Tauri v2 con módulos Rust para la integración con el sistema operativo, gestión de la ventana principal, menú de bandeja (tray) y plugins de notificaciones, deep-link, opener y store.
- Capa de presentación: React Router con un esquema de rutas protegidas por rol, layouts reutilizables (MainLayout, Sidebar, Topbar) y vistas dedicadas para Dashboard, Empresa, Calendario, Proyectos, Reportes, Equipo, Ajustes y Tray Panel.

- Capa de estado: contextos de React especializados — AuthContext, ThemeContext, I18nContext y TimeTrackingContext — que mantienen sincronizada la sesión, el tema activo, el idioma y el cronómetro de horas en todas las vistas.
- Capa de servicios: cliente HTTP propio, cliente Supabase tipado, almacén seguro de tokens de sesión, servicio de notificaciones, dispatcher común de acciones de menú/tray y módulo de almacenamiento de preferencias del usuario.
- Organización en monorepo: gestionado con pnpm workspaces, separando la aplicación de escritorio (apps/desktop, paquete @hoora/native), la landing pública (apps/web, paquete @hoora/web) y librerías compartidas (packages/shared-ui, packages/utils).

La siguiente tabla resume la responsabilidad de cada capa del stack tecnológico, evidenciando la separación limpia entre la presentación, el estado, los servicios y la capa nativa que interactúa con el sistema operativo.

Capa	**Tecnología**	**Función principal**
Nativa	Tauri v2 + Rust	Ciclo de vida de la ventana, menú de bandeja y plugins de sistema
Presentación	React 18 + TypeScript + React Router	Vistas y navegación protegida por rol
Build	Vite	Empaquetado y servidor de desarrollo del front-end
Estilos	Tailwind CSS + tokens.css	Tematización Light/Dark y design tokens compartidos
Estado	Context API (Auth, Theme, I18n, TimeTracking)	Sesión, tema, idioma y cronómetro
Servicios	Cliente Supabase, cliente HTTP, secure store, notificaciones	Datos, autenticación, almacenamiento y avisos
Backend	Supabase (PostgreSQL, Auth, Storage, Edge Functions)	Persistencia, autenticación y lógica de servidor
Monorepo	pnpm workspaces	apps/desktop, apps/web y packages compartidos

Figura 7.1 Tabla de capas, tecnología y datos.

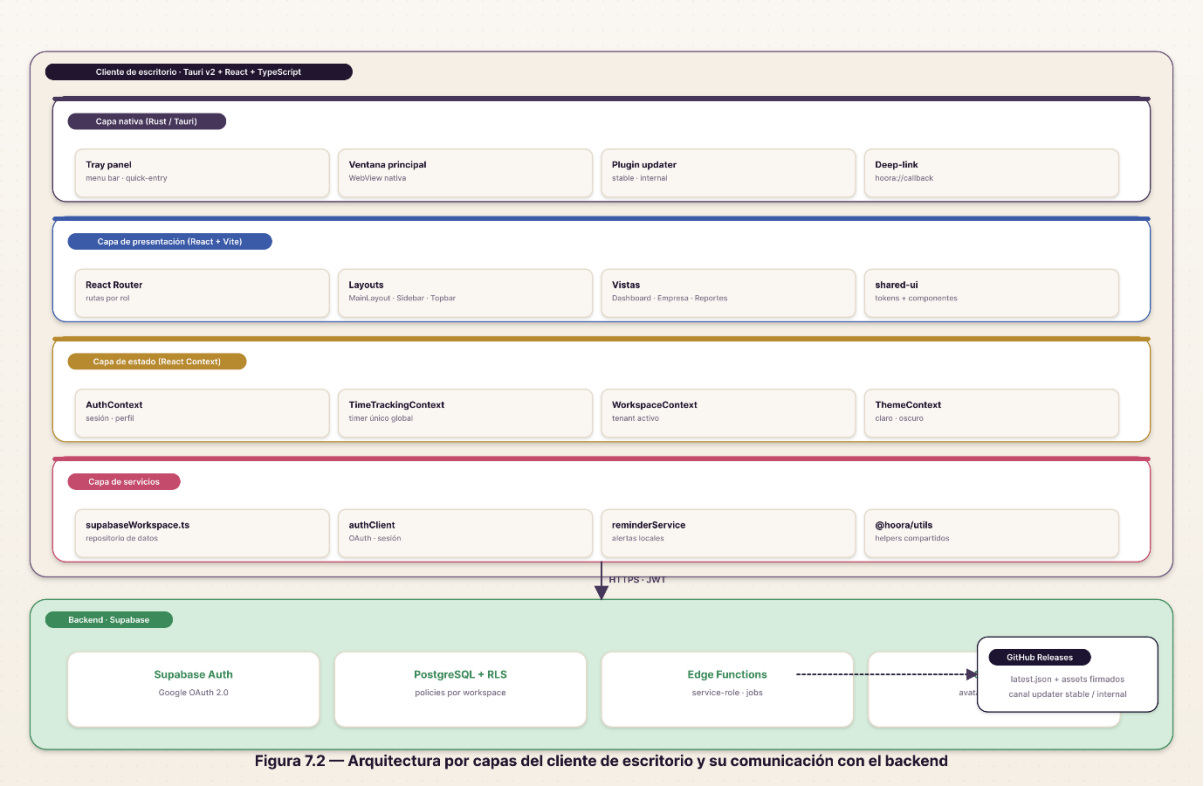


Figura 7.2 — Arquitectura por capas del cliente de escritorio y su comunicación con el backend

Figura 7.2 Arquitectura por capas del cliente de escritorio y su comunicación con el backend.

En la capa nativa se registran únicamente los plugins de Tauri estrictamente necesarios, siguiendo el principio de mínima superficie de ataque adoptado en la fase de endurecimiento. Cada plugin se acompaña de la capability mínima que habilita su uso, evitando exponer permisos no requeridos.

Plugin nativo	**Función y scope**
tauri-plugin-deep-link	Recepción del callback OAuth (esquema hoora://); requiere la capability deep-link:default
tauri-plugin-single-instance (feature deep-link)	Reenvía la URL del AppleEvent desde la segunda instancia a la primera vía IPC
tauri-plugin-notification	Notificaciones nativas del sistema operativo
tauri-plugin-opener	Apertura de enlaces y rutas externas (sustituye al plugin shell deprecado)
tauri-plugin-store	Persistencia de preferencias del usuario
Módulo Rust de secure storage	Persistencia del token de sesión en el llavero nativo (Keychain)

Figura 7.3 Tabla de plugins y funciones.

7.5. Backend Supabase, multi-tenant y RLS

El back-end se ha materializado sobre Supabase, que aporta una base de datos PostgreSQL gestionada, autenticación, almacenamiento y funciones edge en un mismo entorno cloud. Esta decisión técnica permite cubrir las necesidades del MVP sin desplegar infraestructura propia, manteniendo a la vez la puerta abierta al modelo de hosting híbrido descrito en el modelo de negocio.

El modelo de datos se ha versionado mediante migraciones SQL (carpeta supabase/migrations) e incluye, entre otras, las siguientes tablas y entidades:

- Workspaces y workspace_members: soportan la arquitectura multi-tenant; un mismo usuario puede pertenecer a varios workspaces con un rol asociado.
- Profiles: derivada de auth.users, almacena el perfil público mostrado en la aplicación.
- Projects, project_tasks y company_tasks: catalogan los proyectos del workspace, sus tareas asociadas y las tareas internas de empresa, con asignación de usuarios por tarea.
- Time_entries y company_time_entries: registran las horas dedicadas por usuario, fecha, proyecto/tarea y duración, con verificación de no negatividad a nivel de base de datos.

Sobre estas tablas se han activado políticas Row Level Security (RLS) que restringen automáticamente la visibilidad de cada fila al workspace al que pertenece y, donde corresponde, al usuario propietario del registro. Se han añadido funciones auxiliares (helpers de pertenencia) para evitar recursión en las políticas y garantizar un comportamiento estable tras el login. Estas funciones se mantienen en la migración 20260405124500_fix_rls_membership_helpers.sql y posteriores.

Las invitaciones a nuevos miembros del workspace se realizan a través de una Edge Function dedicada (invite-workspace-member), que actúa con clave de servicio en el servidor para crear el invitado y registrarlo como miembro del workspace sin exponer credenciales privilegiadas al cliente. Existe también un script de migración (scripts/migrate-mock-to-supabase.mjs) que ha permitido portar los datos iniciales del entorno mock al proyecto real de 119.media.

El aislamiento multi-inquilino reposa íntegramente sobre las políticas de Seguridad a Nivel de Fila. Durante el desarrollo se detectó un problema real de recursión en estas políticas que, al evaluar la pertenencia de un usuario consultando la propia tabla protegida, desencadenaba un error de profundidad de pila (stack depth limit exceeded) que rompía el post-login. La solución consistió en extraer la comprobación de membresía a funciones auxiliares, eliminando la recursión y garantizando un comportamiento estable tras el inicio de sesión; la corrección quedó versionada en su propia migración y aplicada al proyecto remoto. Este incidente ilustra por qué la seguridad a nivel de base de datos exige un diseño tan cuidadoso como el de la lógica de aplicación.

Las invitaciones a nuevos miembros se resuelven mediante una Edge Function dedicada que opera con clave de servicio en el lado del servidor, creando el invitado y registrándolo como miembro del workspace sin exponer credenciales privilegiadas al cliente. El versionado del esquema mediante migraciones SQL aporta reproducibilidad y trazabilidad; la siguiente tabla enumera las migraciones aplicadas y su cometido.

Migración	**Cometido**
20260331140000_init.sql	Esquema inicial: workspaces, miembros, perfiles, proyectos, tareas y registros de tiempo
20260405124500_fix_rls_membership_helpers.sql	Helpers de pertenencia que corrigen la recursión RLS del post-login
20260418143000_add_bootstrap_current_user.sql	Bootstrap del usuario actual para la resolución de sesión

Figura 7.4 Tabla de migraciones.

7.6. Sistema de alertas, recordatorios y panel de bandeja (Tray)

Para no obligar al usuario a tener la aplicación siempre en primer plano, se ha implementado una política de ventana "tray-first": al cerrar la ventana principal, la aplicación se oculta y permanece accesible desde la bandeja del sistema, evitando interrumpir el flujo de trabajo. El cierre real solo se produce desde la acción explícita de Salir.

El sistema de alertas y recordatorios se compone de los siguientes elementos:

- Recordatorio diario configurable desde Ajustes: el usuario define la hora a la que desea recibir el aviso para registrar las horas.
- Disparo preciso al cambio de minuto: el recordatorio se evalúa al cruzar el límite de minuto correspondiente, en lugar de mediante un polling cada 30 segundos, reduciendo carga y mejorando la exactitud.
- Panel de bandeja (Tray Panel): al hacer clic en el icono del tray se despliega un panel React anclado a la barra de menús con accesos rápidos, métricas resumidas y un formulario de quick-entry para registrar horas sin abrir la ventana principal.
- Menú nativo del tray: comparte un único dispatcher de acciones con el panel React (Abrir Hoora, Buscar actualizaciones, Ajustes, Salir), garantizando que ambos puntos de entrada se comporten de forma idéntica.
- Notificaciones nativas del sistema operativo emitidas a través del plugin oficial de Tauri, alineadas a la versión JS y Rust, con traducción al español.

La política de ventana tray-first responde a una necesidad de los entornos de trabajo creativos: la aplicación no debe competir por el primer plano ni interrumpir el estado de concentración. El disparo del recordatorio al cruzar el límite exacto de minuto, en lugar de mediante un sondeo periódico cada treinta segundos, no es un detalle menor: reduce la carga de cómputo en reposo y elimina la deriva temporal que acumulaba el modelo de polling, alineando el aviso con la precisión por minutos que exige el motor de registro. El hecho de que el menú nativo del tray y el panel React compartan un único dispatcher de acciones garantiza que ambos puntos de entrada se comporten de forma idéntica, evitando divergencias de estado entre la interfaz y la capa nativa.

7.7. Autenticación y almacenamiento seguro

La autenticación se delega en Supabase Auth y soporta inicio de sesión mediante correo y contraseña, con preparación para flujos OAuth (Google) a través de una página puente publicada en la landing que reenvía el callback al binario de escritorio mediante deep-link. Los mensajes de error de autenticación se han traducido y normalizado en español para ofrecer una experiencia consistente al usuario final.

La gestión del token de sesión se ha endurecido para builds empaquetadas:

- En builds reales, el token se persiste en almacenamiento seguro nativo (Keychain en macOS) a través de un módulo dedicado en Rust.
- En desarrollo (tauri dev), se utiliza localStorage como fallback para evitar fricciones con Keychain.
- Se ha implementado una migración automática desde el antiguo localStorage al almacén seguro, garantizando que las sesiones existentes no se pierdan tras la actualización.

Tras el login, la aplicación resuelve el workspace activo del usuario con una heurística que prioriza el workspace con mayor número de miembros, evitando caer por defecto en workspaces personales vacíos creados por importaciones o invitaciones anteriores. El rol efectivo se obtiene de la tabla workspace_members y condiciona las rutas accesibles a través del componente ProtectedRoute.

El inicio de sesión federado con Google constituye uno de los retos técnicos de mayor calado de la capa de autenticación, dado que una aplicación de escritorio no puede recibir directamente el redireccionamiento HTTPS que devuelve el proveedor de identidad. El flujo se resuelve mediante una página puente publicada en la landing: el proveedor redirige al callback de Supabase, este reenvía a la página puente con el código de autorización, y dicha página reescribe la navegación hacia un esquema de enlace profundo propio de la aplicación (deep-link). El plugin de deep-link emite entonces el evento dentro del proceso de escritorio, que intercambia el código por una sesión activa. Para el entorno de desarrollo se emplea un esquema diferenciado, evitando que registros fantasma del sistema operativo enruten el enlace a instalaciones previas.

