UNIVERSIDAD DE VALLADOLID

Escuela de Ingeniería Informática

Grado en Ingeniería Informática
Mención en Tecnología de la Información

Análisis y diseño de una aplicación web para la gestión sin papel de la empresa H2Vital

---

Autor: **Mario Ramos Diez**

Tutor: **Joaquín Adiego Rodríguez**

Valladolid, 30 de Junio de 2025

Agradecimientos

Deseo expresar mi más profundo y sincero agradecimiento a todas aquellas personas e instituciones que, de una u otra manera, han contribuido de forma significativa a la realización de este proyecto.

En primer lugar, quiero agradecer a mi tutor, Joaquín Adiego Rodríguez, por su orientación y disponibilidad durante el desarrollo de este Trabajo de Fin de Grado.

Agradezco también a la empresa H2Vital por brindarme la oportunidad de participar en un caso real y por su colaboración continua. Mi reconocimiento, en particular, a todos los empleados que compartieron su tiempo y sus valiosos comentarios, contribuyendo a la mejora y el enfoque de este proyecto.

No puedo dejar de expresar mi más profundo agradecimiento a mi familia, especialmente a mis padres y mi pareja, por su apoyo incondicional, su comprensión y su constante aliento a lo largo de toda mi trayectoria universitaria. Su confianza en mis capacidades y su respaldo en cada decisión han sido el pilar fundamental que me ha permitido llegar hasta aquí. Gracias por su paciencia durante las largas horas de dedicación y por comprender las ausencias necesarias para alcanzar este objetivo.

A mis amigos, tanto dentro como fuera del ámbito universitario, les agradezco sinceramente su apoyo emocional, sus palabras de ánimo en los momentos de dificultad y por ayudarme a mantener el equilibrio entre el esfuerzo académico y el disfrute personal.

Finalmente, quiero agradecer a la Universidad de Valladolid y a la Escuela de Ingeniería Informática por ofrecerme el entorno académico, las instalaciones y los recursos necesarios para mi formación y para la realización de este Trabajo de Fin de Grado.

Este proyecto no habría sido posible sin la contribución, directa o indirecta, de todas estas personas e instituciones. A todos ellos, mi más sincero agradecimiento.

Mario Ramos Diez, Valladolid, 30 de Junio de 2025

Resumen

Este Trabajo de Fin de Grado aborda el desarrollo de una aplicación web destinada a digitalizar la gestión administrativa y de clientes de H2Vital, una empresa con más de 25 años de experiencia en la distribución de sistemas para mejorar la calidad del agua y el aire en Valladolid. Con una base de más de 20,000 clientes y un equipo de 11 a 50 empleados, H2Vital busca optimizar sus procesos internos mediante la centralización de datos y la reducción de errores manuales.

El objetivo principal del proyecto es crear una herramienta que se adapte a los flujos de trabajo específicos de la empresa, facilitando la gestión de empleados, papeletas, citas comerciales, ventas, contratos, máquinas y servicios técnicos. Para ello, se adoptó una metodología iterativa, basada en el feedback continuo de los empleados de H2Vital, lo que permitió ajustar la plataforma a sus necesidades particulares. Tras un intenso análisis de los procesos de gestión de la empresa se determinó la arquitectura más apropiada para la realización de un proyecto tan extenso por una sola persona. Finalmente, la aplicación fue desarrollada utilizando tecnologías modernas como Next.js para el *frontend* y *backend*, PostgreSQL como base de datos relacional, Material MUI para la interfaz de usuario y Docker Compose para el despliegue automatizado en un servidor local Linux.

Aunque las pruebas formales aún están pendientes, se espera que la implementación de esta solución mejore significativamente la eficiencia operativa de H2Vital y siente las bases para futuras expansiones, como la incorporación de módulos de contabilidad y almacén. Este proyecto ejemplifica la aplicación práctica de la ingeniería informática en la transformación digital de una empresa del sector.

Índice general

Presentación

La digitalización de procesos administrativos es una necesidad creciente en las empresas modernas, especialmente en aquellas que aún dependen de métodos manuales para gestionar clientes, contratos y servicios. Este Trabajo de Fin de Grado se centra en el desarrollo de una aplicación web para H2Vital, una empresa de Valladolid con más de 25 años de experiencia en la distribución de sistemas para mejorar la calidad del agua y el aire, que cuenta con más de 20,000 clientes y un equipo de 11 a 50 empleados.

El objetivo principal es crear una herramienta que centralice la gestión administrativa de empleados, papeletas, citas comerciales, ventas, contratos, máquinas y servicios técnicos, utilizando una metodología iterativa basada en el feedback continuo de los usuarios. Se emplearon tecnologías de última generación como Next.js, PostgreSQL, Material MUI y Docker Compose para garantizar una solución robusta y escalable. Este proyecto explora las capacidades de la ingeniería informática para la transformación digital de un negocio real.

Capítulo 1

Introducción

1.1. Contexto del TFG

La digitalización de los procesos administrativos y de gestión empresarial ha emergido como una necesidad crítica en un mundo cada vez más interconectado y competitivo. En este escenario, las pequeñas y medianas empresas (PYMES) enfrentan el desafío de modernizar sus operaciones para mantenerse relevantes, especialmente aquellas que aún dependen de sistemas manuales o semi-automatizados basados en papel. H2Vital, una empresa con sede en Valladolid, España, es un ejemplo representativo de este contexto. Fundada hace más de 25 años, H2Vital se especializa en la distribución y mantenimiento de sistemas innovadores para mejorar la calidad del agua y el aire, atendiendo a más de 20,000 clientes y empleando un equipo de entre 11 y 50 profesionales. Esta empresa ha construido una sólida reputación en el mercado local y regional, ofreciendo soluciones técnicas que abarcan desde purificadores de agua domésticos hasta sistemas industriales de filtración de aire.

Sin embargo, a pesar de su éxito, H2Vital enfrenta limitaciones significativas debido a su dependencia de procesos manuales para la gestión administrativa y de clientes. Actualmente, la empresa utiliza registros en papel para gestionar contratos, citas comerciales, ventas, mantenimiento de máquinas y servicios técnicos, lo que resulta en una alta probabilidad de errores humanos, retrasos en la toma de decisiones y dificultades para escalar operaciones. Según estimaciones generales del sector, las PYMES que no adoptan herramientas digitales pueden perder hasta un 20 % de eficiencia operativa anual debido a estos cuellos de botella [1]. Para H2Vital, con un volumen de más de 20,000 clientes y un catálogo de productos que incluye más de 50 tipos de sistemas, estos problemas se amplifican, afectando tanto la satisfacción del cliente como la rentabilidad.

El mercado de la calidad del agua y el aire ha experimentado un crecimiento notable en la última década. Según un informe de la Asociación Española de Empresas de Tratamiento de Aguas, el sector generó ingresos superiores a 1,200 millones de euros en 2024, con un crecimiento anual compuesto (CAGR) del 5.8 % desde 2019 [2]. Este crecimiento se debe a una mayor conciencia ambiental y regulatoria, así como a la demanda de soluciones sostenibles en hogares e industrias. H2Vital, como actor clave en esta industria, está bien posicionada para capitalizar esta tendencia, pero necesita una infraestructura digital que le permita gestionar eficientemente su base de clientes, optimizar su cadena

de suministro y ofrecer un servicio postventa más ágil.

La digitalización no solo aborda estos desafíos, sino que también abre oportunidades significativas. Por ejemplo, una plataforma digital puede integrar datos de ventas y mantenimiento en tiempo real, permitiendo a H2Vital predecir la demanda de repuestos o identificar patrones de uso que mejoren la experiencia del cliente. Además, la automatización de tareas repetitivas, como la generación de informes o la programación de citas, puede liberar hasta un 30 % del tiempo del personal administrativo [3], permitiendo a la empresa enfocarse en innovación y expansión. Este TFG propone desarrollar una aplicación web que centralice y modernice estos procesos, utilizando tecnologías como Next.js, PostgreSQL, Material MUI y Docker Compose, adaptadas a las necesidades específicas de H2Vital.