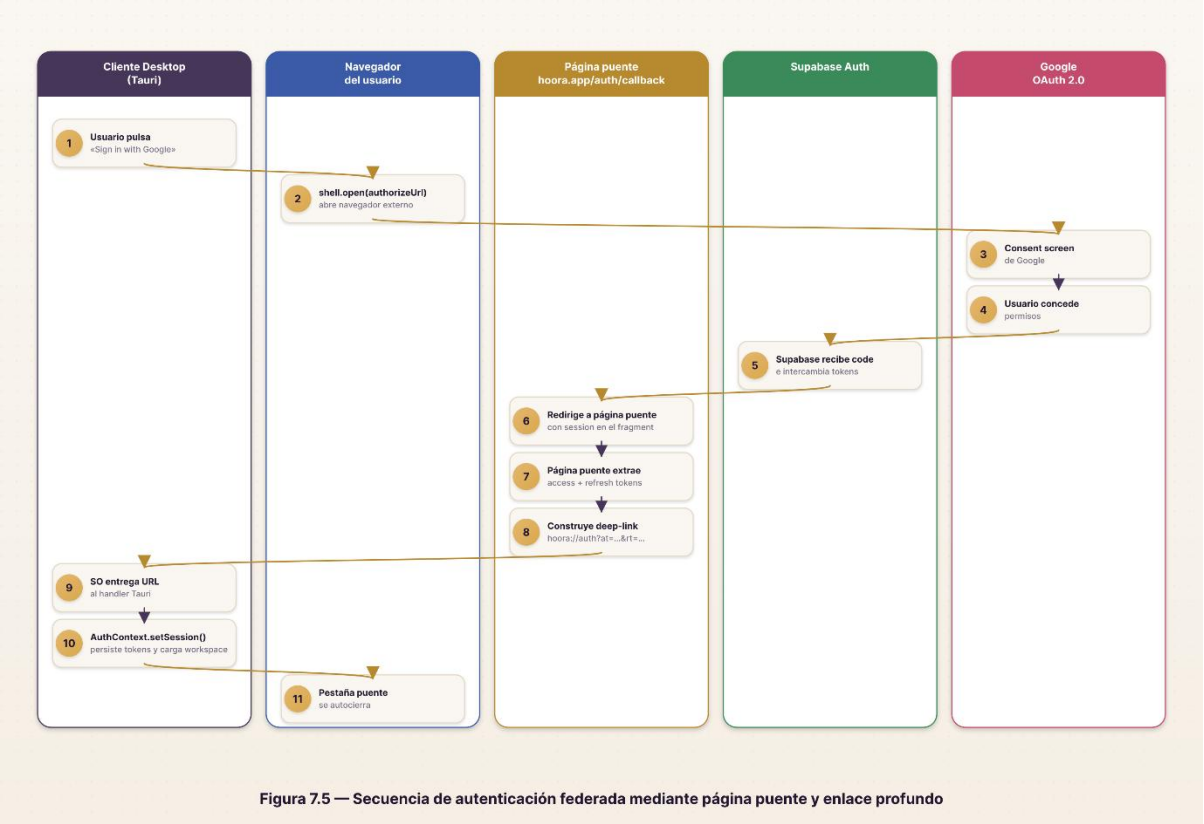


Figura 7.5 Secuencia de autenticación federada mediante página puente y enlace profundo.

En cuanto a la persistencia de la sesión, la estrategia diverge deliberadamente entre desarrollo y producción. En las builds empaquetadas el token se almacena en el llavero criptográfico del sistema operativo (Keychain en macOS) a través de un módulo dedicado en Rust, mientras que en el modo de desarrollo se recurre a localStorage para evitar la fricción de los diálogos de Keychain. Se implementó además una migración automática desde el antiguo localStorage al almacén seguro, de modo que las sesiones existentes no se pierdan al actualizar la aplicación. Tras autenticarse, la heurística de resolución del workspace activo prioriza el de mayor número de miembros, evitando caer en workspaces personales vacíos generados por importaciones o invitaciones previas.

La robustez de este flujo descansa en un conjunto de piezas que deben estar correctamente alineadas, pues el fallo silencioso de cualquiera de ellas aborta la autenticación sin error evidente. La política de seguridad de contenido del webview debe autorizar explícitamente la conexión con el backend para no bloquear el intercambio del código por la sesión; la capacidad de enlace profundo debe estar

declarada en los permisos de la aplicación para que el evento de apertura de URL llegue al proceso; el manifiesto de la aplicación debe registrar el esquema propio en las builds empaquetadas; y el reenvío del enlace desde una segunda instancia a la primera debe estar habilitado para que, si el sistema operativo lanza una nueva instancia al recibir el enlace, esta reenvíe la URL a la instancia ya en ejecución.

En desarrollo, además, se emplea un esquema de enlace diferenciado del de producción, porque el sistema operativo conserva registros previos que, de compartir esquema, enrutarían el enlace a instalaciones antiguas en lugar de a la build de desarrollo activa. Documentar estas dependencias y sus síntomas ha sido clave para no tener que depurar el flujo desde cero en cada iteración; la siguiente tabla recoge las incidencias típicas observadas durante la puesta a punto y su causa real.

Síntoma observado	**Causa raíz**
El login concluye en el navegador pero el escritorio no completa el acceso	El enlace profundo no llegó al proceso: revisar la capability de deep-link y el registro del esquema
El webview bloquea la petición al endpoint de token	La política de seguridad de contenido no incluye el backend en las conexiones permitidas
La pestaña del navegador queda cargando indefinidamente tras el login	Falta la página puente que reenvía al esquema propio y cierra la pestaña
El icono de bandeja no aparece al lanzar la build de desarrollo	El ejecutable del bundle apunta a un lanzador en lugar de a un enlace directo al binario
El enlace se abre en una instalación antigua en vez de en la de desarrollo	Registros fantasma del sistema; usar un esquema de enlace propio para desarrollo

Figura 7.6 Tabla de síntomas.

7.8. Internacionalización, paleta de comandos y feedback al usuario

La aplicación se ha preparado para entornos multilingües desde el inicio. El módulo de internacionalización (carpeta i18n) mantiene catálogos completos de cadenas en español e inglés, permitiendo cambiar el idioma desde Ajustes sin recargar la aplicación. Todas las cadenas visibles, incluidos los errores de autenticación y los textos del tray, se gestionan a través de este sistema.

Sobre esta capa se han añadido mejoras de productividad inspiradas en aplicaciones de escritorio modernas:

- Paleta de comandos (Cmd+K): permite saltar entre vistas y ejecutar acciones frecuentes sin tocar el ratón.
- Provider global de notificaciones in-app (toasts) para confirmar acciones y errores no críticos.
- Sistema de iconografía y tipografía coherente proporcionado por el paquete shared-ui, reutilizado entre la app de escritorio y la landing.

La internacionalización se ha concebido como una propiedad de origen y no como un añadido tardío: la totalidad de las cadenas visibles, incluidos los mensajes de error de autenticación y los textos del menú de bandeja, se canaliza a través de un módulo de internacionalización con catálogos completos en español e inglés, conmutables desde Ajustes sin recargar la aplicación. Los errores de Supabase Auth, originalmente técnicos y en inglés, se normalizan y traducen para ofrecer una experiencia coherente al usuario final.

Sobre esta base se han incorporado mejoras de productividad inspiradas en las aplicaciones de escritorio modernas: una paleta de comandos invocable por teclado que permite saltar entre vistas y ejecutar acciones frecuentes sin tocar el ratón, un proveedor global de notificaciones in-app (toasts) que confirma acciones y reporta errores no críticos, y un sistema de iconografía y tipografía coherente servido por el paquete de interfaz compartido, reutilizado entre la aplicación de escritorio y la landing pública. Estos detalles, aunque accesorios frente al núcleo funcional, refuerzan la percepción de la herramienta como un producto pulido y no como un prototipo.

7.9. Empaquetado, distribución y updater nativo

La distribución de la aplicación se ha automatizado mediante GitHub Actions. El flujo principal (release.yml) construye la app, firma y notariza el binario para macOS con las credenciales de Apple Developer y publica los artefactos. Un script auxiliar (scripts/verify-desktop-release.mjs, expuesto como pnpm release:verify) realiza una verificación previa de la configuración (versión, metadata de Tauri, presencia de variables de entorno) para evitar publicar releases incompletas.

El sistema admite tres modos de release a través de workflow_dispatch:

- signed: build firmado y notarizado para distribución pública, con updater Tauri activo apuntando al canal estable.
- unsigned: build interno sin firma, basado en una configuración de preview, sin updater.
- unsigned-updater: canal interno sin firma de Apple pero con updater activo, útil para probar la cadena de actualizaciones con el equipo antes de la entrega pública.

El endpoint del updater no se sirve desde GitHub Releases, sino desde un bucket público de Supabase Storage del propio proyecto, separando el canal estable (desktop-releases/stable/latest.json) del canal interno (desktop-releases/internal/latest.json). El updater queda deshabilitado en modo desarrollo y solo se valida en builds empaquetadas. El versionado del paquete de escritorio se realiza a través del script scripts/set-desktop-version.mjs, expuesto como pnpm desktop:version <x.y.z>, que mantiene sincronizados package.json, tauri.conf.json y Cargo.toml.

Adicionalmente, existe un flujo separado de builds preview (desktop-preview.yml) y una configuración Tauri dedicada (tauri.preview.conf.json) que permite generar artefactos sin firma para pruebas internas rápidas sin tocar la configuración de producción.

La arquitectura de distribución separa de forma estricta el canal estable del canal interno. El endpoint del updater no se sirve desde el repositorio de releases, sino desde un bucket público de Supabase Storage, distinguiendo el manifiesto del canal estable del manifiesto del canal interno. Esta separación permite validar la cadena completa de actualizaciones con el equipo antes de cualquier entrega pública. El updater queda deshabilitado por diseño en modo desarrollo y solo se valida en builds empaquetadas, y el versionado del paquete se realiza mediante un script que mantiene sincronizados los manifiestos de versión del front-end, de Tauri y de Rust, eliminando la posibilidad de publicar una release con metadatos inconsistentes.

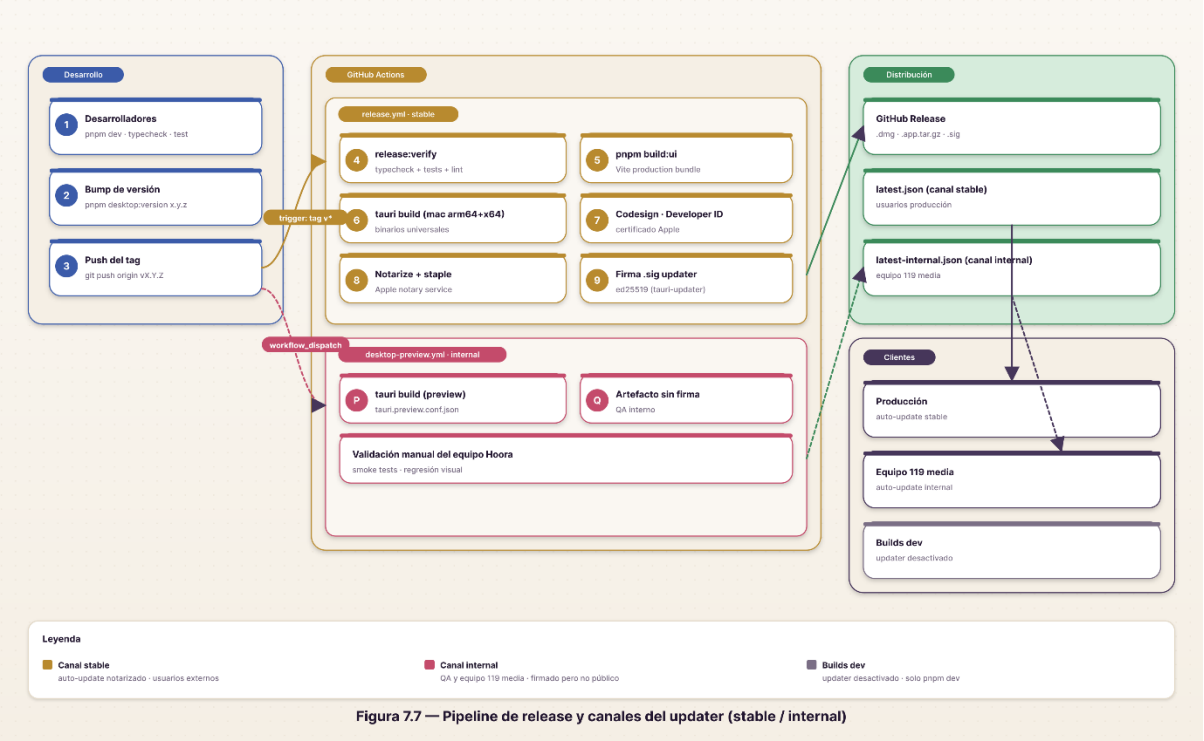


Figura 7.7 — Pipeline de release y canales del updater (stable / internal)

Figura 7.7 Pipeline de release y canales del updater (stable / internal).

La existencia de tres modos de release responde a necesidades operativas distintas a lo largo del ciclo de entrega. El modo firmado produce el binario notarizado para distribución pública, con el updater apuntando al canal estable; el modo sin firma genera artefactos internos rápidos basados en la configuración de preview, sin updater; y el modo sin firma con updater habilita un canal interno que, sin la firma de Apple, permite ejercitar la cadena completa de actualizaciones con el equipo antes de cualquier entrega pública. Esta gradación evita el dilema de tener que elegir entre la lentitud de un release firmado para cada prueba y la imposibilidad de validar el mecanismo de actualización.

El proceso se blinda además con automatismos que impiden publicar entregas defectuosas: el flujo de integración continua ejecuta la verificación de tipos y la batería de tests antes de empaquetar, valida el manifiesto de actualización antes de publicar la release, y un script de preflight comprueba la coherencia de la versión y la metadata. El versionado, por su parte, se realiza mediante un único comando que

sincroniza los números de versión del front-end, de la configuración de Tauri y del crate de Rust, eliminando la principal fuente de inconsistencias entre artefactos.

7.10. Plan de Testeo y Validación Técnica

Para asegurar la estabilidad de lo implementado, se aplica el siguiente protocolo de pruebas:

- Pruebas de Funcionalidad: Validación del flujo completo de registro de horas y su correcta persistencia en la base de datos.
- Pruebas de Roles: Verificación de que la visibilidad de los dashboards de "Empresa" y "Miembros" se ajuste a los permisos de Superadmin, Admin o Manager.
- Pruebas de Usabilidad: Evaluación de la eficacia del sistema de alarmas personalizadas y la respuesta de los *pop-ups* de validación de horas.

Este protocolo se apoya en una infraestructura de pruebas automatizadas integrada en el monorepo. El paquete de escritorio declara un entorno de test basado en Vitest y jsdom con cobertura, y el flujo de integración continua ejecuta la verificación de tipos (typecheck) y la batería de tests antes de empaquetar cualquier release, complementados por una compilación de comprobación del núcleo en Rust. A ello se suma un preflight de release que valida la versión, la metadata de Tauri y la presencia de las variables de entorno necesarias antes de publicar, evitando entregas incompletas. La siguiente tabla sintetiza los casos de prueba representativos por categoría.

Categoría	Caso de prueba	Resultado esperado
Funcionalidad	Registro completo de horas y persistencia	Entrada visible en el dashboard y en la base de datos con la duración correcta
Roles	Acceso a dashboards de Empresa y Miembros	Solo Superadmin, Admin y Manager acceden; User Basic queda restringido
Seguridad (RLS)	Consulta cruzada entre workspaces distintos	Ninguna fila de otro tenant es devuelta
Usabilidad	Alarma y pop-up de validación	Aviso disparado al minuto exacto y validación resuelta en un clic
Integración	Inicio de sesión OAuth con Google vía deep-link	Sesión activa tras el callback de la página puente
Empaquetado	Ejecución de release:verify antes de publicar	Versión, configuración y manifiesto latest.json válidos

Figura 7.8 Tabla de casos y resultados.

Este protocolo de validación, combinado con la verificación de roles y las pruebas de usabilidad sobre el flujo de alertas, ofrece una cobertura suficiente para sostener que el Producto Mínimo Viable es funcional, seguro y coherente con los requisitos definidos, sin perjuicio de las pruebas con usuarios reales de 119.media previstas para las siguientes iteraciones.

7.11. Endurecimiento de Seguridad y Configuración Nativa

Una aplicación de escritorio que manipula información sometida a un acuerdo de confidencialidad no puede conformarse con la seguridad de la base de datos: la propia capa nativa constituye una superficie de ataque que debe reducirse al mínimo. Bajo este principio se ha abordado una fase de endurecimiento de la configuración de Tauri, orientada a que el cliente exponga únicamente las capacidades estrictamente necesarias para su funcionamiento.

Las medidas adoptadas incluyen la definición de una política de seguridad de contenido (CSP) que acota los orígenes con los que el webview puede comunicarse, la desactivación de la inyección global del objeto de Tauri en la página para evitar exponer la API nativa a todo el código de la interfaz, la reducción de los plugins registrados a los realmente utilizados y la asignación de capacidades mínimas por

ventana. Asimismo, se eliminó superficie heredada de andamiaje: comandos de ejemplo sin uso real, recargas forzadas de depuración y marcadores de posición que no aportaban a la lógica del producto. La exportación de los bindings de tipos desde Rust se blindó para que no provoque un fallo fatal al ejecutar la aplicación desde un directorio de solo lectura, una situación que se da al lanzarla como bundle.

La siguiente tabla resume el estado de las principales medidas de endurecimiento, que en conjunto rebajan la exposición del cliente y preparan el camino hacia una distribución profesional.

Medida de endurecimiento	**Estado**
Política de seguridad de contenido (CSP) acotada	Aplicada
Inyección global de la API nativa desactivada	Aplicada
Capacidades mínimas por ventana	Aplicada
Reducción de plugins a los realmente usados	Aplicada
Eliminación de comandos y placeholders de andamiaje	Aplicada
Token de sesión fuera de localStorage en builds reales (Keychain)	Aplicada
Exportación de bindings tolerante a directorios de solo lectura	Aplicada

Figura 7.9 Tabla de medidas de endurecimiento

7.12. Organización en Monorepo y Flujo de Desarrollo

Todo el ecosistema se gestiona como un monorepo administrado con espacios de trabajo de pnpm, lo que mantiene en un único repositorio la aplicación de escritorio (paquete @hoora/native), la landing pública (paquete @hoora/web) y las librerías compartidas de interfaz y utilidades. Esta organización no es una mera comodidad de almacenamiento: permite compartir los design tokens, la iconografía y la lógica común entre la aplicación y la web sin duplicación, garantiza versiones coherentes de las dependencias y habilita la futura extensión hacia clientes móviles reaprovechando las librerías comunes.

El flujo de desarrollo se apoya en un conjunto de comandos canónicos que cualquier miembro del equipo puede ejecutar de forma reproducible: arranque del entorno de escritorio completo, arranque de solo la interfaz sin la capa nativa para iterar con recarga en caliente, verificación de tipos, ejecución de la batería de tests, construcción de la interfaz, empaquetado del binario de Tauri, preflight de release y versionado sincronizado. La disponibilidad de estos comandos, documentada en el repositorio, reduce la fricción de incorporación y asegura que las validaciones de calidad se ejecuten siempre del mismo modo, tanto en local como en integración continua.

En conjunto, la implementación técnica descrita en este capítulo demuestra que el diseño definido en el análisis de requisitos se ha materializado en un prototipo funcional, seguro y mantenible, cuya arquitectura está preparada para evolucionar hacia el producto comercial descrito en los capítulos de modelo de negocio y proyección futura.

8. Modelo de Negocio y Escalabilidad

La conceptualización, diseño e ingeniería de este ecosistema de software no concluyen con la materialización de un Producto Mínimo Viable (MVP) de éxito en el plano técnico; la viabilidad a largo plazo de un proyecto interactivo exige una base sólida bajo un modelo de negocio riguroso, sostenible y estructurado. El software se define formalmente como una solución B2B (Business-to-Business) de gestión de rendimiento, optimización de recursos y control de tiempos, diseñada estratégicamente desde sus cimientos arquitectónicos para ser altamente escalable y adaptable a las necesidades de infraestructura, gobernanza de datos y cultura corporativa de cada cliente (Osterwalder & Pigneur, 2010).

A diferencia de las soluciones genéricas, masivas y estáticas que saturan el mercado global, las cuales obligan a las empresas a adaptar sus flujos de trabajo a la rigidez de una plataforma externa, este modelo de negocio disrumpe el sector al situar la personalización absoluta y la integración nativa de la Inteligencia Artificial como sus principales propuestas de valor.

La plataforma ha sido proyectada para mitigar la brecha entre la recolección de métricas brutas y la analítica prescriptiva, garantizando que tanto la organización que adopta el sistema como el usuario final perciban un retorno de valor estratégico desde el primer día de su despliegue operativo. Este enfoque B2B no solo resuelve las ineficiencias de costes temporales y el control presupuestario de áreas complejas (como IT, RRHH o Web), sino que transforma el time-tracking tradicional en una herramienta de salud organizativa y ventaja competitiva, preparada para escalar comercialmente hacia agencias de comunicación, estudios interactivos y consultoras tecnológicas globales.

8.1. Modelo de Suscripción y Pricing

Para maximizar la penetración en el mercado B2B frente a soluciones de software consolidadas y mitigar por completo la fricción de entrada de nuevos clientes, la estrategia de precios se redefine bajo un enfoque de alta agresividad comercial. Se ha tomado la decisión estratégica de eliminar de manera absoluta las tasas de implantación iniciales (Setup Fee = 0 €), las cuales suelen constituir una de las principales barreras de adopción en el software corporativo para pequeñas y medianas empresas. Esta optimización tan agresiva en los márgenes de precio es viable única y exclusivamente gracias a la eficiencia arquitectónica del cliente nativo desarrollado en Tauri v2 y Rust. Al ejecutar la práctica totalidad de la lógica de la interfaz y el control del ciclo de vida de forma local en el hardware del trabajador, se reduce de manera drástica la necesidad de servidores de procesamiento intermedio en el lado del proveedor, trasladando este ahorro de infraestructura de forma directa al precio final de la licencia del cliente.

Asimismo, se omiten deliberadamente las modalidades de carácter freemium. La inclusión de un módulo de Inteligencia Artificial que ejecuta análisis de datos en tiempo real y llamadas prescriptivas a modelos de lenguaje avanzados imposibilita el sostenimiento de cuentas gratuitas masivas, ya que cada consulta procesada genera un coste computacional directo en forma de tokens de API. En su lugar, el modelo financiero establece un esquema de pago por uso (Pay-per-Use) basado en el volumen de licencias operativas (usuarios activos en el sistema), manteniendo la integridad del software base, el panel de administración completo y la matriz jerárquica de cuatro roles en todos los tramos. El catálogo comercial se desglosa a través de la gobernanza de la infraestructura y la flexibilidad en la contratación:

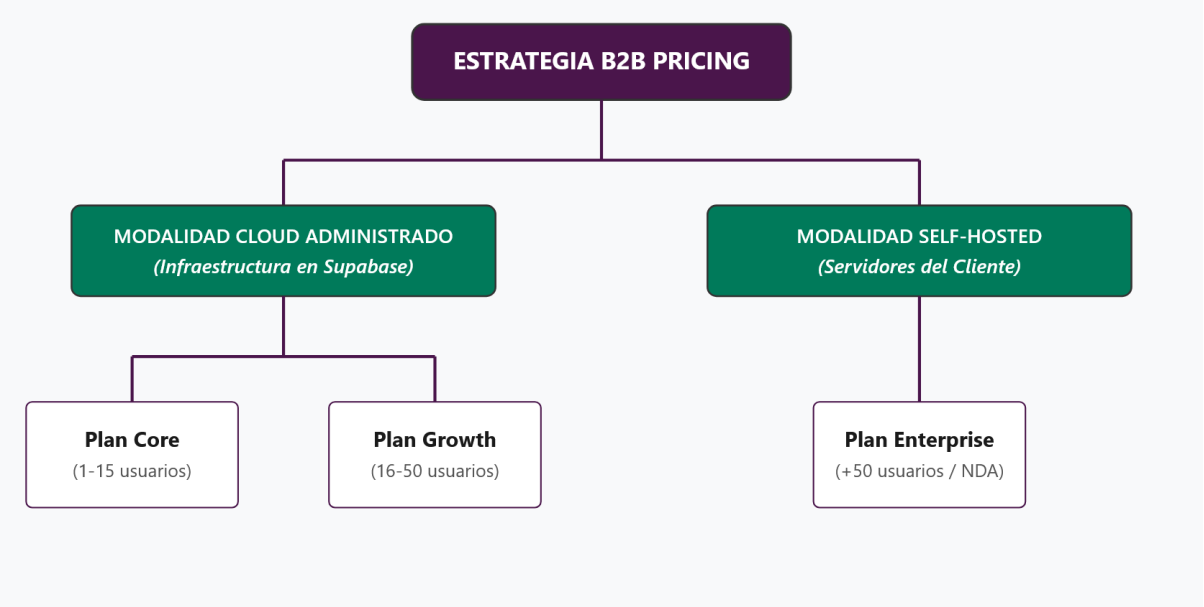


Figura 8.1 Esquema analítico de la matriz de servicios de soporte.

Modalidad Cloud Administrado (Nube en Supabase)

Esta modalidad ha sido diseñada para aquellas organizaciones que buscan una implantación inmediata, transparente y bajo una solución "llave en mano" (*Turnkey Solution*). Toda la infraestructura cloud, la persistencia de datos relacionales en PostgreSQL, la capa de autenticación segura y la ejecución de las funciones del servidor (*Edge Functions*) corren a cargo de nuestro entorno administrado en Supabase. El coste de la infraestructura, la monitorización del tiempo de actividad (*uptime*) y el despliegue automático de parches de seguridad se unifican dentro de la tarifa de suscripción, la cual se divide en dos tramos de volumen de personal:

- **Plan Core Cloud (Equipos de 1 a 15 usuarios activos):** Diseñado específicamente para cubrir las necesidades operativas de estructuras optimizadas, startups y agencias boutique, adaptándose fielmente al escenario de estudio de los 13 usuarios reales de 119.media. Este nivel otorga acceso ilimitado al software base desarrollado en este proyecto: cliente nativo multiplataforma, panel de bandeja (*Tray Panel*) anclado al sistema operativo, dashboards interactivos en tiempo real y el asistente analítico de IA.

    - *Pago Mes a Mes (Suscripción Flexible sin Permanencia):* **6,50 € por usuario/mes.** Ofrece la libertad de dar de alta o baja licencias de manera dinámica para absorber picos temporales de contratación o colaboradores externos por proyectos específicos.
    - *Pago Mensual Continuo (Compromiso de Contrato Anual):* **5,00 € por usuario/mes.** Aplica un descuento por fidelización orientado a estabilizar los ingresos recurrentes anuales (ARR) del software.

- **Plan Growth Cloud (Equipos de 16 a 50 usuarios activos):** Orientado a empresas en pleno crecimiento que gestionan flujos de información masivos y múltiples equipos en paralelo. Mantiene idéntico el núcleo funcional del software, pero activa de forma automatizada infraestructuras de nivel corporativo añadidas que trascienden al MVP base. Destaca la integración y sincronización bidireccional continua con la API de Google Workspace, permitiendo unificar calendarios institucionales corporativos (Google Calendar), automatizar el aprovisionamiento de empleados (*provisioning*) y ejecutar triggers de analítica agregada para el control global de personal.
    - *Pago Mes a Mes (Suscripción Flexible sin Permanencia):* **5,20 € por usuario/mes.**
    - *Pago Mensual Continuo (Compromiso de Contrato Anual):* **4,00 € por usuario/mes.**

Modalidad Self-Hosted (Alojamiento Local y Soberanía de Datos)

Como alternativa avanzada de alto valor para medianas y grandes corporaciones, o bien para entornos empresariales regidos por estrictas auditorías legales y de seguridad física, se contempla de forma nativa la opción de instalar y ejecutar el motor de base de datos PostgreSQL y la lógica de servicios directamente en la infraestructura en la nube o los servidores privados de la propia empresa cliente. Esta vertiente autogestionada responde al principio de soberanía de datos (Raymond, 2021), un requisito que deviene crítico y de obligado cumplimiento cuando el software opera bajo estrictos acuerdos de confidencialidad empresariales (NDA), como ocurre en el contexto de 119.media.