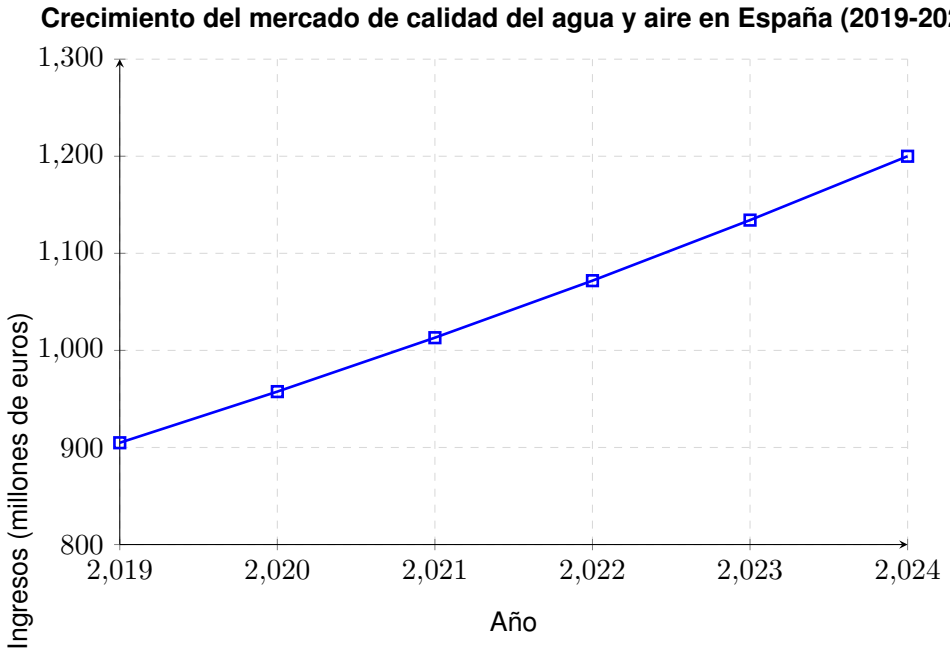


Figura 1.1: Evolución de los ingresos del mercado español de calidad del agua y aire durante el período 2019-2024, mostrando un crecimiento constante del sector.

La importancia de este proyecto trasciende H2Vital, ya que sirve como un caso de estudio para otras PYMES en sectores similares. La transformación digital no solo mejora la eficiencia operativa, sino que también reduce el impacto ambiental al disminuir el uso de papel [4] y facilita el cumplimiento de normativas europeas como el Reglamento General de Protección de Datos (RGPD), que exige una gestión segura de los datos de los clientes. Para H2Vital, esto es especialmente relevante dado que maneja información sensible de más de 20,000 hogares y empresas, lo que requiere un sistema robusto y seguro.

1.2. Motivación

La motivación detrás de este TFG radica en la oportunidad de aplicar conocimientos de ingeniería informática a un problema real y tangible. La experiencia de trabajar con H2Vital ofrece un escenario práctico para explorar cómo las tecnologías web modernas pueden transformar una empresa tradicional.

Personalmente, me impulsa el deseo de contribuir a la modernización de una empresa local con un legado de 25 años, al mismo tiempo que adquiero habilidades en desarrollo de software, bases de datos y despliegue de aplicaciones. Además, el proyecto responde a una necesidad creciente en el mercado laboral, donde las empresas demandan profesionales capaces de liderar iniciativas de transformación digital.

1.3. Objetivos

1.3.1. Objetivo General

El objetivo principal de este Trabajo de Fin de Grado es desarrollar una aplicación web integral que digitalice la gestión administrativa y de clientes de H2Vital, eliminando por completo el uso de papel en la empresa. La aplicación busca adaptarse específicamente a las necesidades y flujos de trabajo existentes en H2Vital, simplificando procesos operativos, asegurando los datos de los clientes y reduciendo costes asociados a la gestión manual. Esta solución no solo pretende centralizar la información en un entorno digital seguro, sino también optimizar la eficiencia operativa mediante la automatización de tareas repetitivas, como la gestión de citas comerciales, contratos, ventas, mantenimiento de máquinas y servicios técnicos. Al eliminar el papel, H2Vital podrá ahorrar costes significativos en materiales de oficina (estimados en un 15 % anual según datos generales del sector), reducir el impacto ambiental y mejorar la trazabilidad de los datos, garantizando un acceso rápido y seguro a la información desde cualquier dispositivo conectado.

1.3.2. Objetivos Específicos

Para alcanzar el objetivo general, se han definido los siguientes objetivos específicos:

- Diseñar e implementar una interfaz de usuario intuitiva y adaptada a los empleados de H2Vital, utilizando el framework Material MUI para garantizar una experiencia de usuario fluida y consistente.
- Crear una base de datos relacional robusta con PostgreSQL que permita almacenar, gestionar y consultar de forma eficiente la información de clientes, contratos, máquinas y servicios técnicos.
- Adaptar la aplicación a los flujos de trabajo específicos de H2Vital, asegurando que módulos como la gestión de citas comerciales y el seguimiento de servicios técnicos reflejen las prácticas actuales de la empresa, pero optimizadas para un entorno digital.
- Asegurar los datos de los clientes mediante la implementación de autenticación de usuarios, cifrado de datos y buenas prácticas de seguridad, cumpliendo con el Reglamento General de Protección de Datos (RGPD).

- Automatizar procesos repetitivos, como la generación de informes de ventas o la programación de citas, para simplificar las operaciones diarias y reducir el tiempo empleado por el personal en tareas administrativas.
- Desplegar la aplicación en un entorno controlado utilizando Docker Compose, lo que facilitará su instalación y escalabilidad futura, al tiempo que reduce costes asociados a infraestructura física.
- Evaluar el impacto de la aplicación en términos de ahorro de costes, mejora de la eficiencia y satisfacción del usuario mediante pruebas con los empleados de H2Vital.

1.4. Metodología

El desarrollo de este proyecto se llevó a cabo mediante una metodología ágil basada en el marco de trabajo Scrum, seleccionada por su flexibilidad y capacidad para incorporar retroalimentación continua de los usuarios finales. Este enfoque permitió adaptar la aplicación a las necesidades reales de H2Vital, asegurando que el producto final fuera funcional y alineado con los flujos de trabajo de la empresa. El proceso se estructuró en las siguientes fases:

1. **Análisis de requisitos**: Se realizaron entrevistas y sesiones de observación con los empleados de H2Vital para comprender sus procesos actuales, identificar puntos de dolor y definir los requisitos funcionales y no funcionales de la aplicación. Esta fase incluyó la elaboración de un documento de especificaciones que sirvió como base para el diseño.
2. **Diseño iterativo**: Se crearon prototipos de baja y alta fidelidad de la interfaz de usuario, utilizando herramientas como Figma para validar el diseño con los usuarios antes de la implementación. Paralelamente, se diseñó el modelo de datos relacional en PostgreSQL, asegurando que cumpliera con los requisitos de escalabilidad y seguridad.
3. **Desarrollo incremental**: La implementación se dividió en sprints de dos semanas, durante los cuales se desarrollaron y probaron módulos específicos de la aplicación (gestión de clientes, citas comerciales, contratos, etc.). Cada sprint concluyó con una demostración al equipo de H2Vital, lo que permitió ajustes basados en su retroalimentación.
4. **Pruebas y validación**: Se llevaron a cabo pruebas unitarias y de integración durante el desarrollo para garantizar la calidad del código. Una vez completada la aplicación, se realizaron pruebas de usuario con empleados de H2Vital para evaluar su usabilidad, funcionalidad y rendimiento, ajustando los últimos detalles antes del despliegue.
5. **Despliegue y entrega**: La aplicación se desplegó en un servidor local utilizando Docker Compose, acompañado de un manual de instalación detallado para facilitar su uso y mantenimiento por parte de H2Vital.

Este enfoque iterativo y colaborativo aseguró que la aplicación no solo cumpliera con los requisitos técnicos, sino que también se integrara de manera natural en las operaciones diarias de H2Vital, maximizando su adopción y utilidad.

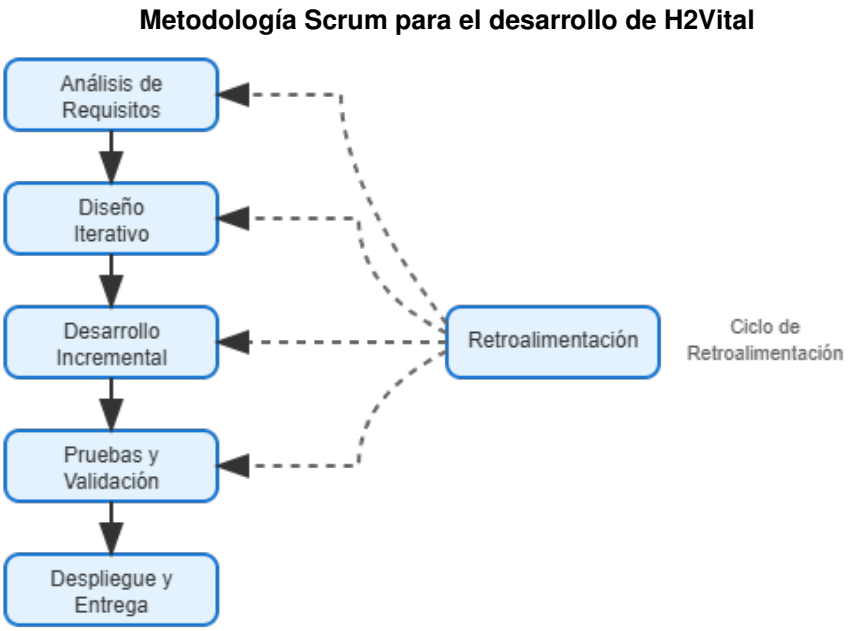


Figura 1.2: Metodología Scrum aplicada al proyecto H2Vital, ilustrando las fases del proceso de desarrollo y el ciclo de retroalimentación continua con los usuarios.

1.5. Organización de la Memoria

Esta memoria está estructurada de manera lógica y secuencial para reflejar el desarrollo completo del Trabajo de Fin de Grado. A continuación, se detalla el contenido de cada capítulo:

- **Capítulo 1: Introducción**: Expone el contexto del proyecto, la motivación personal y académica, los objetivos generales y específicos, la metodología empleada y la estructura de la memoria.
- **Capítulo 2: Procesos de la Empresa**: Análisis en profundidad del funcionamiento actual de H2Vital, incluyendo la gestión manual de contactos, clasificación de papeletas, sistema de referencias, creación de citas, proceso de ventas, contratos y servicio técnico.
- **Capítulo 3: Marco Tecnológico**: Presenta las tecnologías utilizadas (Next.js, PostgreSQL, Material MUI, Docker Compose), su justificación técnica y una comparativa con alternativas disponibles.
- **Capítulo 4: Planificación del Proyecto**: Describe la planificación temporal del TFG, incluyendo cronogramas, hitos clave y gestión de riesgos, así como una estimación de costes y recursos necesarios.

- **Capítulo 5: Análisis de Requisitos**: Especifica de manera detallada los requisitos funcionales y no funcionales del sistema, incluyendo casos de uso, actores involucrados, matriz de trazabilidad y criterios de aceptación para la aplicación H2Vital.
- **Capítulo 6: Análisis del Sistema**: Desarrolla el modelado conceptual del sistema mediante diagramas UML, incluyendo diagramas de casos de uso, secuencia, estados y clases, estableciendo la base arquitectónica del sistema.
- **Capítulo 7: Diseño del Sistema**: Detalla el diseño técnico de la solución, abarcando la arquitectura de software, patrones de diseño aplicados, diseño de la base de datos, interfaces de usuario y consideraciones de seguridad y usabilidad.
- **Capítulo 8: Implementación**: Explica el proceso de desarrollo de la aplicación, incluyendo la configuración del entorno, la implementación de los módulos principales, integración de tecnologías y los retos técnicos superados durante el desarrollo.
- **Capítulo 9: Conclusiones y Trabajo Futuro**: Resume los logros del proyecto, evalúa el cumplimiento de los objetivos, analiza el impacto en H2Vital y propone líneas de trabajo futuro, como la integración de nuevos módulos (contabilidad, almacén, etc.).
- **Anexos**: Incluye un manual de instalación detallado, capturas de pantalla de la aplicación, fragmentos de código relevantes y otros materiales de soporte.

Capítulo 2

Procesos de la Empresa

2.1. Introducción

En un entorno empresarial cada vez más dinámico y competitivo, la transformación digital se ha convertido en una necesidad para las organizaciones que buscan optimizar sus procesos, reducir costos, aumentar la eficiencia y mejorar su capacidad de respuesta frente a los cambios del mercado. Una de las transiciones más comunes en este contexto es la informatización de procesos que, tradicionalmente, se han gestionado de manera manual y en soporte papel.

El objetivo de este trabajo es analizar y justificar la necesidad de conocer en profundidad el funcionamiento actual de la empresa H2Vital que opera completamente en formato papel, como paso previo indispensable para su informatización. Esta comprensión es fundamental para garantizar que la digitalización no solo replique los procesos existentes, sino que los mejore, eliminando redundancias, corrigiendo ineficiencias y adaptándolos a las nuevas herramientas tecnológicas.

Antes de implementar un sistema digital, es imprescindible entender cómo circula la información dentro de la empresa, qué documentos se generan, quién los utiliza, en qué momentos del proceso son requeridos y cuáles son sus funciones específicas. Solo con este conocimiento se puede diseñar una solución informática que responda adecuadamente a las necesidades reales del negocio, sin generar resistencia al cambio ni provocar disrupciones en la operación.

Además, el análisis del modelo actual permite identificar oportunidades de mejora que, muchas veces, no son evidentes cuando se permanece dentro de una lógica completamente manual. Informatizar sin comprender podría llevar a una automatización deficiente, en la que se replican errores estructurales o se implementan sistemas que no se adaptan a la lógica de trabajo de la organización. Realizar esta tarea permitirá obtener un valor a su base de datos de clientes.

Por tanto, conocer el funcionamiento de la empresa en su estado actual no es simplemente un paso previo: es una fase crítica del proceso de transformación. Este conocimiento permite abordar la informatización con una visión estratégica, orientada a la mejora continua, a la sostenibilidad del cambio y a la generación de valor.

En este trabajo se expondrá cómo se lleva a cabo esta fase de análisis, qué elementos deben ser relevados y cómo esta información servirá de base para proponer una solución informática eficaz y

adecuada a las características específicas de la empresa.

2.2. Introducción al Modelo de Negocio

La empresa objeto de este estudio opera en el sector de la venta de sistemas de ósmosis de agua para uso doméstico, un mercado en crecimiento impulsado por la creciente preocupación por la salud, la sostenibilidad y la calidad del agua potable en los hogares. Su modelo de negocio se basa en una estrategia de captación directa de clientes potenciales mediante su participación activa en ferias comerciales, eventos temáticos y espacios públicos de gran afluencia.

A través de estas acciones presenciales, el equipo comercial de la empresa interactúa con los asistentes, les presenta los beneficios del sistema de ósmosis inversa y recopila sus datos de contacto. Posteriormente, estos datos son gestionados internamente y derivados a un equipo de ventas especializadas que realiza visitas domiciliarias para ofrecer una demostración del producto y cerrar la venta.

Este modelo combina una estrategia de marketing experiencial y personal con un enfoque de venta consultiva, donde el contacto humano y la demostración práctica juegan un papel central en la toma de decisión del cliente. A diferencia de la venta puramente online o telefónica, este enfoque busca generar confianza y credibilidad a través de la interacción directa.

No obstante, a pesar de los beneficios de este modelo en términos de conversión, la empresa gestiona actualmente todas sus operaciones de manera manual y en soporte papel. Desde la recolección de datos en los eventos, pasando por la asignación de citas, hasta el seguimiento de los clientes, todo el flujo de información depende de registros físicos y procesos no digitalizados. Esto genera una serie de desafíos, como la pérdida o duplicación de datos, la dificultad para hacer seguimiento en tiempo real, errores de transcripción y una limitada capacidad de análisis comercial.

Por estas razones, se vuelve imprescindible realizar una informatización progresiva y estratégica del modelo de negocio. Sin embargo, para que dicha digitalización sea efectiva y esté alineada con la operativa real de la empresa, primero es necesario comprender a fondo cómo funciona el modelo actual: desde la captación inicial hasta el cierre de la venta. Este análisis permitirá identificar los procesos clave, los puntos de mejora, los cuellos de botella y las oportunidades de automatización, garantizando así que la solución tecnológica propuesta no solo agilice las tareas, sino que también potencie el modelo comercial de la empresa.

En los siguientes capítulos se describe la lógica operativa de la empresa, los roles involucrados, los procesos que sostienen su funcionamiento actual, y cómo estos procesos se integran en su propuesta de valor. Este conocimiento es esencial para plantear una transformación digital coherente, eficaz y adaptada a la realidad del negocio.

2.3. Gestión Manual de los Contactos y Clasificación de Papeletas

Uno de los pilares fundamentales del modelo de negocio de la empresa es la recolección de datos de clientes potenciales durante su participación en ferias, eventos y otros espacios promocionales. Esta tarea es llevada a cabo por personal capacitado, generalmente azafatas, quienes interactúan directamente con los visitantes, explican de forma básica el producto y, si detectan interés, invitan a las personas a rellenar una papeleta con sus datos de contacto.

2.3.1. Contenido de la Papeleta

Cada papeleta física contiene información esencial para el seguimiento comercial posterior. Aunque su diseño puede variar ligeramente según el evento, la estructura básica incluye los siguientes campos:

- **Nombre informal de la pareja**, se refiere al nombre de pila (o nombres) de los interesados, muchas veces sin apellidos, ya que el trato es personalizado y cercano.
- **Código postal**, este dato permite ubicar geográficamente al potencial cliente y evaluar si se encuentra dentro del área de cobertura de visitas domiciliarias. Es la base principal para planificar rutas de los comerciales.
- **Número de teléfono**, es el principal medio de contacto y seguimiento, es importante captar este dato y que sea real.
- **Origen de la papeleta**, se especifica en qué feria o evento fue obtenida, lo cual permite analizar la eficacia de cada acción de captación. Es importante conocer que eventos son más rentables obteniendo papeletas.
- **Observaciones**, campo libre donde la azafata puede anotar detalles relevantes, como el nivel de interés, situación particular del hogar, tipo de agua utilizada, o alguna preferencia horaria para el contacto.
- **Fecha de obtención**, indica el día en que se registró el contacto, importante para organizar la prioridad de llamadas.

Esta información, aparentemente simple, es clave para construir una base de datos que sustente las acciones de venta posteriores. Actualmente, las estadísticas y organización de las papeletas es muy ineficaz; se realiza a mano y requiere una inversión en horas de personal bastante grande, hay que ser consciente que tienes una habitación con todos estos datos organizados en archivadores A-Z, que se pretende desaparezca.