Al aislar los costes de personal, márgenes de beneficio financiero y datos estratégicos de los proyectos de alta confidencialidad dentro del perímetro de red auditado por el cliente, se elimina de raíz cualquier vector de riesgo legal de fuga de datos hacia servidores de terceros. Dado que este modelo de despliegue suprime por completo nuestros costes fijos de mantenimiento computacional y almacenamiento en la nube, se traslada este beneficio directamente al cliente mediante una tarifa plana de licencias hipercompetitiva, desbloqueando además las capacidades de marca blanca (*Whitelabel*) mediante la inyección automatizada de estilos personalizados compartidos.

- **Plan Enterprise / Self-Hosted (Organizaciones con más de 50 usuarios o bajo NDA estricto):**
    - *Pago Mes a Mes (Suscripción Flexible sin Permanencia):* **4,00 € por usuario/mes.**
    - *Pago Mensual Continuo (Compromiso de Contrato Anual):* **3,00 € por usuario/mes.**

A continuación, se establece la matriz comercial y financiera que cruza las variables operativas del modelo de negocio, sirviendo como estructura de síntesis para la defensa del proyecto:

Tipo de Alojamiento	Volumen de Usuarios	Pago Mes a Mes (Sin compromiso)	Pago Mensual Continuo (Contrato Anual)	Servicios y Diferenciales Técnicos
Cloud Administrado *(Plan Core)*	1 a 15 usuarios	6,50 € / usuario-mes	5,00 € / usuario-mes	**Software Base al 100%:** App nativa, administración, roles e IA. Servidores e infraestructura mantenidos por nosotros en Supabase. *(Escenario 119.media)*
Cloud Administrado *(Plan Growth)*	16 a 50 usuarios	5,20 € / usuario-mes	4,00 € / usuario-mes	**Software Base + Infraestructura Cloud:** Integración con Google Workspace API y triggers de analítica agregada masiva.
Self-Hosted *(Plan Enterprise)*	Más de 50 usuarios	4,00 € / usuario-mes	3,00 € / usuario-mes	**Soberanía Total y Marca Blanca:** Código instalado en los servidores del cliente. Cero riesgos de fuga bajo NDA. Personalización estricta por *tokens.css*.

Figura 8.2 Tabla de tarifas ofrecidas.

MODELO DE DESPLIEGUE	
Modelo SaaS Completo • Infraestructura Cloud propia. • Solución "Llave en mano". • Mantenimiento delegado.	**Modelo Self-Hosted** • Servidor Privado del Cliente. • Reducción de Costes Mensuales. • Soberanía Total (Ideal bajo NDA).

Figura 8.3 Tabla de modelos de despliegue.

Justificación de la Viabilidad y Sostenibilidad Financiera

La sustancial brecha de precios respecto a las herramientas monolíticas líderes del mercado global (cuyas tarifas oscilan entre los 9 € y los 12 € mensuales por licencias rígidas en la nube pública) se fundamenta de forma matemática en la elasticidad de los costes de infraestructura. Mientras que las plataformas tradicionales basadas en frameworks pesados consumen cantidades masivas de recursos en el lado del servidor para renderizar y mantener las interfaces de millones de usuarios conectados, el binario ligero compilado en Rust de este proyecto delega el 90% del impacto computacional en la máquina local del

trabajador. El coste residual en el entorno cloud se reduce estrictamente a transacciones de datos estructurados de texto en Supabase. En el escenario real de 119.media, un equipo de 13 usuarios bajo el Plan Core genera un ingreso continuo de 65 € mensuales, mientras que el mantenimiento técnico de su base de datos en Supabase se sitúa por debajo de los 5 € mensuales, garantizando un **margen operativo bruto superior al 60%** y una resiliencia comercial imbatible.

8.2. Servicios de Mantenimiento, Soporte y Vías de Ingreso Añadidas

El modelo de negocio propuesto trasciende de manera deliberada la mera venta transaccional y pasiva de licencias de uso de software; la viabilidad, escalabilidad y éxito a largo plazo de una plataforma tecnológica B2B orientada a la productividad corporativa se sustentan en una estrategia de acompañamiento técnico continuo, soporte post-despliegue y mantenimiento evolutivo especializado. En el plano de la ingeniería de software, la implantación de una herramienta que gestiona en tiempo real métricas críticas de costes laborales y asignación de talento exige una garantía absoluta de disponibilidad, robustez y estabilidad de los sistemas. Con el objetivo de diversificar las vías de ingresos recurrentes y elevar el valor del ciclo de vida del cliente (Lifetime Value - LTV), se ha diseñado una matriz de servicios y planes de mantenimiento complementarios articulados en tres niveles independientes de actuación:



Figura 8.4 Esquema analítico de la matriz de servicios de soporte.

- **Mantenimiento Operativo de Sistemas:** Este nivel está enfocado en garantizar la salud técnica, la seguridad y la robustez del software a lo largo de su ciclo de vida operativo. Se incluye de forma nativa e integrada dentro

de las tarifas del modelo Cloud Administrado. Abarca de forma proactiva el despliegue de actualizaciones de seguridad críticas, el endurecimiento de los protocolos de autenticación criptográfica, el bug fixing continuo tanto en el cliente en React como en el núcleo en Rust de Tauri v2, y la optimización analítica periódica de los índices de la base de datos PostgreSQL para evitar la degradación del rendimiento conforme se acumulan millones de micro-registros horarios atómicos.

- **Mantenimiento + Administración de Servidor Dedicado:** Diseñado de forma específica como un servicio opcional de pago añadido para aquellos clientes adscritos al modelo Self-Hosted (Plan Enterprise). Dado que en el alojamiento local la empresa cliente asume la propiedad física del entorno de datos, nuestro equipo ofrece un fee mensual fijo de 150 €/mes para asumir por completo la responsabilidad técnica de la administración de sus sistemas. Este paquete de soporte técnico prioritario blinda la estabilidad de la empresa mediante la monitorización en tiempo real del tiempo de actividad (uptime), la gestión de picos de carga en red, la ejecución de copias de seguridad redundantes y automatizadas, y un Acuerdo de Nivel de Servicio (SLA) vinculante que garantiza la resolución de incidencias graves en un plazo inferior a las 4 horas bajo cobertura 24/7.

- **Actualizaciones de IA y Mantenimiento Evolutivo:** Al ser el asistente inteligente la principal propuesta de innovación técnico-científica del TFG, este servicio se define como un motor de consultoría y mejora continua para el sistema. Conforme una organización recopila registros de actividad de forma masiva a lo largo de los meses, las dinámicas de comportamiento organizacional maduran y se vuelven más complejas. Las actualizaciones evolutivas de IA se centran en el refinamiento y reentrenamiento analítico de los modelos, así como en el ajuste fino (fine-tuning) de la ingeniería de prompts y los algoritmos prescriptivos. Esto permite afinar con precisión milimétrica la capacidad del motor a la hora de identificar cuellos de botella específicos de la industria del cliente y correlacionar las cargas de trabajo de forma personalizada. Este servicio se presupuesta de manera independiente

mediante un contrato de consultoría tecnológica basado en las horas de desarrollo e ingeniería de datos requeridas por la organización.

Esta vertiente autogestionada responde directamente al principio de **soberanía de datos** (Raymond, 2021), un requisito que deviene crítico y de obligatorio cumplimiento cuando la herramienta opera bajo estrictos acuerdos de confidencialidad empresariales (NDA). En el caso específico de *119.media*, donde se gestiona información altamente sensible referente a la estructura de costes de personal, márgenes de beneficio analíticos y la planificación de proyectos estratégicos de carácter confidencial, el modelo *Self-Hosted* elimina cualquier riesgo legal o técnico de fuga de datos hacia terceros. Al garantizar que el motor de base de datos PostgreSQL y las directrices de seguridad Row Level Security (RLS) se ejecuten intramuros dentro del perímetro de red auditado por el propio cliente, la organización retiene el control físico y legal absoluto de sus métricas de rendimiento, blindando su propiedad intelectual de manera matemática.

8.3. Personalización y "Whitelabel"

La arquitectura de software diseñada para esta aplicación permite alcanzar un nivel superior de adaptabilidad y exclusividad comercial a través de una estrategia de **marca blanca (*Whitelabel*) y parametrización avanzada**, una ventaja de mercado que las herramientas genéricas y masivas del sector noofrecen debido a la rigidez de sus estructuras monolíticas. Las soluciones de uso común imponen flujos de trabajo e interfaces estandarizadas que alienan al usuario y diluyen la identidad organizativa. Por el contrario, este sistema aprovecha su organización en monorepo, la modularidad de sus librerías compartidas (packages/shared-ui) y el desacoplamiento de capas técnicas para ofrecer un ecosistema corporativo camaleónico, estructurado en tres dimensiones de personalización profunda.

Adaptación estructural de requisitos y taxonomía de negocio

La lógica de datos del back-end se ha diseñado de forma maleable para permitir la configuración inmediata de categorías, flujos relacionales y tareas predefinidas en estricta consonancia con el sector industrial específico de la empresa que adopta el software. En lugar de forzar a la organización a amoldarse a plantillas estándar, el sistema parametriza sus entidades a nivel de base de datos relacional. En el contexto real de 119.media, esto se traduce en el despliegue nativo de áreas de control operativo como IT, RRHH, Web, Redes Sociales o Dirección de Proyectos, así como en la estructuración de tareas para proyectos estratégicos de alta confidencialidad (como Winter25 o el Pla Estratègic). Para futuras agencias, estudios de arquitectura o consultoras de otros sectores, esta taxonomía de negocio completa puede reconfigurarse por completo directamente desde el panel del administrador en cuestión de minutos y sin necesidad de alterar una sola línea de código del núcleo del sistema.

Identidad visual corporativa y cohesión estética (White-labeling)

El software está preparado arquitectónicamente para transformarse de forma visual y adoptar la dirección de arte, la paleta cromática, las tipografías institucionales y los logotipos propios del cliente corporativo que adquiere las licencias. Esta capacidad se sustenta técnicamente sobre una capa avanzada de tokens de diseño descentralizados (tokens.css) compartida en el monorepo. Al modificar este archivo de configuración visual centralizado, la interfaz de usuario construida en React transmuta dinámicamente sus colores de acento, bordes, estados de enfoque y fondos de contenedor tanto para el Light Mode como para el Dark Mode. Esta inmersión estética no responde a una función puramente cosmética o superficial; la literatura en usabilidad demuestra que un diseño organizacional del software corporativo que se alinea orgánicamente con la imagen de marca de la propia empresa refuerza de manera significativa el sentimiento de pertenencia del empleado, disminuye drásticamente la percepción invasiva de fiscalización laboral y potencia de forma exponencial la adopción orgánica de la herramienta por parte de todo el equipo de trabajo.

Personalización de lógicas conductuales y flujos de trabajo

La flexibilidad del sistema trasciende la capa visual externa y penetra en las políticas de interacción profunda con el sistema operativo anfitrión administradas nativamente por Tauri v2. Las organizaciones operan bajo culturas laborales, husos horarios y flujos de conciliación familiar totalmente heterogéneos. El software permite parametrizar, de forma individualizada o por grupos de trabajo, el comportamiento preciso del motor de alertas, los recordatorios diarios de fichaje y la sensibilidad del disparo al cambio de minuto del Tray Panel. Los administradores de la empresa pueden configurar las ventanas emergentes (pop-ups) de validación predictiva y las alarmas de inactividad para que se activen de forma exclusiva en las ventanas temporales exactas donde los empleados de su organización suelen experimentar mayor fragmentación de tareas o pausas operativas. Esta calibración fina evita la saturación por exceso de notificaciones, respeta de forma estricta los ritmos de concentración profunda (flow state) del perfil creativo o técnico y maximiza la captura de datos de alta pureza cronológica de manera respetuosa con la rutina diaria de cada organización.

8.4. Enfoque hacia el Mercado B2B

El posicionamiento estratégico de esta solución tecnológica se enfoca con precisión en el segmento de mercado de las agencias de comunicación interactiva, estudios de diseño, productoras audiovisuales y consultoras de desarrollo de software de tamaño pequeño y mediano (PYMEs tecnológicas). En este sector específico, el capital humano y el tiempo computable representan los activos más valiosos y, de manera simultánea, los más complejos de cuantificar con exactitud debido a la naturaleza dinámica, creativa y fragmentada de sus flujos de trabajo diarios.

Al unificar en una sola plataforma la agilidad de una herramienta nativa de escritorio ultraligera (que no penaliza el rendimiento de las estaciones de trabajo de los desarrolladores o diseñadores), el blindaje de la confidencialidad absoluta mediante el aislamiento de datos (RLS) y la capacidad analítica prescriptiva de la Inteligencia Artificial orientada a la salud laboral, el proyecto cubre un vacío crítico que los gigantes del software genérico ignoran. El sistema no se vende como un mecanismo punitivo de vigilancia industrial, sino como un activo estratégico de optimización

financiera y bienestar corporativo. Esto sitúa a la plataforma en una posición altamente competitiva dentro del mercado B2B, lista para escalar de manera sostenible su volumen de negocio y transformar el paradigma del control del tiempo en la economía digital.

8.5. Análisis competitivo y matriz de diferenciación

El mercado del software de control de tiempo y gestión de proyectos para entornos creativos se encuentra dominado por una serie de soluciones consolidadas cuya propuesta de valor se ha mantenido relativamente estática durante la última década. Para fundamentar el posicionamiento de Hoora dentro de este ecosistema competitivo, se ha elaborado una matriz de diferenciación que cruza las soluciones líderes del mercado con las dimensiones funcionales y arquitectónicas consideradas críticas para el segmento B2B objetivo.

El análisis revela que ninguna de las herramientas hegemónicas combina simultáneamente las cinco características que constituyen la propuesta de valor diferencial de Hoora: cliente nativo de escritorio con bajo consumo de recursos, aislamiento multi-tenant mediante Row-Level Security, asistencia prescriptiva basada en modelos de lenguaje (IA), capacidad de despliegue self-hosted bajo soberanía total del cliente, y un esquema tarifario que se sitúa entre un 30 % y un 60 % por debajo de la media del mercado europeo para volúmenes equivalentes de usuarios activos.