2.3.2. Rol de las Azafatas

Las azafatas desempeñan un papel crucial no solo en la captación de datos, sino también en la calidad de los mismos. Su capacidad de comunicación, simpatía y criterio influyen directamente en

la disposición del visitante para entregar sus datos reales y mostrar interés. Además, en el momento de completar la papeleta, las azafatas suelen clasificar informalmente el contacto según señales que pueden anticipar su potencial como cliente (por ejemplo, si preguntan mucho sobre el producto o si se muestran evasivos).

El proceso depende casi totalmente de la habilidad personal, ya que no existe una herramienta digital que guíe o estandarice la captura de información. Esto también implica que pueden surgir inconsistencias o errores por interpretación subjetiva, escritura ilegible o falta de datos.

2.3.3. Ciclo de Vida de Papeletas

Una vez que las papeletas son recogidas y llevadas a la oficina, se inicia un proceso de clasificación manual. A medida que los comerciales intentan contactar con los interesados, cada papeleta se reetiqueta según el resultado del contacto telefónico. Para ello, se utilizan abreviaturas internas que indican el estado actual de la papeleta. Estas clasificaciones son:

- **Nueva**, papeleta recién llegada que aún no ha sido gestionada.
- **Rellamar RD"**, el contacto mostró interés pero no se pudo cerrar la cita en ese momento; se sugiere llamar de nuevo.
- **No contesta "NC"**, se ha intentado llamar sin obtener respuesta.
- **No perfil "NP"**, el contacto no encaja con el perfil de cliente objetivo (por ubicación, renta, vivienda, etc.).
- **No perfil para rellamar "NP_RD"**, similar al anterior, pero se considera que podría volverse a intentar el contacto más adelante.
- **No quiere "NQ"**, el contacto respondió pero no está interesado en recibir información ni una visita.
- **Repetir**, término interno utilizado cuando una papeleta se reutiliza porque la persona asignada no podrá asistir a la cita; se intenta reasignar.
- **Errónea**, los datos son incorrectos o incompletos (por ejemplo, número de teléfono inválido).

Esta clasificación se realiza escribiendo las abreviaturas directamente en la papeleta, lo que genera un archivo físico de contactos organizados según su estado. Sin embargo, este método presenta varios inconvenientes: dificultad para realizar búsquedas rápidas, falta de trazabilidad del historial de llamadas y una limitada capacidad para obtener métricas en tiempo real sobre la efectividad del equipo comercial o la calidad de los clientes potenciales.

2.4. Las Referencias

Además del proceso de captación directa en ferias y eventos, la empresa utiliza un segundo canal de generación de clientes potenciales: el Sistema de Referencias. Este sistema opera de manera complementaria a las papeletas físicas, y tiene como objetivo aprovechar la red de contactos de clientes satisfechos y del propio equipo comercial, técnico y administrativo para ampliar la base de potenciales compradores.

El funcionamiento es sencillo pero estratégico: tanto clientes actuales como empleados pueden referir a personas de su entorno que pudieran estar interesadas en el sistema de ósmosis domiciliaria. Estas referencias se recogen en un formato similar al de las papeletas, incluyendo los datos básicos del referido (nombre, teléfono, código postal) pero añadiendo un campo adicional: quién realizó la referencia.

Este dato permite establecer una trazabilidad sobre el origen del contacto y evaluar la eficacia de cada canal de recomendación. Por ejemplo, si un cliente refiere a tres personas y una de ellas concreta una venta, el sistema lo registra para posibles incentivos. En el caso del personal interno, también se puede hacer seguimiento de su desempeño o iniciativa comercial más allá de las acciones en terreno.

2.4.1. Incentivos por Referencias

Para estimular el uso de este canal, la empresa ofrece distintos incentivos tanto a clientes como a trabajadores:

- **Clientes**, pueden recibir descuentos en el servicio de mantenimiento del equipo, upgrades de producto, o incluso una bonificación económica directa si la referencia culmina en una venta.
- **Empleados**, en función de la política interna, pueden obtener recompensas como días libres, bonos por rendimiento o reconocimiento dentro del equipo comercial.

Este sistema convierte al cliente en un promotor activo del producto, apoyándose en la confianza que ya ha generado el uso del equipo en su hogar. De igual forma, permite que el personal interno participe de manera más proactiva en la expansión comercial sin depender exclusivamente de las ferias o los eventos.

2.4.2. Registro y Limitaciones Actuales

Actualmente, las referencias también se gestionan en papel, lo que implica varios desafíos similares a los de las papeletas tradicionales:

- **Dificultad para rastrear vínculos**, aunque se anota quién refirió a quién, no existe una base de datos digital que permita visualizar fácilmente estas conexiones o medir cuántas ventas provienen del sistema de referencias.

- **Falta de automatización de recompensas**, el cálculo de incentivos se realiza manualmente, lo que puede generar retrasos o errores.
- **Seguimiento informal**, en ausencia de un sistema digital, la información queda dispersa y muchas veces se pierde la oportunidad de premiar a tiempo o de dar continuidad al ciclo de recomendación.

Informatizar el sistema de referencias permitiría no solo una gestión más eficiente, sino también una mayor capacidad de análisis para detectar patrones, evaluar el rendimiento de los diferentes canales de captación y fortalecer las acciones de fidelización de clientes.

2.5. Creación y Gestión de Citas

Una vez que una papeleta o referencia ha sido clasificada como viable, es decir, cuando el contacto ha mostrado interés y cumple con los criterios básicos del perfil de cliente objetivo, se procede a la creación de una cita comercial. Este es un momento clave en el proceso de ventas, ya que marca la transición de una oportunidad potencial a una interacción directa entre el cliente y el equipo comercial en su propio domicilio.

La creación de la cita se realiza manualmente, a partir de la información contenida en la papeleta y la llamada telefónica de contacto. Es la teleoperadora asignada quien se encarga de organizar la cita, coordinando la disponibilidad del cliente con la del equipo de ventas.

2.5.1. Información Contenida en la Cita

Cada cita generada contiene una serie de datos esenciales que permiten preparar adecuadamente la visita y maximizar las posibilidades de conversión:

- **Nombres informales de la pareja**, utilizados como identificación amigable del núcleo familiar, especialmente útil cuando se han captado datos no formales.
- **Dirección completa del domicilio** , fundamental para planificar la ruta de los comerciales y verificar la viabilidad logística de la visita.
- **Estado civil o situación de convivencia**: Este dato puede ser relevante, ya que en muchos casos la decisión de compra involucra a más de una persona, y en condiciona las estrategias comerciales de la visita.
- **Teleoperadora asignada**, identifica a la persona que organizó la cita, permitiendo dar seguimiento a su gestión.
- **Comerciales responsables**, indica quiénes asistirán a la visita, lo cual es importante tanto para el control de agenda como para la evaluación de desempeño del equipo. Habitualmente solo se asigna un comercial pero a veces para formar comerciales se pueden enviar dos.

- **Fecha y hora de la visita**, detalle central para la organización del calendario comercial.
- **Datos adicionales relevantes**, incluyen hábitos de consumo de agua (agua embotellada, filtro de jarra, agua del grifo, etc.), presencia de niños o personas mayores en casa, tipo de vivienda (piso, casa, etc.), y cualquier otra observación que pueda ser útil para adaptar el discurso comercial.

2.5.2. Reprogramaciones y Cambios

Dado que los clientes no siempre pueden mantener la fecha inicialmente pactada, el sistema manual actual contempla la posibilidad de reprogramar citas. Estas modificaciones también se anotan en papel o en agendas individuales, lo que implica un alto riesgo de errores, solapamientos o pérdida de información.

En muchos casos, las observaciones sobre estos cambios (por ejemplo, reprogramado para el lunes por visita médica", o "prefiere horarios después de las 18:00") son anotadas de forma libre, sin un formato estandarizado. Esto dificulta el análisis posterior o la recuperación de información útil para futuras visitas.

2.5.3. Observaciones en el Seguimiento

A lo largo del proceso, tanto las teleoperadoras como los comerciales pueden dejar observaciones sobre cada cita. Estas notas incluyen comentarios sobre:

- Receptividad del cliente durante la llamada o visita anterior.
- Posibles objeciones detectadas (precio, confianza, desinformación).
- Problemas logísticos detectados (dirección difícil de encontrar, ausencia sin aviso, etc.).

Sin embargo, estas observaciones también se gestionan de manera dispersa, en hojas sueltas o cuadernos personales, lo que limita seriamente su reutilización o análisis sistemático.

2.5.4. Limitaciones del Sistema Manual de Citas

El actual sistema de gestión de citas, basado en registros en papel y comunicaciones informales, presenta una serie de debilidades:

- Dificultad para coordinar agendas: La falta de una base de datos centralizada dificulta la asignación eficiente de comerciales y el control de disponibilidad.
- Falta de trazabilidad: No existe un historial claro de citas anteriores, cancelaciones o reprogramaciones, lo que dificulta el seguimiento adecuado.
- Errores humanos frecuentes: La transcripción manual de datos puede dar lugar a errores que afecten directamente a la puntualidad, preparación o éxito de la visita.