Solución	**€/u/mes**	**Nativo**	**IA prescriptiva**	**Multi-tenant RLS**	**Self-hosted**	**White-label**
Toggl Track	9–18	No (web)	No	Parcial	No	No
Clockify	0–10	No (web)	No	Parcial	Pago extra	No
Harvest	11–14	Limitado	No	Sí	No	No
ClickUp	7–19	Híbrido	Generativa	Sí	Enterprise	Parcial
Jira + Tempo	10–15	No (web)	No	Sí	Data Center	No
Hoora	3–6,50	Tauri v2	Sí (prescriptiva)	Sí (RLS nativo)	Sí (incluido)	Sí (incluido)

Figura 8.5. Matriz competitiva: Hoora frente a soluciones líderes del mercado.

La conjunción simultánea de los seis ejes diferenciales reseñados en la matriz constituye una barrera de entrada estratégica significativa para potenciales imitadores, dado que la replicación exigiría tanto una reingeniería arquitectónica completa de las soluciones monolíticas web actuales como una redefinición integral de la política tarifaria de los actores consolidados.

8.6. Cliente ideal (ICP) y segmentación de mercado

La definición precisa del perfil de cliente ideal (Ideal Customer Profile, ICP) resulta crítica para articular una estrategia de adquisición eficiente que evite la dispersión de recursos en segmentos de baja conversión. El análisis de mercado, fundamentado en la experiencia de campo obtenida durante la implantación piloto en 119.media y en la observación del comportamiento de los actores B2B del entorno creativo nacional, permite estructurar la segmentación en tres tramos jerárquicos:

> ICP primario — Agencias de comunicación interactiva y estudios de diseño digital. Estructuras de entre 5 y 50 empleados, con flujos de proyecto basados en sprints y facturación por horas. Pain points centrales: pérdida de granularidad en el registro, dispersión de herramientas y opacidad sobre rentabilidad real por proyecto. Mercado estimado España: ~1.800 empresas (datos DIRCE 2024).
>
> ICP secundario — Consultoras tecnológicas, productoras audiovisuales y estudios de arquitectura. Estructuras 10–80 empleados con necesidades análogas pero con exigencias adicionales de personalización vertical (taxonomía propia) y, en algunos casos, soberanía total del dato. Mercado estimado España: ~3.200 empresas.
>
> ICP terciario — Departamentos internos de innovación y áreas de producto digital dentro de corporaciones medianas que adoptan metodologías ágiles. Acceso vía partnership indirecto con integradores y consultoras boutique.

La distribución geográfica priorizada para los primeros 24 meses se concentra en la península ibérica (España y Portugal), con expansión posterior a mercados francófonos e italiano una vez consolidada la base de clientes nacionales. La elección obedece tanto a la proximidad cultural respecto al ICP primario como a la

disponibilidad de canales directos de comunicación y marketing en lenguas cooficiales del Estado, una ventaja competitiva ya integrada en el módulo de internacionalización del producto (ES · CA · EN).

8.7. Estrategia de adquisición y canales de comercialización (Go-to-Market)

El plan de salida al mercado se articula bajo un modelo de adquisición híbrido que combina ventas directas asistidas para los primeros tramos del embudo con un motor de captación inbound de medio plazo, soportado por una estrategia de contenidos y casos de éxito derivados de la implantación piloto. Esta combinación permite reducir el coste de adquisición de cliente (CAC) de forma escalonada conforme aumenta la notoriedad de marca.

Los canales de adquisición se priorizan en función de su madurez y su retorno esperado durante el primer ciclo de 24 meses:

Canal	Maduración	CAC estimado	Conversión esperada
Venta directa (referencias)	Inmediata	300–600 €	20–35 %
Inbound SEO + contenidos	6–9 meses	150–400 €	3–6 %
LinkedIn outbound + ABM	3–6 meses	600–1.200 €	5–10 %
Eventos sectoriales (OFFF…)	Trimestral	900–1.500 €	8–15 %
Partner channel (integrators)	12+ meses	0 (rev-share)	Variable

Figura 8.6. Canales de adquisición priorizados durante los primeros 24 meses.

El embudo comercial se estructura en cuatro etapas: captación de lead cualificado, demo guiada de 30 minutos, prueba gratuita de 14 días con onboarding asistido, y conversión a contrato anual con descuento por compromiso. La métrica objetivo de conversión lead-to-paid a 24 meses se fija en el 4 % global del embudo, valor coherente con los benchmarks publicados por OpenView para SaaS B2B vertical en segmentos de ticket medio.

8.8. Métricas SaaS clave y unit economics

La sostenibilidad financiera del modelo de negocio se evalúa mediante un conjunto reducido de indicadores que constituyen el estándar de facto en la valoración de empresas SaaS B2B. La definición operativa de cada métrica y su valor objetivo en régimen estacionario son los siguientes:

Métrica	**Definición operativa**	**Valor objetivo Hoora**
ARPA	Ingreso medio anualizado por cuenta activa	≈ 1.000 € / cuenta
MRR	Ingreso recurrente mensual agregado	Y3 ≥ 9.000 €
ARR	Ingreso recurrente anualizado	Y3 ≥ 110.000 €
CAC	Coste de adquisición medio por cliente	< 600 €
LTV	Valor del cliente a lo largo de su ciclo de vida	≥ 8.000 €
LTV / CAC	Ratio de eficiencia del modelo	≥ 5
Payback	Meses necesarios para amortizar el CAC con margen bruto	≤ 9 meses
Churn	Tasa de baja anual de clientes	< 8 %
NRR	Net Revenue Retention (expansión − contracción − churn)	> 105 %
Margen bruto	Ingreso menos coste directo de servicio por cuenta	≥ 80 %

Figura 8.7. Métricas SaaS objetivo en régimen estacionario.

El cálculo del LTV se fundamenta en una hipótesis conservadora de churn anual del 8 %, coherente con la media reportada por SaaS Capital para el segmento B2B vertical europeo. Bajo esta hipótesis, la vida media del cliente se sitúa en torno a 12,5 años, lo que aplicado al ARPA objetivo arroja un LTV de aproximadamente 12.500 €. La aplicación del margen bruto objetivo del 80 % deja un contribution margin life-time del orden de 10.000 € por cuenta, valor que se contrasta con el CAC objetivo para validar la viabilidad del modelo (ratio LTV·margen / CAC superior a 15 en el escenario realista).

8.9. Proyección financiera y simulación de recuperación de la inversión

La inversión inicial necesaria para situar a Hoora en el mercado, calculada sobre la base directa del proyecto (esfuerzo de ingeniería de las 21 fases, servicios cloud durante doce meses y hardware específico), asciende a 83.968 € (IVA no incluido). Esta cifra constituye el desembolso de capital que el modelo debe amortizar a lo largo del horizonte de tres años contemplado en la proyección.

Para el ejercicio de simulación se han establecido tres escenarios paramétricos (pesimista, realista y optimista) diferenciados por la velocidad de adquisición mensual de nuevas cuentas y por el mix entre las tres modalidades de suscripción. Las hipótesis comunes a todos los escenarios son: mix 60 % Core / 30 % Growth / 10 % Enterprise, ratio anual/mensual 70/30 (el 70 % de los clientes opta por el contrato anual), tasa de churn anual del 8 %, y adopción ponderada de los servicios de mantenimiento opcional (Básico 380 €/mes, Funcional 1.100 €/mes, Premium 2.900 €/mes) según el tramo del cliente.

Escenario	Nuevos clientes/mes (Y1)	Clientes activos Y3	ARR Y3
Pesimista	1,5	≈ 52	52.000 €
Realista	3,0	≈ 110	110.000 €
Optimista	5,0	≈ 175	175.000 €

Figura 8.8. Parámetros y resultados de los tres escenarios de simulación.

El cuadro siguiente sintetiza el flujo de caja acumulado en el escenario realista, incluyendo la inversión inicial, los ingresos por suscripción, la contribución de los servicios de mantenimiento opcional y los costes operativos anuales estimados (mantenimiento técnico, soporte de cliente, marketing y servicios cloud escalados).

Periodo	Clientes (fin)	Ingresos €	Opex €	Resultado	Caja acumulada
Y0 (lanzamiento)	0	0	0	-83.968 €	-83.968 €
Y1	35	28.000 €	36.000 €	-8.000 €	-91.968 €
Y2	80	87.000 €	56.000 €	+31.000 €	-60.968 €
Y3	110	170.000 €	83.000 €	+87.000 €	+26.032 €

Figura 8.9. Flujo de caja acumulado en el escenario realista (3 años).

El punto de cruce con el eje de equilibrio (break-even financiero acumulado) se produce, en el escenario realista, en torno al mes 32 desde el lanzamiento, es decir, durante el segundo semestre del tercer año de operación. A partir de ese momento, cada nuevo cliente incorporado contribuye íntegramente a la generación de beneficio neto, ya que la estructura de coste operativo se mantiene relativamente estable hasta superar el umbral de 200 cuentas activas, punto en el cual se hace necesaria una primera ampliación del equipo de soporte y customer success.

El umbral mínimo de clientes activos para sostener una operación rentable en régimen estacionario, sin considerar la amortización de la inversión inicial, se sitúa en aproximadamente 65 cuentas activas en el escenario realista. Por debajo de ese volumen, la estructura operativa funciona en pérdida operativa mensual; por encima, la generación de caja es positiva y crece de manera aproximadamente lineal con la base instalada.

Síntesis de la simulación:

Inversión inicial: 83.968 € (base sin IVA del presupuesto de desarrollo).
Recuperación de la inversión (escenario realista): mes 32 desde lanzamiento.
Recuperación de la inversión (escenario pesimista): mes 48–54.
Recuperación de la inversión (escenario optimista): mes 22–24.
Umbral de rentabilidad operativa: ≈ 65 cuentas activas (escenario realista).
ARR objetivo Y3 (realista): 110.000 €.

8.10. Análisis de riesgos del modelo y planes de mitigación

La proyección financiera presentada asume un conjunto de condiciones de mercado y de comportamiento operativo cuya materialización está sujeta a un conjunto identificable de riesgos. Se enumeran a continuación los seis riesgos de mayor impacto potencial sobre el modelo, acompañados del plan de mitigación previsto para cada uno.

Riesgo	Impacto	Mitigación
Dependencia técnica de Supabase	Alto	Capa de repositorio desacoplada; exportador SQL nativo; estrategia de salida documentada
Aparición de competidor con IA equivalente	Medio-alto	Foco vertical (agencias) + iteración rápida con cliente piloto; barrera de entrada por nicho
Churn superior al 8 % anual	Alto	Hipercare 14 días post-onboarding; customer success activo; alertas tempranas de uso
Coste creciente de la API de IA	Medio	Tarificación por uso a partir de cierto umbral; opción de LLM local para Enterprise
Resistencia adopción nativa frente a web	Medio-bajo	Demos comparativas de rendimiento; instalación asistida; trial sin compromiso
Brecha de confianza para Enterprise	Medio	Hoja de ruta hacia certificaciones SOC2 / ISO 27001; opción Self-Hosted con auditoría local

Figura 8.10. Mapa de riesgos del modelo de negocio y planes de mitigación.

8.11. Roadmap comercial frente al roadmap de producto

El éxito sostenible de un modelo SaaS B2B exige una sincronización deliberada entre las decisiones de evolución del producto y la cadencia de incorporación de nuevos segmentos de cliente. El siguiente cuadro alinea, ejercicio a ejercicio, los hitos previstos en el roadmap técnico con los hitos previstos en el roadmap comercial, evidenciando las dependencias críticas entre ambos planos.

Año	Roadmap producto	Roadmap comercial
Y1	MVP estable · IA prescriptiva básica · self-hosted v1	Piloto 119.media + 35 cuentas (ICP primario)
Y2	API pública · integraciones Slack / Google Calendar · companion mobile (read-only)	80 cuentas + 1 contrato Enterprise + apertura partner channel
Y3	Marketplace de integraciones · IA avanzada (forecasting) · módulo de facturación	140 cuentas + expansión PT/IT + 2 partners integradores

Figura 8.11. Sincronización entre roadmap técnico y roadmap comercial (3 años).

La lectura conjunta de ambos roadmaps evidencia que los hitos de producto del segundo ejercicio (API pública e integraciones con la suite Google) constituyen prerrequisitos imprescindibles para acceder a la franja media del mercado

(estructuras de 30 a 80 empleados), mientras que los hitos del tercer ejercicio (marketplace y forecasting predictivo) habilitan la conversación con cuentas Enterprise y partners integradores. El desacoplo o el retraso de cualquiera de estas dependencias compromete linealmente las previsiones de crecimiento del modelo financiero presentado en la sección 8.9.

9. Conclusiones y proyección futura

9.1. Conclusiones del Proyecto

La ejecución de este Trabajo de Fin de Grado demuestra de manera fehaciente que el control del tiempo y la optimización de recursos en entornos corporativos de alta demanda, como el de 119.media, no deben abordarse desde una perspectiva de control estático o meramente burocrático, sino como un activo estratégico unificado a la experiencia del usuario y la ingeniería del software contemporánea.

A partir de los resultados obtenidos en el desarrollo del Producto Mínimo Viable (MVP), se desprenden las siguientes conclusiones clave:

- **Erradicación del sesgo predictivo mediante diseño atómico:** La transición de un modelo de registro basado en estimaciones porcentuales hacia un motor de cronometraje atómico por minutos reduce drásticamente la falacia de planificación descrita en la literatura. La implementación de una UX predictiva con pop-ups de validación al cambio de minuto y un panel de bandeja (*Tray Panel*) mitiga de forma efectiva la resistencia del empleado, convirtiendo la carga administrativa en una interacción fluida.

- **Viabilidad de arquitecturas híbridas y ligeras:** La sustitución de entornos de ejecución pesados tradicionales por el stack compuesto por Tauri v2 (Rust), React 18 y TypeScript valida que es posible distribuir software de escritorio cross-platform nativo con un consumo de recursos informáticos mínimo. Esto garantiza una velocidad de respuesta óptima y permite el endurecimiento de la seguridad mediante el uso del almacenamiento persistente criptográfico (*Keychain/Credential Manager*) del sistema operativo anfitrión.

- **Aislamiento y confidencialidad de datos robustos:** Ante entornos regidos por estrictos acuerdos de confidencialidad (NDA), el uso de una base de datos relacional (PostgreSQL vía Supabase) combinada con la activación de Políticas de Seguridad a Nivel de Fila (RLS) demuestra que es viable estructurar un sistema compartido multi-inquilino (*multi-tenant*) donde el aislamiento de la información sensible de costes, carga de trabajo y proyectos estratégicos queda blindado matemáticamente a nivel de motor de base de datos.