- Imposibilidad de análisis en tiempo real: No se pueden generar estadísticas automáticas sobre tasas de cancelación, tiempos medios entre contacto y visita, ni sobre la conversión por tipo de cliente o zona geográfica.

Estas limitaciones refuerzan la necesidad de una informatización integral del sistema de citas, que permita automatizar la creación, asignación, seguimiento y reprogramación de visitas, así como centralizar toda la información relevante en un sistema accesible y seguro.

2.6. La Venta y Creación de Clientes

Una vez realizada la visita comercial y tras una presentación efectiva del sistema de ósmosis, si el cliente muestra interés y acepta la propuesta, se concreta la venta del producto, lo que conlleva la creación formal del cliente en los registros de la empresa.

Este proceso tiene lugar también de forma manual actualmente, y suele comenzar a partir de la papeleta y la información recabada durante la cita. Sin embargo, es importante destacar que, aunque la cita haya sido agendada con un nombre informal o un miembro específico del hogar, la venta puede quedar registrada a nombre de otra persona del mismo domicilio, por lo que se requiere validar la identidad y los datos completos del comprador al momento de formalizar la operación.

2.6.1. Datos del Cliente

En la creación del cliente se recopilan y consolidan los siguientes campos clave:

- **Nombre completo del comprador**
- **Dirección principal** (donde se instalará el producto)
- **Teléfonos de contacto** (pueden ser múltiples)
- **Direcciones alternativas** si se indica un lugar distinto de instalación o facturación
- **Correo electrónico** (si está disponible)
- **Tipo de cliente** (nuevo, referido, recurrente)

Estos datos son esenciales para mantener una relación continua con el cliente, tanto para la instalación como para futuros servicios de mantenimiento, gestión de garantías o ampliación de productos.

2.6.2. Relación entre Cita y Cliente

Los datos originados en la cita inicial se utilizan como base para crear el expediente del nuevo cliente. No obstante, la falta de digitalización genera varios riesgos:

- Errores de transcripción o pérdida de papeletas

- Dificultad para vincular adecuadamente la venta con su origen (papeleta, evento o referencia)
- Imposibilidad de segmentar la base de clientes por canal de entrada, comercial responsable o fecha de primera cita

La informatización de este flujo permitiría automatizar gran parte de este proceso, estandarizar la creación de fichas de cliente, y generar vínculos directos entre la cita comercial, la venta y el contrato.

2.7. Gestión de Ventas y Contratos

Cada venta realizada implica no solo el cierre comercial, sino también la formalización mediante un contrato, documento legal que recoge todos los detalles del acuerdo entre la empresa y el cliente. Este contrato contiene datos fundamentales tanto para la parte comercial como para los departamentos de instalación, mantenimiento, administración y servicio técnico.

2.7.1. Elementos Clave del Contrato

- Número identificativo del contrato
- Fecha de venta (firma del contrato)
- Fecha estimada de instalación
- Comercial asignado que cerró la venta
- Estado de la venta: exitosa, fallida, cancelada o pendiente de verificación
- Datos de financiación (si aplica): entidad financiera, importe financiado, número de cuotas, importe mensual
- Promociones aplicadas: descuentos por referencias, campañas activas, regalos incluidos

El contrato físico incluye la firma del cliente y del comercial, y en algunos casos, información adicional sobre la necesidad de realizar adecuaciones en el domicilio previo a la instalación.

2.8. Relación con el Servicio Técnico (SAT)

Una vez generado el contrato, se activa automáticamente la necesidad de instalación del equipo. Esto requiere comunicación directa con el equipo del Servicio de Asistencia Técnica (SAT), que recibe los datos esenciales del cliente y la información técnica sobre el modelo vendido.

En el sistema actual, esta transición se realiza de forma manual, mediante el paso físico de documentos o avisos informales, lo que puede provocar retrasos, pérdida de información o instalaciones mal planificadas.

2.8.1. Instalación y Servicio Técnico (SAT)

La instalación del sistema de ósmosis es un paso crítico tanto en la experiencia del cliente como en la continuidad del proceso de venta, es vial que el cliente quede satisfecho ya que la potítica de empresa implica que el equipo se puede devolver antes de cumplir 15 días de la instalación. Una vez firmado el contrato, se abre una cita técnica de instalación.

2.8.2. Información de la Cita de Instalación

Cada cita de instalación debe incluir:

- Tipo de instalación requerida (según modelo del producto y características del domicilio)
- Fecha y hora programada
- Dirección completa del cliente
- Técnico asignado (SAT)
- Observaciones específicas: si el cliente necesita una preinstalación, si se requieren herramientas especiales, si hay mascotas u otros factores a considerar
- Modelo del equipo adquirido
- El técnico también tiene la responsabilidad de devolver retroalimentación tras la instalación, incluyendo la hora de finalización, el estado de la instalación y cualquier incidencia detectada.

2.8.3. Devoluciones o Instalaciones Fallidas

En los casos en que la instalación no pueda realizarse (cliente ausente, condiciones técnicas no adecuadas, cancelación), se abre un registro manual de devolución o reprogramación. En algunos casos, esto también implica la cancelación de la venta, lo cual debe notificarse a los departamentos correspondientes.

El proceso actual carece de trazabilidad digital, lo que dificulta el análisis de tasas de éxito en instalaciones, errores logísticos o desempeño del personal técnico.

Todas las incidencias del SAT debería registrarse con los siguientes datos:

- Fecha de solicitud
- Fecha de visita técnica
- Diagnóstico
- Solución aplicada
- Tiempo estimado de próxima revisión

Los tipos de intervención del SAT son:

- Matenimiento, el modelo de negocio no termina con la instalación del producto. La empresa ofrece un servicio postventa integral, que incluye mantenimiento preventivo. Los sistemas de ósmosis requieren revisiones periódicas para cambiar filtros, limpiar componentes o hacer ajustes técnicos..
- Incidencias, solución de incidencias y atención personalizada al cliente.

Actualmente, estas tareas se gestionan manualmente, por lo que muchos mantenimientos se realizan de forma reactiva (ante una llamada del cliente) y no proactiva.

El mantenimiento y el buen servicio postventa son claves para mantener la relación a largo plazo con el cliente. Sin embargo, en ausencia de un sistema digital centralizado, la empresa no puede hacer un seguimiento eficaz del ciclo de vida del cliente ni ofrecer campañas de fidelización, upgrades o referidos con base en su historial.

2.9. Objetivos

Situación Actual: H2Vital opera completamente con procesos manuales y documentación en papel para vender sistemas de ósmosis doméstica. Su modelo de negocio se basa en captación directa en ferias y eventos, seguido de visitas domiciliarias. En la figura 2.1 podemos ver un diagrama de flujo sencillo del modelo de negocio de H2Vital. En el esquematizamos el ánalisis obtenido de las entrevistas llevadas a cabo con los directivos, encargados de la recogida de datos, ventas y SAT de la empresa.

Los problemas que intentaremos resolver con este proyecto serán los siguientes:

- Ineficiencia operativa. Debido a la gestión manual de papeletas, citas y seguimiento de clientes genera pérdida de tiempo y recursos que intentaremos minimizar.
- Pérdida de información. Exite un riesgo constante de extravío, duplicación y errores de transcripción, que solventaremos con la aplicación desarrollada.
- Falta de trazabilidad, en estos momentos realizar una trazabiliadad de los procesos es muy costoso y las métricas que necesitan los procesos comerciales son complejas, con la informatizción podremos resolver este problema con realativa sencillez.

Finalmente, fijaremos los siguientes objetivos:

- Eliminar la "habitación de archivadores A-Z"
- Automatizar la clasificación y seguimiento de contactos
- Optimizar la gestión de citas y referencias
- Generar valor real de la base de datos de clientes

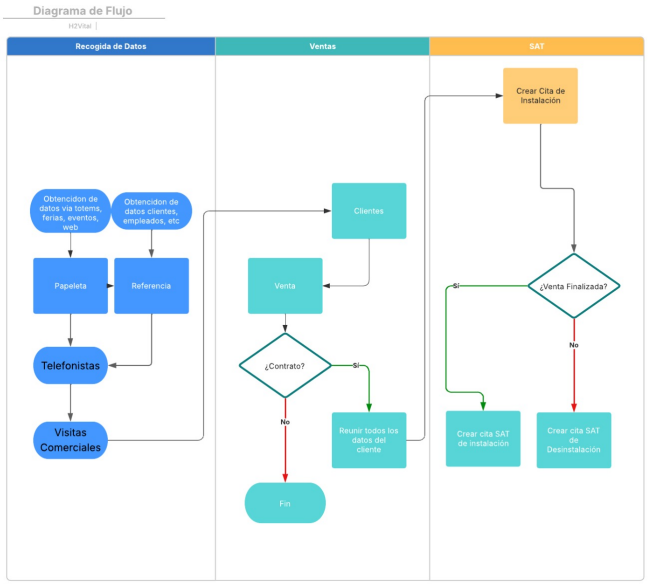


Figura 2.1: Diagrama de Flujo

- Mejorar la coordinación entre equipos

Somos conscientes de la amplitud del proyecto y que probablemente no abarcaremos todos los problemas existentes actualmente pero el enfoque estratégico consistirá en dejar la infraestructura necesaria pra la digitalización progresiva y basada en una comprensión profunda del modelo actual, no una simple replicación digital de procesos manuales, sino una oportunidad de mejora integral del negocio.