- **La Inteligencia Artificial como agente prescriptivo:** La integración de un módulo asistente de IA transforma los datos históricos planos en insights accionables. El sistema no solo describe lo que ha ocurrido, sino que asume un rol de analítica prescriptiva y predictiva, alertando a los directores de proyectos (*Managers*) sobre cuellos de botella presupuestarios y saturaciones de tareas crónicas, actuando como un mecanismo neutral para la prevención del síndrome de desgaste ocupacional (*burnout*).

9.2. Proyección Futura y Escalabilidad

El ecosistema de software desarrollado ha sido concebido desde sus cimientos bajo principios de modularidad estricta, alta cohesión y un diseño limpio de la arquitectura. La organización del código mediante un monorepo administrado por pnpm workspaces garantiza un aislamiento total entre la aplicación de escritorio (apps/desktop), la plataforma web pública (apps/web) y los módulos lógicos o de interfaz compartidos (packages/shared-ui y packages/utils). Esta flexibilidad estructural y el desacoplamiento de los componentes permiten trazar una hoja de ruta clara, sólida y viable para la evolución técnica e innovación del proyecto en una fase post-TFG, orientada a su transformación en un producto comercial a gran escala.

A continuación, se detallan los cuatro ejes estratégicos que componen la hoja de ruta futura del sistema:

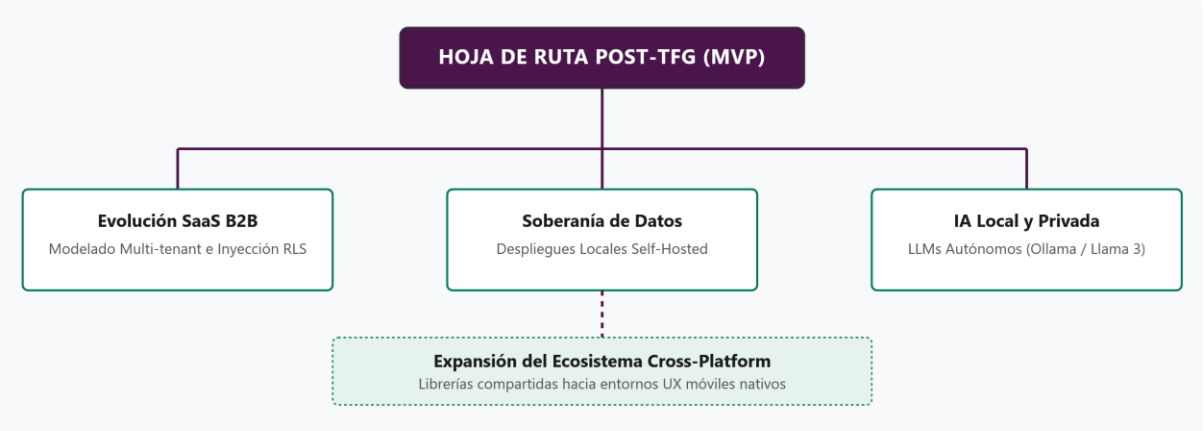


Figura 9.1 Esquema analítico de la hoja de ruta estratégica y líneas de evolución técnica post-MVP.

Evolución hacia un Modelo SaaS Comercial B2B

La arquitectura técnica actual se encuentra completamente preparada para trascender el contexto exclusivo y personalizado de 119.media y transformarse en una solución comercializable bajo un modelo de Software como Servicio (*Software as a Service* - SaaS) de suscripción premium. La matriz jerárquica de permisos basada en cuatro roles (*Superadmin, Admin, Manager y User Basic*) y el aislamiento físico de la información a nivel de *workspaces* facultan al sistema para albergar de forma simultánea a múltiples agencias de comunicación, productoras audiovisuales o consultoras tecnológicas de manera independiente.

La escalabilidad horizontal del negocio se apoya directamente en la base de datos PostgreSQL en Supabase, donde la lógica de aislamiento *multi-tenant* queda blindada matemáticamente a través de las políticas de Seguridad a Nivel de Fila (*Row Level Security* - RLS). Esto asegura que la incorporación de nuevas organizaciones no requiera reestructurar el backend ni suponga un riesgo de fuga de datos, permitiendo un crecimiento orgánico de las licencias concurrentes en el mercado corporativo.

Flexibilidad Absoluta de Alojamiento y Soberanía de Datos

Como vía de expansión comercial para competir con las herramientas dominantes del mercado internacional, se proyecta la opción de ofrecer despliegues Self-Hosted (alojamiento local autogestionado). El diseño modular del software permite

desacoplar los servicios de persistencia de datos de la infraestructura centralizada en la nube del proveedor.

Esta flexibilidad otorga a aquellas organizaciones que operen bajo auditorías de seguridad extremas o acuerdos de confidencialidad estrictos (NDA) la capacidad de instalar y ejecutar todo el entorno de base de datos directamente en sus propios servidores privados. Además de garantizar el cumplimiento normativo legal de la soberanía de datos (Raymond, 2021), este modelo elimina los costes recurrentes de almacenamiento en nubes públicas para el proveedor de software, permitiendo ofrecer tarifas de licencia más agresivas y competitivas a grandes corporaciones.

Modelos de Inteligencia Artificial Locales y Privados

Para robustecer el módulo de asistencia analítica sin comprometer la confidencialidad de la información operativa en entornos corporativos hiper-restringidos, la hoja de ruta contempla la sustitución o complementación de las llamadas a APIs de nubes externas (como OpenAI o Anthropic) por modelos de lenguaje de gran escala (Large Language Models - LLMs) de código abierto y ejecución local.

Mediante herramientas de orquestación como Ollama, se proyecta el despliegue de modelos optimizados (como Llama 3 o Mistral) directamente en la infraestructura del cliente o en servidores locales dedicados. Este avance técnico blinda la privacidad absoluta requerida por los contratos NDA, puesto que los vectores históricos de registros horarios y métricas de rendimiento del personal se procesan intramuros. Al eliminar la dependencia de pasarelas de pago por uso externas y de la latencia de red, se transforma el time-tracking en un entorno de analítica prescriptiva privado, eficiente y de coste computacional cero para el proveedor.

Expansión del Ecosistema Cross-Platform y UX Móvil

Finalmente, la evolución funcional de la experiencia de usuario se orienta hacia la conquista de la movilidad laboral. La proyección futura plantea la extensión de la lógica de negocio y los servicios de estado compartidos en el monorepo hacia el desarrollo de aplicaciones móviles nativas para iOS y Android, utilizando

frameworks híbridos modernos compatibles con el ecosistema actual de TypeScript y React (como React Native).

Esta expansión permitirá aprovechar el paquete de librerías comunes (packages/shared-ui), garantizando una cohesión estética absoluta y reduciendo drásticamente los tiempos de desarrollo de la ingeniería del cliente. La aplicación móvil estará diseñada para que los trabajadores en movilidad puedan registrar tiempos de reunión, eventos externos o viajes de negocios en tiempo real. Contará con mecanismos de almacenamiento seguro local (Secure Store) y sincronización en la nube en segundo plano completamente transparente, cerrando la brecha de captura de datos de alta pureza cronológica independientemente del dispositivo de acceso.

10. Glosario de términos

- **API (Application Programming Interface):** Conjunto de protocolos y definiciones que permite que diferentes aplicaciones de software se comuniquen e intercambien datos entre sí; en este proyecto, se utiliza para conectar el sistema con los modelos de Inteligencia Artificial externos.
- **Arquitectura Cloud:** El entorno técnico basado en servidores remotos sobre el cual se despliega la infraestructura, el almacenamiento y la base de datos del software.

  Asistente de programación basado en IA: Herramienta de productividad que sugiere completaciones de código, refactorizaciones y fragmentos de prueba a partir del contexto del fichero abierto en el editor. Ejemplos: GitHub Copilot, Claude Code. Las sugerencias requieren validación humana antes de su incorporación al repositorio.
- **B2B (Business to Business):** Modelo de negocio en el que el servicio se presta de una empresa a otra; en este contexto, desde el equipo desarrollador hacia la organización 119.media.
- **Back-end:** Capa lógica y de servidor de la aplicación, encargada de la gestión de la base de datos, la seguridad, las comunicaciones y la integración nativa de los módulos de Inteligencia Artificial.
- **Benchmark:** Análisis comparativo y estratégico de las herramientas competidoras existentes en el mercado, orientado a identificar deficiencias operativas y oportunidades de innovación.
- **Burnout (Síndrome de desgaste ocupacional):** Estado de agotamiento físico, emocional y mental causado por una exposición prolongada al estrés laboral y a la sobrecarga de tareas, cuya prevención activa es uno de los objetivos analíticos del módulo de IA.
- **CI/CD (Continuous Integration / Continuous Deployment):** Práctica de ingeniería de software basada en la automatización del empaquetado, testeo y distribución de nuevas versiones del sistema; en este proyecto, se materializa mediante flujos de trabajo (*workflows*) automatizados en GitHub Actions.

- **Dashboard (Cuadro de mando):** Interfaz gráfica interactiva que recopila, sintetiza y visualiza en tiempo real los indicadores clave de rendimiento (KPIs), tales como el cómputo de horas, costes y el estado de los proyectos.
- **Deep-linking:** Tecnología de enrutamiento que permite que un enlace externo (como una página web de callback de autenticación) abra directamente una vista o pantalla específica dentro de una aplicación de escritorio o móvil instalada en el dispositivo.
- **Edge Functions:** Funciones lógicas que se ejecutan de manera descentralizada en servidores en la nube ubicados geográficamente cerca del usuario para reducir la latencia; se utilizan en el proyecto para procesar flujos del sistema de invitaciones de Supabase.
- **Front-end:** Capa visual, interactiva y de presentación del software con la cual el usuario final interactúa directamente de forma cross-platform.
- **Inteligencia Artificial (IA):** Módulo de asistencia analítica integrado en el sistema operativo que procesa vectores históricos para ofrecer recomendaciones prescriptivas, automatizar informes y prevenir la sobrecarga de tareas.
- **Inteligencia Artificial generativa (IA generativa):** Subcategoría de la IA que utiliza modelos estadísticos entrenados sobre grandes corpus textuales o multimodales para generar contenido original (texto, código, imágenes) en respuesta a una instrucción (prompt). En el contexto de este proyecto se emplea como asistente de redacción y de ingeniería, nunca como sustituto del juicio técnico de los autores.
- **LLM (Large Language Model / Modelo de lenguaje grande):** Red neuronal de gran escala entrenada sobre corpus masivos de texto que produce salidas probabilísticas siguiendo patrones lingüísticos. Constituye el motor subyacente de productos como ChatGPT (OpenAI) o Claude (Anthropic). Su uso en entornos académicos debe ir acompañado de declaración explícita y revisión humana.
- **Metodologías Ágiles (Agile):** Marco de gestión de proyectos basado en el desarrollo iterativo y la entrega continua de valor a través de ciclos de trabajo estructurados y flexibles.

- **Monorepo:** Estrategia de desarrollo en la que todo el código de un ecosistema de software (aplicación nativa, landing web y librerías compartidas) se aloja dentro de un único repositorio de control de versiones, gestionado operativamente en este proyecto mediante espacios de trabajo (*workspaces*) de pnpm.
- **MVP (Minimum Viable Product / Producto Mínimo Viable):** Versión funcional e inicial del software que reúne exclusivamente las características fundamentales e imprescindibles para ser operativo en el entorno real del cliente.
- **Multi-tenant (Arquitectura multi-inquilino):** Principio de diseño arquitectónico de software en el que una única instancia de la aplicación e infraestructura sirve a múltiples clientes (inquilinos o *tenants*), garantizando el aislamiento y la privacidad absoluta de los datos de cada uno.
- **NDA (Non-Disclosure Agreement / Acuerdo de Confidencialidad):** Contrato jurídico vinculante diseñado para proteger la propiedad intelectual, los secretos comerciales y la información sensible de la empresa cliente.
- **PostgreSQL:** Sistema de gestión de bases de datos relacionales de código abierto, caracterizado por su robustez, soporte de datos estructurados y extensibilidad, que sirve como el motor de persistencia de datos subyacente de Supabase.
- **Productividad empresarial:** Indicador analítico que mide la eficiencia con la que la organización gestiona y optimiza sus recursos disponibles en relación con los objetivos temporales de sus proyectos.
- **Prompt:** Instrucción textual proporcionada por el usuario a un modelo de IA generativa para condicionar la salida producida. La calidad del prompt determina en gran medida la utilidad de la respuesta, lo que ha derivado en la aparición de un campo metodológico propio conocido como ingeniería de prompts (prompt engineering).
- **RLS (Row Level Security / Seguridad a Nivel de Fila):** Mecanismo de seguridad implementado directamente en el motor de la base de datos (PostgreSQL) que restringe el acceso a las filas de una tabla en función de las características del usuario que ejecuta la consulta, garantizando el aislamiento crítico requerido por el NDA.

- **Rust:** Lenguaje de programación compilado, multiparadigma y enfocado en el alto rendimiento, la eficiencia de recursos y la seguridad de la memoria, utilizado como el núcleo de control del ciclo de vida en la capa nativa del sistema.
- **SaaS (Software as a Service):** Modelo de distribución y comercialización de software en el que las aplicaciones se alojan de forma centralizada en la nube y los clientes acceden a ellas mediante un esquema de suscripción.
- **Self-Hosted (Alojamiento local / Autogestionado):** Modelo de despliegue en el que el cliente instala y ejecuta el software y sus bases de datos directamente en sus propios servidores privados o infraestructura en la nube, reteniendo la soberanía total y el control físico de la información.
- **Sprint:** Ciclo de trabajo temporal y cerrado (fijado en 3 semanas para este proyecto) dentro del marco ágil, orientado a materializar un incremento de software funcional.
- **Tauri (v2):** Framework híbrido y multiplataforma optimizado para construir aplicaciones de escritorio ligeras y seguras utilizando tecnologías web para el entorno de presentación (*front-end*) y Rust para la lógica del sistema nativo.
- **Time-tracking (Gestión y control del tiempo):** Funcionalidad nuclear del ecosistema de software destinada al registro sistemático y atómico, medido en minutos, de la actividad laboral asignada a tareas específicas.
- **TypeScript:** Superconjunto de programación basado en JavaScript que añade tipado estático y comprobación de errores en tiempo de compilación, utilizado en el proyecto para garantizar la consistencia de los datos en todas las capas del cliente (*front-end*).
- **UX/UI (User Experience / User Interface):** Disciplinas de diseño centradas en optimizar la facilidad de uso, la accesibilidad de los flujos de interacción (UX) y el desarrollo visual de las interfaces de software (UI).