Capítulo 3

Marco Tecnológico

3.1. Contexto Tecnológico

Para el desarrollo de la aplicación web de H2Vital, se han seleccionado tecnologías modernas y robustas que garantizan escalabilidad, seguridad y facilidad de mantenimiento. Estas herramientas han sido cuidadosamente elegidas para satisfacer los requisitos específicos del proyecto, como la gestión eficiente de datos administrativos, la protección de información sensible y la creación de una interfaz de usuario intuitiva para el personal de la empresa. La infraestructura tecnológica abarca tanto hardware como software, diseñados para proporcionar una solución integral que soporte las necesidades actuales y futuras de la empresa.

3.2. Comparativa de Tecnologías

La tecnología avanza a un ritmo vertiginoso, ofreciendo cada vez más opciones y soluciones para todo tipo de necesidades. En este capítulo, exploraremos diversas comparativas entre las posibles tecnologías para el desarrollo del proyecto, analizando sus características, ventajas y desventajas. El objetivo es proporcionar una visión clara y objetiva para ayudar a comprender las decisiones tomadas. Abordamos este estudio en tres bloques bien diferenciados:

- **Frameworks frontend** son conjuntos preescritos de código que facilitan y estandarizan el desarrollo de la parte visual de una aplicación web, es decir, la interfaz que interactúa con el usuario. Ofrecen herramientas, componentes y estructuras predefinidas que aceleran el proceso de desarrollo, mejoran la organización del código y permiten crear interfaces de usuario complejas de manera más eficiente.

- **Frameworks de backend** son herramientas y bibliotecas que simplifican el desarrollo de la parte «oculta» de una aplicación web (o aplicación móvil), es decir, el código que no ve el usuario final. Facilitan la creación de la lógica del servidor, la gestión de datos, la autenticación de usuarios y otras tareas esenciales, permitiendo a los desarrolladores enfocarse en la lógica principal de la aplicación.

- **Bases de datos**, una base de datos es un sistema organizado para almacenar, gestionar y recuperar información de forma digital. Sirve para organizar datos de forma que sean accesibles, fáciles de manipular y utilzables para diferentes propósitos.
- **Bibliotecas de UI (User Interface)**

3.2.1. Frameworks Frontend

Vue.js: Framework progresivo diseñado para ser accesible y flexible, ideal para aplicaciones de tamaño mediano y pequeño [5]. Actualmente en la versión 4, está basado en JavaScript/TypeScript, utiliza plantillas declarativas y sintaxis intuitiva. También soporta PWA y SSR con Next.js. Sus características principales son:

- Reactividad. Sistema de reactividad automático que actualiza la UI cuando cambian los datos. Detección automática de dependencias. Actualizaciones eficientes del DOM.
- Arquitectura basada en componentes. Componentes reutilizables y modulares. Encapsulación de lógica, template y estilos. Comunicación entre componentes mediante props y events
- Template syntax declarativa. Sintaxis familiar similar a HTML. Directivas intuitivas (v-if, v-for, v-model). Interpolación de datos simple con
- Facilidad de aprendizaje. Curva de aprendizaje suave. Documentación excelente y detallada. Sintaxis intuitiva para desarrolladores web.
- Flexibilidad. Puedes usar Vue con o sin build tools. Compatible con TypeScript. Múltiples formas de escribir componentes (Options API, Composition API).

Angular: Framework completo de TypeScript desarrollado por Google para construir aplicaciones web robustas y escalables. Sus características principales son:

- TypeScript por defecto: Tipado estático, mejor con utlilidades, detección de errores en tiempo de desarrollo y refactoring más seguro.
- Arquitectura basada en componentes: Componentes reutilizables con decoradores, encapsulación de lógica, template y estilos.
- Inyección de dependencias: Sistema robusto de inyección de dependencias que facilita las pruebas, modularidad y mantenimiento del código.
- Two-way data binding: Sincronización automática entre modelo y vista usando [(ngModel)] y systema de detección de cambios.
- Angular CLI: Herramienta de línea de comandos potente para generar, construir, probar y desplegar aplicaciones.

- RxJS integrado: Programación reactiva con Observables para manejar eventos, HTTP requests y estados complejos.
- Directivas estructurales: *ngIf, *ngFor, *ngSwitch para manipular el DOM de forma declarativa.
- Formularios avanzados: Reactive Forms y Template-driven Forms con validación robusta y manejo de estado.
- Modularidad: Sistema de módulos (@NgModule) para organizar la aplicación en bloques funcionales.
- Testing integrado: Jasmine y Karma incluidos, soporte nativo para unidades de prueba y pruebas de integración.
- PWA ready: Soporte oficial para Progressive Web Apps con Service Workers.

Angular es ideal para aplicaciones empresariales complejas que requieren una estructura sólida, escalabilidad y mantenimiento a largo plazo, aunque tiene una curva de aprendizaje más pronunciada que otros frameworks.

Next.js Framework progresivo diseñado para ser accesible y flexible, ideal para aplicaciones de tamaño mediano y pequeño [6]. Está implementado sobre la biblioteca de React, desarrollada por Meta (Facebook) para construir interfaces de usuario interactivas y reutilizables. Sus principales características son:

- Permite el renderizado híbrido: Combina renderizado del lado del servidor SSR(SSR), generación estática incremental (ISR) y (SSG) renderizado del lado del cliente según las necesidades de cada página.
- Utiliza JSX (JavaScript XML): para la creación de interfaces.
- Sistema de rutas basado en archivos: Las rutas se crean automáticamente basándose en la estructura de carpetas en el directorio pages o app.
- Incluye optimización de imágenes, división automática de código, pre-carga de páginas y minificación sin configuración adicional.
- API Routes: Permite crear endpoints de API dentro de la misma aplicación, facilitando el desarrollo full-stack.
- Soporte TypeScript integrado: Configuración automática de TypeScript sin setup manual.
- CSS y Sass (SCSS) integrados: Soporte nativo para CSS Modules, Sass y CSS-in-JS.
- Image Component: Componente optimizado que maneja automáticamente el redimensionado, lazy loading y formatos modernos como WebP.

Estas características hacen que Next.js sea ideal para crear aplicaciones web rápidas, SEO-friendly y con una excelente experiencia de usuario.

	Vue.js	**Angular**	**Next.js**
Lenguaje base	JavaScript/TypeScript	TypeScript (JavaScript compatible)	JavaScript/TypeScript (React)
Arquitectura	Progresiva, basada en componentes, MVVM	Completa, basada en componentes, MVC/MVVM	Framework de React con SSR/SSG
Curva de aprendizaje	Suave - Fácil para principiantes	Pronunciada - Requiere conocimiento de TypeScript y conceptos avanzados	Moderada - Requiere conocimiento de React
Rendimiento	Excelente	Bueno	Excelente
Casos ideales	Desarrollo rápido, proyectos medianos	Apps empresariales grandes, sistemas complejos	Sitios web con SEO, e-commerce, aplicaciones full-stack

Tabla 3.1: Comparativa de frameworks frontend (2025)

Conclusiones

Tras analizar estas tecnologías, observamos que Vue.js destaca por su facilidad de aprendizaje y flexibilidad; Angular es ideal para aplicaciones empresariales complejas, y Next.js sobresale en rendimiento web y SEO. La elección depende principalmente de las necesidades específicas: si priorizas simplicidad y curva de aprendizaje suave, Vue.js es excelente; si desarrollas aplicaciones empresariales grandes, Angular es la mejor opción; y si necesitas SEO y rendimiento web óptimo, Next.js es la elección ideal.

Tras analizar las comparativas de estas tecnologías, se decidió utilizar Next.js por diferentes razones: curva de aprendizaje, tamaño de la aplicación y rendimiento. Es una arquitectura completa o full stack. Las razones principales para descartar Angular y Vue.js, fue que al centrarse en CSR (Client-Side Rendering), presentan limitaciones en SEO y tiempos de carga inicial, menos adecuados para este caso [7]. Otro factor fundamental fue la curva de aprendizaje y la velocidad de desarrollo de aplicaciones ya que era solo una persona para desarrollar una aplicación compleja en un tiempo corto.

3.2.2. Frameworks Backend

Node.js/Express Node.js y Express son dos tecnologías populares en el mundo del desarrollo web, especialmente en aplicaciones backend. Son un Runtime de JavaScript del lado del servidor con framework web minimalista y flexible.