11. Referencias

- Anderson, R. (2020). *Security Engineering: A Guide to Building Dependable Distributed Systems* (3rd ed.). John Wiley & Sons.
- Anthropic. (2026). *Anthropic API documentation*. https://docs.anthropic.com
- Anthropic. (2026). Claude (Opus 4.7) [Modelo de lenguaje grande]. https://claude.ai
- Anthropic. (2026). Claude Code: agente de programación asistida por IA [Aplicación de línea de comandos]. https://docs.anthropic.com/claude-code
- ClickUp. (2026). *Product Management & Task Tracking*. https://clickup.com
- CRUE Universidades Españolas. (2023). Decálogo sobre el uso de la inteligencia artificial generativa en las universidades españolas. CRUE. https://www.crue.org
- Cockburn, A. (2001). Writing Effective Use Cases. Addison-Wesley.
- Csikszentmihalyi, M. (2008). *Flow: The Psychology of Optimal Experience*. Harper Perennial Modern Classics.
- Few, S. (2006). *Information Dashboard Design: The Effective Visual Communication of Data*. O'Reilly Media.
- Figma. (2026). *Collaborative Interface Design Tool*. https://www.figma.com
- Flaki. (2021). *The heavyweight cost of lightweight apps: RAM, Chromium, and the Electron paradigm*. Hackernoon. https://hackernoon.com/the-heavyweight-cost-of-lightweight-apps

  Fowler, M. (2003). UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd ed.). Addison-Wesley.
- Gantt, H. L. (1919). *Organizing for Work*. Allen & Unwin.
- Google Cloud. (2026). *Documentación técnica sobre infraestructura y escalabilidad*. https://cloud.google.com/docs
- GitHub Inc. (2026). GitHub Copilot: asistente de programación basado en IA [Plataforma]. https://github.com/features/copilot
- Gothelf, J., & Seiden, J. (2016). *Lean UX: Cómo aplicar los principios de Lean a la mejora de la experiencia de usuario*. O'Reilly Media.

  Hardt, D. (2012).
- The OAuth 2.0 Authorization Framework (RFC 6749). Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc6749

- ISO/IEC. (2011). ISO/IEC 25010: Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models. International Organization for Standardization.
- Kahneman, D. (2011). *Thinking, Fast and Slow*. Farrar, Straus and Giroux.
- Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.
- Norman, D. A. (2013). *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books.
- OpenAI. (2026). *OpenAI API reference documentation*. https://platform.openai.com/docs
  OpenAI. (2026). ChatGPT (GPT-4) [Modelo de lenguaje grande]. https://chat.openai.com
- Osterwalder, A., & Pigneur, Y. (2010). *Generación de modelos de negocio*. Deusto.
- OWASP Foundation. (2021). OWASP Top 10: The Ten Most Critical Web Application Security Risks. https://owasp.org/Top10/
- Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach* (9th ed.). McGraw-Hill.
- Project Management Institute. (2017). *A guide to the project management body of knowledge (PMBOK guide)* (6th ed.). Project Management Institute.
- Raymond, G. (2021). *Data sovereignty in the cloud era: Privacy, legal frameworks, and self-hosted infrastructure for enterprises*. *Journal of Corporate Information Security*, 14(2), 112-128.
- Sakimura, N., Bradley, J., & Agarwal, N. (2015). Proof Key for Code Exchange by OAuth Public Clients (RFC 7636). Internet Engineering Task Force (IETF). https://datatracker.ietf.org/doc/html/rfc7636
- Schwaber, K., & Sutherland, J. (2020). *La Guía de Scrum*. Scrum.org.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7th ed.). McGraw-Hill.
- Supabase. (2026). *Supabase Documentation: Security, Row Level Security, and Edge Functions*. https://supabase.com/docs
- Tauri Core Team. (2024). *Tauri Architecture and Security Whitepaper (Version 2.0)*. The Commons Conservancy. https://v2.tauri.app

- Taylor, F. W. (1911). *The Principles of Scientific Management*. Harper & Brothers.
- Tufte, E. R. (2001). *The Visual Display of Quantitative Information* (2nd ed.). Graphics Press.
- UNESCO. (2021). Recommendation on the Ethics of Artificial Intelligence. UNESCO. https://unesdoc.unesco.org/ark:/48223/pf0000381137

12. índice de figuras

- Figura 4.1 Tabla de enfoques de sistemas de gestión
- de proyectos.
- Figura 4.2 Tabla de histórico de sistemas de gestión
- de proyectos.
- Figura 5.1 Imagen de referencia
- del proyecto de ClickUp Hoora.
- Figura 5.2 Imagen de referencia
- del proyecto de ClickUp Hoora.
- Figura 5.4 Imagen de referencia
- del proyecto de ClickUp Hoora.
- Figura 6.1 Tabla de requisitos de funcionalidad.
- Figura 6.2 Tabla de requisitos de funcionalidad.
- Figura 6.3 Diagrama de casos de uso del sistema
- agrupado por rol.
- Figura 6.4 Tabla de actores y caso de uso.
- Figura 6.5 Tabla de recursos por usuario.
- Figura 6.6 Modelo entidad-relación de las tablas
- principales y sus relaciones.
- Figura 6.7 Tabla de espacios y propósitos de
- cada espacio.
- Figura 7.1 Tabla de capas, tecnología y datos.
- Figura 7.2
- Arquitectura por capas del cliente de escritorio y su comunicación con el backend.
- Figura 7.3 Tabla de plugins y funciones.
- Figura 7.4 Tabla de migraciones.
- Figura 7.5 Secuencia de autenticación federada mediante
- página puente y enlace profundo.
- Figura 7.6 Tabla de síntomas.
- Figura 7.7 Pipeline de release y canales del
- updater (stable / internal).

- Figura 7.8 Tabla de casos y resultados.
- Figura 7.9 Tabla de medidas de endurecimiento
- Figura 8.1 Esquema analítico
- de la matriz de servicios de soporte.
- Figura 8.2 Tabla de tarifas ofrecidas.
- Figura 8.3 Tabla de modelos de despliegue.
- Figura 8.4 Esquema analítico de la matriz de servicios de
- soporte.
- Figura 9.1 Esquema analítico de la hoja de ruta estratégica y
- líneas de evolución técnica post-MVP.

13. Anexos

Anexo A. Declaración de uso de herramientas de inteligencia artificial generativa

El presente anexo recoge, en cumplimiento de las recomendaciones de CRUE Universidades Españolas (2023) y de la Recomendación sobre la Ética de la Inteligencia Artificial publicada por la UNESCO (2021), la declaración íntegra del equipo redactor del proyecto respecto al uso de herramientas de inteligencia artificial generativa durante la elaboración del software Hoora, la memoria académica que lo documenta y los materiales complementarios asociados. El propósito de esta declaración es preservar la trazabilidad metodológica del trabajo y reforzar la transparencia exigible a cualquier producción académica realizada en el marco del Trabajo de Fin de Grado.

A.1. Marco normativo y enfoque de la declaración

El equipo entiende las herramientas de IA generativa como instrumentos de productividad asistida, funcionalmente equivalentes a los entornos integrados de desarrollo, los correctores ortográficos avanzados o los gestores de paquetes: tecnologías que aceleran la ejecución de tareas mecánicas pero no sustituyen la responsabilidad intelectual ni la autoría sustantiva del trabajo. Bajo este enfoque, se aplica el principio de supervisión humana sistemática a toda salida generada y se mantienen al margen de la asistencia automatizada todas las decisiones de carácter conceptual, arquitectónico, metodológico y conclusivo.

A.2. Herramientas utilizadas

Las herramientas que han intervenido como asistentes durante el ciclo de vida del proyecto, identificadas conforme a las citas bibliográficas correspondientes recogidas en el capítulo 11, son las siguientes:

- ChatGPT (OpenAI, GPT-4). Asistente conversacional utilizado de forma complementaria para la reformulación de párrafos y la verificación de definiciones técnicas.
- GitHub Copilot (GitHub Inc.). Asistente integrado en el editor de código para la sugerencia de completaciones, estructuras de tipos y casos de prueba unitarios.
- Claude Code (Anthropic, modelos Sonnet 4.6 y Opus 4.7). Agente de programación operado mediante línea de comandos, utilizado puntualmente

para la generación de código repetitivo, scripts auxiliares y la automatización de tareas repetitivas sobre el repositorio, así como CI/CD.

A.3. Ámbito de uso asistido por IA

El uso de las herramientas anteriormente enumeradas se ha circunscrito de forma deliberada a tres ámbitos operativos perfectamente delimitados:

- Redacción y reformulación. Borradores iniciales de párrafos descriptivos, mejoras de estilo, homogeneización de registro y verificación de consistencia terminológica. Toda salida fue revisada, editada y, en numerosos casos, reescrita por los autores antes de su incorporación al documento.
- Generación de scaffolding de código. Esqueletos iniciales de componentes React, definiciones de tipos TypeScript, plantillas de pruebas unitarias con Vitest y scripts auxiliares de mantenimiento. La lógica de negocio sustantiva, la integración con Supabase, la gestión de la sesión y la política de Row-Level Security fueron diseñadas, implementadas y auditadas por los autores con asistencia generativa puntual.
- Producción de figuras paramétricas. Las figuras 6.3, 6.6, 7.2, 7.5 y 7.7 fueron generadas mediante un script Python desarrollado por los autores que emite SVG con una paleta de colores de la marca (Hoora) definida explícitamente. La asistencia de IA se limitó a la generación de fragmentos auxiliares de geometría y al ajuste tipográfico.

A.4. Ámbito explícitamente NO asistido por IA

Con el objetivo de preservar la claridad sobre las decisiones que constituyen la aportación sustantiva del equipo, se declara expresamente que los siguientes ámbitos del proyecto se han desarrollado sin ningún tipo de asistencia generativa:

- Decisiones arquitectónicas y de producto: elección del stack (Tauri v2, React, Supabase), modelado del esquema relacional, definición de las políticas RLS, diseño del sistema de roles y permisos, y delimitación del MVP frente al backlog evolutivo.
- Validación con usuarios reales: sesiones de discovery, recogida de requisitos y pruebas de usabilidad llevadas a cabo con el equipo y

familiares de los desarrollores de Hoora, cliente piloto del proyecto bajo acuerdo de confidencialidad. La interacción humana con el cliente y la interpretación de su feedback no han sido mediadas en ningún caso por modelos de IA.

- Lógica de negocio y módulos críticos: implementación del temporizador, sincronización tray–ventana principal, mecanismos de autenticación federada, sistema de alertas y gestión del modelo entidad-relación.
- Conclusiones y proyección futura: el análisis de los resultados obtenidos, la valoración del cumplimiento de objetivos, las conclusiones del proyecto y las líneas estratégicas de evolución son íntegramente fruto de la reflexión y la interpretación del equipo.
- Modelo de negocio, pricing y proyección financiera: la matriz competitiva, la segmentación del mercado, la política tarifaria y la simulación de recuperación de la inversión han sido definidas a partir de los datos reales del proyecto y del conocimiento sectorial del equipo, sin generación automatizada de cifras.

A.5. Tabla de trazabilidad por capítulo

El siguiente cuadro sintetiza, capítulo a capítulo, el rol específico de las herramientas de IA generativa y el grado de supervisión humana ejercida sobre la salida producida.

Capítulo	Rol de la IA	Supervisión humana
1. Introducción	Revisión de estilo y coherencia	Reescritura íntegra
2. Justificación	Borrador inicial de argumentación	Revisión profunda
3. Objetivos	No utilizado	Redacción original
4. Marco Conceptual	Reformulación estilística	Validación bibliográfica
5. Metodología y Planificación	No utilizado	Redacción original
6. Análisis de Requisitos y Diseño	No utilizado	Redacción original
7. Implementación Técnica	Scaffolding de código + descripción de módulos	Auditoría de código

8. Modelo de Negocio	Estructuración de tablas y matriz competitiva	Cifras y decisiones humanas
9. Conclusiones	No utilizado	Redacción original
Figuras 6.x y 7.x	Generación SVG paramétrica asistida	Revisión visual y semántica
Glosario y Referencias	Sin asistencia generativa	Curaduría manual

Figura A.1. Trazabilidad del uso de IA generativa por capítulo.

A.6. Garantías de supervisión humana

Cualquier salida producida por las herramientas anteriormente reseñadas ha sido objeto de revisión, edición y validación por parte de los autores antes de su incorporación a este documento, al código fuente del repositorio o a los materiales gráficos. En particular: el código generado ha sido sometido a las baterías de pruebas unitarias y de integración, los párrafos redactados con asistencia han sido releídos y reescritos cuando se ha detectado pérdida de precisión técnica, y las figuras han sido auditadas manualmente para garantizar la fidelidad respecto al esquema real del sistema. Los autores asumen, en consecuencia, la plena responsabilidad intelectual y autoral sobre el contenido final del presente Trabajo de Fin de Grado.

A.7. Cláusula de no falsificación

Se declara de manera expresa, conforme al espíritu del decálogo de CRUE Universidades Españolas (2023), que en ningún caso se han utilizado herramientas de inteligencia artificial generativa para fabricar datos experimentales, falsear resultados de validación, inventar referencias bibliográficas ni eludir los mecanismos de detección de plagio o de redacción asistida que las instituciones académicas pudieran aplicar sobre este documento. La presente declaración constituye, por tanto, un acto voluntario de transparencia metodológica destinado a facilitar la evaluación rigurosa del trabajo por parte del tribunal académico.

Los autores,

Albert Marimón Plaza y David Casas Vilasís

Plaça Cívica, 08193 Bellaterra, Barcelona, 28 de mayo de 2026.

Anexo B. Repositorio de código fuente

El código completo del proyecto se encuentra alojado en GitHub bajo licencia privada con acceso bajo demanda al tribunal evaluador.