Esta arquitectura está basada en eventos no bloqueantes. Su principal ventaja es que es el mismo lenguaje tanto en backend como en frontend. Es excelente para microservicios. Una de sus mayores limitaciones es que es Single-Threaded, aunque esto hace que sea sencillo su escalado. De cara al desarrollador, la depuración de aplicaciones es bastante compleja y la arquitectura final de la aplicación depende en gran parte del programador. Los casos de uso ideales son el desarrollo de juegos y aplicaciones en tiempo real, así como desarrollo rápido de prototipos.

Express es un framework sencillo, fácil de aprender y usar, dispone de mucha información en la web y una amplia comunidad de desarrolladores. Node.js es un entorno de ejecución que permite

ejecutar JavaScript en el servidor de manera eficiente y escalable, ideal para aplicaciones que necesitan manejar muchas conexiones concurrentes. Express.js es un framework de Node.js que proporciona una estructura simple pero potente para crear aplicaciones web y APIs, facilitando la creación de rutas, middlewares y manejo de solicitudes HTTP.

Para el desarrollo de aplicaciones backend, Node.js + Express es una excelente combinación. Además, su popularidad y la facilidad de usar JavaScript en ambos lados (frontend y backend) hacen que sea una opción muy atractiva para desarrolladores. De cara al proyecto a realizar esta sería una tecnología perfectamente válida.

Django (Python) Framework web de alto nivel en Python que sigue el principio de "Don't Repeat Yourself"(DRY), lo que significa que busca reducir la duplicación de código y facilita el desarrollo rápido de aplicaciones web seguras y escalables. Una de sus principales ventajas el desarrollo muy rápido para aplicaciones complejas. Por defecto, dispone de seguridad contra inyección de SQL, XSS o CSRF.

Dispone de un interface de administración automático. Después de crear los modelos de base de datos, Django genera automáticamente una interfaz web para gestionar los datos. Esto es extremadamente útil para crear aplicaciones donde se necesita gestionar contenido dinámico.

Django sigue la arquitectura MTV (Modelo-Plantilla-Vista), que es similar al patrón MVC. Aquí, el Modelo se encarga de la estructura de la base de datos, la Vista es responsable de mostrar la información al usuario y la Plantilla es la que genera el HTML dinámico.

Django incluye un sistema ORM (Object-Relational Mapping) que permite a los desarrolladores interactuar con bases de datos relacionales utilizando código Python, sin necesidad de escribir SQL directamente. Esto facilita la gestión de la base de datos y hace que el código sea más limpio y mantenible.

Aunque Django se centra principalmente en el backend, incluye un sistema de plantillas (Django Templates) que permite la creación de interfaces frontend dinámicas. Las plantillas son fáciles de usar y tienen una sintaxis clara que permite mezclar código Python con HTML. Se puede integrar fácilmente con frameworks JavaScript como React, Vue.js o Angular para crear interfaces de usuario ricas y dinámicas. Django se encarga del backend y de la lógica, mientras que el frontend se maneja por separado con estas tecnologías.

Django proporciona con un sistema de pruebas unitarias que te permite escribir pruebas automatizadas de tu aplicación para asegurarte de que tu código funcione como esperas. Es útil para asegurarte de que la funcionalidad se mantiene mientras se desarrollan nuevas características. Una gran ventaja a la hora de probar el desarrollo.

Django es un framework full-stack ideal para construir aplicaciones web robustas y escalables. Aunque es más conocido y utilizado en el desarrollo backend, su sistema de plantillas y su integración con tecnologías frontend modernas lo convierten en una opción viable para construir aplicaciones completas. Es muy adecuado para crear aplicaciones que van desde blogs y sitios simples hasta plataformas grandes y complejas, gracias a su enfoque modular, su facilidad de uso y su extensa documentación.

Como backend, Django es potente, seguro y rápido, con un sistema ORM que facilita la interacción con bases de datos y un sistema de administración muy útil para gestionar el contenido de la aplicación a desarrollar en este proyecto. Como frontend, aunque no es su fuerte, Django ofrece lo necesario para generar interfaces dinámicas a través de su sistema de plantillas y su integración con tecnologías frontend modernas que hemos analizado en el capítulo anterior.

Spring Boot (Java) Spring Boot es un framework de Java que simplifica enormemente el desarrollo de aplicaciones backend. Spring Boot destaca por su autoconfiguración inteligente que detecta las dependencias en el classpath y configura automáticamente los componentes necesarios. Incluye un servidor embebido (Tomcat, Jetty o Undertow) que elimina la necesidad de desplegar en servidores externos, y ofrece Spring Boot Starters que son dependencias preconfiguradas para diferentes funcionalidades como web, JPA, security, etc. Spring Boot ofrece mayor estructura y herramientas enterprise externas a su ecosistema. Es más pesado que otros backend como Node.js. Spring Boot está orientado a aplicaciones enterprise complejas: Spring Boot brilla en sistemas corporativos con lógica de negocio compleja, múltiples integraciones y requisitos estrictos de seguridad. Su arquitectura por capas, patrones empresariales y gestión de transacciones son superiores para aplicaciones bancarias, ERP, sistemas de gestión hospitalaria o plataformas de comercio electrónico de gran escala. Este backend no es apropiado para el desarrollo de este proyecto dada la envergadura del mismo.

Next.js Next.js ha evolucionado significativamente como solución backend completa, especialmente con las App Router y Server Actions introducidas en las versiones recientes. Su capacidad para manejar tanto frontend como backend en un solo framework lo convierte en una opción poderosa para muchos casos de uso. Destacamos las siguientes carácteristicas de Next.js como backend:

- Desarrollo full-stack unificado: Next.js permite que un solo desarrollador o equipo maneje toda la aplicación sin cambios de contexto entre tecnologías. Las API routes, Server Components, y Server Actions proporcionan capacidades backend completas usando JavaScript/TypeScript consistente en todo el stack.
- Tiempo de desarrollo ultra-rápido: Para MVPs, prototipos, y aplicaciones de tamaño medio, Next.js permite ir desde idea a producción mucho más rápido que Spring Boot. Su configuración zero-config y despliegue sencillo en Vercel, Netlify o similares eliminan la complejidad de setup.
- Rendering híbrido avanzado: Next.js ofrece SSR (Server-Side Rendering), SSG (Static Site Generation), ISR (Incremental Static Regeneration), y CSR (Client-Side Rendering) en una sola aplicación. Esta flexibilidad es imposible de lograr con Spring Boot tradicional sin arquitecturas complejas.
- Rendimiento: Con React Server Components, streaming, y optimizaciones automáticas, Next.js puede ofrecer experiencias de usuario superiores en términos de tiempo de carga y interactividad.

El edge computing, esto es, la computación local de los datos y CDN distribution están integrados nativamente.

Uno de los principales inconvenientes de esta tecnología es la escalabilidad ya que Node.js es single-thread lo cual hace que para entornos de tratamiento de imágenes o cálculos matemáticos no sea adecuado. Tampoco dispone de un gestor de transacciones distribuidas

	Next.js	**Node.js/Express**	**Spring Boot**
Lenguaje base	JavaScript/TypeScript (React)	JavaScript/TypeScript	Java
Arquitectura	Full-stack framework con API Routes, Server Components y rendering híbrido	Minimalista, servidor HTTP con middleware	Framework empresarial con arquitectura por capas y IoC
Curva de aprendizaje	Moderada - Requiere conocimiento de React y conceptos de SSR/SSG	Suave - Conceptos básicos de servidor HTTP y middleware	Pronunciada - Requiere conocimiento de Java, Spring ecosystem y patrones enterprise
Rendimiento	Excelente - Optimizaciones automáticas, edge computing, caching inteligente	Bueno - Requiere optimizaciones manuales	Excelente - Especialmente para aplicaciones CPU-intensivas
Casos ideales	Aplicaciones web modernas, MVP, startups, content-heavy sites, full-stack development	APIs personalizadas, microservicios, aplicaciones con requirements específicos	Aplicaciones enterprise, sistemas complejos, alta concurrencia, integración con sistemas legacy

Tabla 3.2: Comparativa de tecnologías backend (2025)

Conclusiones Next.js ofrece una arquitectura más optimizada y productiva: Mientras Express es minimalista y requiere decisiones constantes sobre estructura, routing, middleware, y tooling, Next.js proporciona una arquitectura predefinida con convenciones claras. Las API Routes siguen la estructura de carpetas automática, eliminando configuración de routing manual. Funcionalidades backend avanzadas integradas, mientras que Express es básicamente un servidor HTTP que requiere librerías adicionales para todo. Next.js incluye nativamente middleware, authentication helpers, database connection pooling (con herramientas como Prisma), file uploads, y optimizaciones automáticas. Al ser una arquitectura Full-stack coherente, Next.js unifica todo en un solo proyecto. Características como Server Components y rendering híbrido que solo dispone esta tecnología nos parece uno de los elementos claves para decidirnos por ella como sistema de backend.

3.2.3. Bases de Datos

La elección de la base de datos es una de las decisiones más críticas en el desarrollo de aplicaciones web modernas. Esta decisión impacta directamente en el rendimiento, escalabilidad, mantenibilidad y capacidad de evolución de un proyecto. En el ecosistema de Next.js, framework que hemos elegido, debemos buscar la elección correcta de base de datos; esto puede marcar la diferencia entre una

aplicación ágil y escalable, o una que se vuelva difícil de mantener con el tiempo.