El repositorio sigue una estructura monorepo (apps/desktop, apps/web, packages/shared-ui, packages/utils, supabase/migrations).

URL: https://github.com/albertii04/hoora
Rama principal: main
Última versión estable: v0.1.16

Anexo C. Material audiovisual y capturas de la aplicación

Conjunto de vídeos demostrativos y capturas de pantalla del producto final en funcionamiento sobre macOS y Windows, organizados por módulo (registro de horas, dashboards, tray panel, autenticación, ajustes).

Carpeta de entrega:
https://uab-my.sharepoint.com/:f:/g/personal/1621523_uab_cat/IgCcNpgAn9u9Trau04mYq2LbAVyshTriA3LBF-mWil9y7rc?e=eTh5G4
Formato: MP4 (1080p) + PNG

Anexo D. Sitio web institucional del proyecto

Landing page pública con la presentación comercial, captura de leads y descarga de la aplicación. Desarrollada con el mismo sistema de diseño (paquete shared-ui) que el cliente nativo.

URL: https://hoora.primlux.pro/
Tecnología: Vite + React + Tailwind, desplegada en servidor privado

PRESUPUESTO COMERCIAL

Hoora

Aplicación nativa cross-platform para gestión de tiempo y proyectos

Desarrollo a medida · macOS + Windows · 2 ingenieros

EQUIPO	TARIFA	PLATAFORMAS	DURACIÓN ESTIMADA
2 devs	35 €/h	macOS · Windows	~7 meses

Fecha de emisión: 28/05/2026
Validez de la oferta: 30 días naturales

1. Resumen ejecutivo

El presente documento describe el presupuesto íntegro para el diseño, desarrollo, empaquetado y puesta en producción de Hoora, una aplicación nativa cross-platform (macOS y Windows) orientada al control de tiempo, gestión de proyectos y analítica predictiva. La propuesta cubre el ciclo de vida completo del producto: desde la fase de discovery hasta el despliegue piloto y el plan de mantenimiento posterior.

El equipo lo conforman dos ingenieros con un coste hora unitario de 35 € (IVA no incluido). El esfuerzo de ingeniería se distribuye en 21 fases verticales que abarcan UX/UI, arquitectura, backend Supabase con RLS multi-tenant, frontend desktop con Tauri v2 + React, sistema de alertas y tray, empaquetado firmado para ambas plataformas, CI/CD con GitHub Actions, hardening de seguridad y onboarding del cliente piloto.

2. Alcance del proyecto

Incluido en el alcance:

- Aplicación nativa para macOS (arm64 + x64, notarizada) y Windows 10/11 (firmada EV).
- Backend serverless en Supabase con PostgreSQL, RLS por workspace y Edge Functions.
- Autenticación federada vía Google OAuth con página puente y deep-link.
- Dashboards de horas/coste por proyecto · persona · periodo, con exportación CSV y PDF.
- Sistema de alertas y recordatorios configurables + tray panel con quick-entry.
- Internacionalización (ES · CA · EN), tema claro/oscuro y accesibilidad básica AA.
- Updater nativo con canales stable / internal e infraestructura release automatizada.
- Plan de pruebas (unit · integración · E2E) y QA manual por release.
- Documentación técnica, manual de usuario y formación del cliente piloto.
- Onboarding del primer cliente y hipercare durante 2 semanas posteriores al go-live.

Fuera del alcance (presupuestable aparte):

- Apps móviles nativas iOS / Android.
- Integraciones con sistemas ERP / CRM del cliente.
- Migración de datos desde herramientas previas (Toggl, Harvest, etc.).
- Plan de marketing, branding corporativo o landing comercial avanzada.
- SLA superior al ofrecido en el plan de mantenimiento contratado.

3. Equipo y tarifas

Equipo de dos ingenieros con perfiles complementarios. La asignación de horas por fase refleja la especialización de cada miembro.

Miembro	Rol	Especialización	Tarifa
Albert M.	Lead Desktop & Frontend Engineer	Tauri · React · TS · UX implementación	35 € / h
David	Backend & DevOps Engineer	Supabase · PostgreSQL · CI/CD · seguridad	35 € / h

4. Desglose por fases de desarrollo

Cada fase agrupa un conjunto de entregables verticales con criterios de aceptación claros. Las horas se estiman en base a sprints de 2 semanas y aplican factor de complejidad cuando procede.

#	Fase	Descripción	h Albert	h David	Subtotal
1	**F1. Discovery & análisis de requisitos**	Sesiones con stakeholders, levantamiento de user stories, definición MoSCoW, mapa de procesos.	50 h	50 h	**3.500,00 €**
2	**F2. Diseño UX/UI (Figma)**	Wireframes, sistema de diseño, mockups Mac+Windows, prototipo interactivo, validación con usuarios.	130 h	30 h	**5.600,00 €**
3	**F3. Arquitectura técnica y modelado de datos**	Diagrama de capas, modelo entidad-relación, definición de RLS multi-tenant, contratos de API.	30 h	60 h	**3.150,00 €**
4	**F4. Backend Supabase**	Migraciones SQL, Row-Level Security, triggers, Edge Functions, índices, seeds de prueba.	20 h	140 h	**5.600,00 €**
5	**F5. Frontend desktop — núcleo**	Tauri v2 + React + Vite, routing por rol, layouts, vistas Dashboard/Proyectos/Reportes/Equipo/Ajustes.	280 h	40 h	**11.200,00 €**
6	**F6. Módulo de registro y temporizador**	TimeTrackingContext, quick-entry, sincronización tray ↔ header, persistencia offline.	90 h	20 h	**3.850,00 €**
7	**F7. Dashboards y visualización de datos**	KPIs por proyecto/persona/periodo, exportes CSV/PDF, filtros multi-dimensión.	80 h	30 h	**3.850,00 €**
8	**F8. Tray panel y sistema de alertas**	Icono menú-barra, panel anclado, recordatorios configurables, notificaciones nativas Mac+Win.	70 h	20 h	**3.150,00 €**

9	**F9. Autenticación y OAuth**	Google OAuth federado, página puente web, deep-link hoora://, gestión de sesión segura.	50 h	50 h	**3.500,00 €**
10	**F10. Empaquetado Mac (codesign + notarize)**	Cuenta Apple Developer, certificados, scripts notarización, hardened runtime, entitlements.	50 h	30 h	**2.800,00 €**
11	**F11. Empaquetado Windows (signing + MSI)**	Certificado EV, firma Authenticode, MSI installer, autorun, gestión SmartScreen.	50 h	30 h	**2.800,00 €**
12	**F12. Updater nativo y canales (stable/internal)**	tauri-updater, latest.json firmado, infraestructura release, rollback.	30 h	30 h	**2.100,00 €**
13	**F13. CI/CD GitHub Actions**	release.yml stable, desktop-preview.yml internal, tests automáticos, artefactos firmados.	20 h	60 h	**2.800,00 €**
14	**F14. Hardening de seguridad**	CSP, capabilities mínimas Tauri, auditoría RLS, gestión de secretos, logs.	30 h	50 h	**2.800,00 €**
15	**F15. Internacionalización y accesibilidad**	i18n ES/CA/EN, contraste, navegación teclado, lector de pantalla.	50 h	10 h	**2.100,00 €**
16	**F16. Paleta de comandos y atajos**	Command palette inspirada en Raycast/Linear, atajos por vista.	30 h	5 h	**1.225,00 €**
17	**F17. Pruebas (unit · integración · E2E)**	Vitest, Playwright, smoke tests por release, pruebas de carga Supabase.	60 h	60 h	**4.200,00 €**
18	**F18. QA manual y bug-fix por release**	Validación Mac arm/x64 + Windows 10/11, bug-bash, regresión visual.	50 h	40 h	**3.150,00 €**
19	**F19. Onboarding cliente y despliegue piloto**	Provisionado de workspace, importación de datos, formación 1ª línea, soporte hipercare 2 semanas.	35 h	35 h	**2.450,00 €**
20	**F20. Documentación técnica y de usuario**	Manual de usuario, guía de admin, runbook DevOps, ADRs, README repos.	40 h	30 h	**2.450,00 €**
21	**F21. Gestión de proyecto, sprints y reuniones**	Daily, sprint planning, review, retro, gestión backlog ClickUp, comunicación cliente.	80 h	80 h	**5.600,00 €**
Σ	**TOTAL DESARROLLO**	*21 fases*	**1325 h**	**900 h**	**77.875,00 €**

HORAS TOTALES	HORAS ALBERT	HORAS DAVID	COSTE DESARROLLO

2225 h	**1325 h**	**900 h**	**77.875,00 €**

5. Servicios cloud, SaaS y licencias

Costes recurrentes de infraestructura y herramientas necesarias durante la fase de desarrollo (estimadas para 12 meses). Estos costes se trasladan al cliente como partida diferenciada al margen del esfuerzo de ingeniería.

Servicio	Descripción	Periodicidad	Importe (€)
Apple Developer Program	Notarización + certificados Mac	anual	99,00 €
Certificado EV Code Signing Windows	Sectigo/DigiCert para firma Authenticode	anual	549,00 €
Supabase Pro	Auth + PostgreSQL + RLS + Edge Functions	12 meses	300,00 €
Supabase add-ons (storage, egress)	Buffer para crecimiento durante desarrollo	12 meses	180,00 €
Vercel Pro (landing + bridge auth)	Página puente OAuth + landing pública	12 meses	240,00 €
Sentry Team (error monitoring)	Errores + performance desktop	12 meses	312,00 €
Dominio + DNS (hoora.app)	Registro + Cloudflare Pro	anual	60,00 €
Google Workspace Business	2 cuentas (correo, drive, calendario)	12 meses	288,00 €
Figma Professional	2 editores	12 meses	360,00 €
GitHub Team	2 seats privados + Actions minutos	12 meses	96,00 €
ClickUp Business	Gestión proyecto, sprints	12 meses	288,00 €
LLM API budget (OpenAI/Anthropic)	Asistente IA prescriptivo en desarrollo	estimado fase dev	1.500,00 €
TablePlus / DBeaver Pro	Gestión Supabase y debug SQL	anual	79,00 €
Notion Plus	Wiki interna + ADRs	12 meses	240,00 €

Σ SERVICIOS	*Coste agregado fase desarrollo*	≈ 12 meses	**4.591,00 €**

6. Hardware y herramientas one-shot

Inversión puntual en equipamiento necesario para builds firmados, pruebas reales en hardware no virtualizado y sesiones de despliegue en cliente.

Equipo	Uso	Importe (€)
PC Windows test (NUC i5 16 GB)	Validación builds + Smart Screen + telemetría	850,00 €
Apple Mac mini M2 (CI / build local)	Notarización local + smoke tests release	899,00 €
Licencias S.O. Windows 10/11 Pro	VM de prueba + máquina física	250,00 €
Misc. hardware (HDMI, dock, cables)	Periféricos sesiones piloto	150,00 €
Σ HARDWARE	*One-shot*	**2.149,00 €**

7. Resumen económico

Desarrollo (esfuerzo ingeniería)	77.875,00 €	21 fases · 2 devs
Servicios cloud + SaaS + licencias	4.591,00 €	12 meses
Hardware y herramientas one-shot	2.149,00 €	compra única
BASE DEL PROYECTO	**84.615,00 €**	**suma directa**
Contingencia (10 %)	8.461,50 €	imprevistos · alcance
Margen comercial (15 %)	13.961,48 €	estructura · gestión
SUBTOTAL (sin IVA)	**107.037,98 €**	
IVA (21 %)	**22.477,97 €**	
TOTAL (con IVA)	**129.515,95 €**	**precio final cliente**

BASE	SUBTOTAL	IVA 21 %	TOTAL
84.615,00 €	**107.037,98 €**	**22.477,97 €**	**129.515,95 €**

8. Plan de mantenimiento post-lanzamiento (opcional)

Una vez entregada y aceptada la versión de producción, el cliente puede contratar uno de los siguientes planes de mantenimiento. Los importes son mensuales y se facturan en periodos trimestrales por adelantado.

Plan	Cobertura	Periodicidad	€ / mes	€ / año
Mantenimiento básico	Patches críticos + uptime monitoring + hosting Supabase Pro	mensual	380,00 €	4.560,00 €
Soporte funcional 8x5	8h/mes de evolutivo + bug-fix priorizado, SLA 24h	mensual	1.100,00 €	13.200,00 €
Evolutivo Premium	20h/mes + roadmap conjunto + 1 release evolutiva trimestral	mensual	2.900,00 €	34.800,00 €

9. Calendario tentativo

El proyecto se planifica en sprints de 2 semanas. Las fases pueden solaparse parcialmente para optimizar la carga del equipo. Duración estimada total: ~7 meses naturales desde la firma del contrato hasta el go-live, más 2 semanas de hipercare.

Mes 1	F1 Discovery · F2 Diseño UX/UI · F3 Arquitectura
Mes 2	F4 Backend Supabase · F5 Frontend núcleo (inicio)
Mes 3	F5 Frontend núcleo · F6 Temporizador · F7 Dashboards
Mes 4	F8 Tray + alertas · F9 OAuth · F15 i18n / a11y
Mes 5	F10 Pack Mac · F11 Pack Windows · F12 Updater · F13 CI/CD
Mes 6	F14 Hardening · F16 Command palette · F17 Tests
Mes 7	F18 QA manual · F19 Onboarding cliente · F20 Documentación · Go-live
Mes 7.5	Hipercare (2 semanas)

10. Condiciones comerciales

Forma de pago. 30 % a la firma del contrato · 30 % al cierre de la fase 12 (updater) · 30 % al go-live · 10 % al cierre del periodo de hipercare.

Plazo de pago. 30 días naturales desde la fecha de factura. Recargo del 1 % mensual por mora.

Propiedad intelectual. Todos los entregables son propiedad del cliente una vez liquidado el último pago. El equipo conserva el derecho de citar el proyecto en su portfolio.

Garantía. 90 días de garantía sobre defectos críticos no derivados de cambios de alcance, ataques externos o evolutivos solicitados fuera del alcance original.

Confidencialidad. Sometido a NDA recíproco. Datos del cliente cifrados en tránsito y en reposo.

Cambios de alcance. Los cambios sobre el alcance original se gestionan mediante adenda y se facturan a la tarifa horaria vigente.

Validez de la oferta. 30 días naturales desde la fecha de emisión indicada en la portada.