PostgreSQL PostgreSQL es un tipo de base de datos relacional (SQL), sus principales características son:

- Sistema ACID completo con transacciones robustas
- Soporte avanzado para JSON y tipos de datos complejos
- Extensibilidad excepcional (extensiones como PostGIS para datos geoespaciales)
- Rendimiento excelente en consultas complejas
- Soporte nativo para full-text search
- Funciones window, CTEs recursivos, y características SQL avanzadas
- Replicación y particionado sofisticados
- Cumplimiento estricto de estándares SQL

MYSQL MYSQL es un tipo de base de datos relacional (SQL), sus principales características son:

- Amplia adopción y ecosistema maduro
- Excelente rendimiento en operaciones de lectura
- Múltiples motores de almacenamiento (InnoDB, MyISAM)
- Replicación master-slave eficiente
- Menor consumo de recursos que PostgreSQL
- Sintaxis SQL más permisiva (puede ser ventaja o desventaja)
- Integración sólida con aplicaciones web tradicionales

MongoDB MongoDB es un tipo de base de datos documental (NOSQL), sus principales características son:

- Esquema flexible con documentos JSON/BSON
- Escalabilidad horizontal nativa (sharding)
- Consultas rápidas en estructuras de documentos anidados
- Agregación pipeline potente
- Ideal para datos no estructurados o semi-estructurados

- Desarrollo ágil sin necesidad de migraciones de esquema
- Réplicas automáticas y alta disponibilidad
- Por qué PostgreSQL es Superior para Next.js
- Ecosistema JavaScript/TypeScript Excepcional

Si estudiamos las características de las bases de datos presentadas y teniendo en cuenta el ecosistema JavaScript/TypeScript en el que desplegaremos el frontend y el backend, concluimos que PostgreSQL tiene las mejores librerías para el ecosistema Node.js, como son:

- Prisma ORM: Soporte de primera clase con generación de tipos TypeScript automática. Permite escribir consultas a la base de datos usando JavaScript/TypeScript, así que cue cuando defines un esquema de base de datos, Prisma automáticamente crea los tipos de TypeScript para ti.
- Drizzle ORM: ORM moderno con excelente integración TypeScript. Similar a Prisma, pero más moderno y ligero, también te da tipos TypeScript automáticos. Más cercano al SQL tradicional pero con la comodidad de TypeScript.
- node-postgres (pg): Driver nativo robusto y eficiente entre TypeScript y PostgreSQL
- Soporte completo para async/await y Promises. Promises es la forma moderna de manejar operaciones asíncronas (como consultas a base de datos). Async/await: Sintaxis que hace que el código asíncrono se vea como código normal.

ORMs (Object-Relational Mapping) son herramientas de software que simplifican la interacción entre aplicaciones y bases de datos relacionales. Permiten mapear objetos de programación con entidades de la base de datos, lo que facilita el acceso a la información sin necesidad de escribir código SQL explícito. La gestión de datos, por otro lado, abarca todas las operaciones y procesos relacionados con la creación, almacenamiento, recuperación, modificación y eliminación de datos. En resumen, PostgreSQL es la elección superior para Next.js porque combina madurez relacional para aplicaciones empresariales, flexibilidad NoSQL cuando la necesites (JSON, arrays), un ecosistema TypeScript/JavaScript de primera clase, integración nativa con plataformas de deployment modernas y por último un rendimiento excepcional en aplicaciones web concurrentes.

Análisis: PostgreSQL destaca frente a MySQL por su soporte superior para consultas complejas y frente a MongoDB por su capacidad para manejar datos relacionales estructurados, más adecuados para las necesidades administrativas de H2Vital [8].

3.2.4. Bibliotecas de UI (User Interface)

Introducción

Una vez que tenemos claro la selección del backend y el frontend, debemos encontrar una interface que nos facilite un desaroollo ágil dado que el proyecto es enorme. Hemos dejado claro que Next.js

	PostgreSQL	**MySQL**	**MongoDB**
Tipo	Relacional (SQL) con soporte JSON nativo	Relacional (SQL) tradicional	NoSQL documental (BSON)
Arquitectura	MVCC, extensible, cumplimiento estricto SQL	Múltiples motores de almacenamiento, sintaxis permisiva	Colecciones de documentos, esquema flexible
Curva de aprendizaje	Moderada - SQL avanzado + características específicas de PostgreSQL	Suave - SQL familiar, sintaxis más permisiva	Moderada - Paradigma NoSQL, agregation pipeline
Rendimiento	Excelente en consultas complejas, concurrencia alta, operaciones mixtas	Excelente en lectura, bueno en escritura, menos concurrencia	Excelente en escritura/lectura de documentos, limitado en joins
Casos ideales	Aplicaciones Next.js, sistemas que requieren ACID+ flexibilidad JSON, análisis complejos	Aplicaciones web tradicionales, sistemas de lectura intensiva, presupuestos limitados	Aplicaciones con datos no estructurados, prototipado rápido, sistemas de alta escritura

Tabla 3.3: Comparativa de Bases de Datos

se ha consolidado como uno de los frameworks más populares para el desarrollo de aplicaciones web modernas en React, gracias a su enfoque en el rendimiento, la generación de sitios estáticos y dinámicos, y la experiencia del desarrollador. Sin embargo, a medida que crecen las exigencias en diseño y usabilidad, la elección de una biblioteca o framework de interfaz de usuario (UI) se vuelve fundamental para lograr aplicaciones visualmente atractivas, consistentes y fáciles de mantener.

En este contexto, existen múltiples soluciones UI diseñadas para integrarse con React y, por lo tanto, compatibles con Next.js. Algunas de las más populares incluyen Tailwind CSS, Chakra UI, Material UI (MUI), Radix UI, entre otras. Cada una ofrece un enfoque distinto en cuanto a personalización, accesibilidad, rendimiento y curva de aprendizaje.

Esta comparativa tiene como objetivo analizar las principales bibliotecas de UI compatibles con Next.js, evaluando aspectos clave como la facilidad de integración, la experiencia del desarrollador, el rendimiento, el soporte para SSR (Server-Side Rendering), la accesibilidad y la comunidad. Con ello, se busca proporcionar una visión clara que ayude a elegir la solución más adecuada según las necesidades del proyecto.

Tailwind CSS + Headless UI

La principal ventajas de esta biblioteca es:

- Máxima flexibilidad: Control total sobre diseño
- Es un paquete bien optimizado
- Curva de aprendizaje relativamente fácil para desarrolladores CSS
- Ecosistema creciente, hay muchas aportaciones en github como, Headless UI, Heroicons, Tailwind UI

Desventakas de la biblioteca:

- Tiempo de desarrollo elevado, cada componente debe construirse desde cero
- Mantenimiento complejo, requiere disciplina para mantener coherencia
- No dispone de componentes avanzados

Chakra UI

Ventajas:

- API intuitiva, sintaxis limpia y predecible
- Modularidad, importación granular de componentes
- Sistema de tokens, muy bien estructurados
- Comunidad activa, dispone de buena documentación y ejemplos

Desventajas:

- Limitaciones de componentes, dispone de poca variedad
- Ecosistema reducido, dispone de pocos componentes avanzados
- Actualizaciones lentas, las aportaciones al proyecto son menos frecuentes que otros
- Adecuar los componentes exsistentes es complejo y requiere un gran esfuerzo

Material UI (MUI)

Ventajas:

- Ecosistema integral: MUI Core + MUI X + MUI System + MUI Joy
- Componentes de nivel empresarial: DataGrid, DatePicker, Charts listos para producción
- Design System robusto: Basado en Material Design 3, garantiza consistencia visual
- TypeScript nativo: Tipado completo sin configuración adicional
- Soporte SSR/SSG perfecto: Integración sin fricción con Next.js App Router
- Comunidad masiva hay muchas aportacioens enGitHub, documentación exhaustiva

Desventajas:

- Los componentes son algo pesados lo cual puede afectar a su rendimiento de carga
- Estilo rígido es complejo cambiar el look Material Design

- Complejidad de personalización avanzada: Theming profundo requiere conocimiento del sistema interno
- Curva de aprendizaje del sistema, el sistema de tokens complejo para principiantes
- Vendor lock-in: Migrar fuera de MUI es costoso una vez adoptado

Radix UI

Ventajas:

- Accesibilidad excepcional
- Máxima flexibilidad de estilo
- API moderna
- Calidad de código

Desventajas:

- Solo primitivas, requiere diseño de componentes y CSS adicional
- Curva de aprendizaje muy compleja
- Ecosistema limitado, sin componentes de alto nivel
- Tiempo de desarrollo alto

En resumen, Material UI (MUI) es perfecto si te gusta el estilo de Google Material Design o necesitas una librería madura y robusta. Puede ser algo pesado y más rígido en personalización, pero considero que es la mejor decisión, sobretodo por la documentación existente.

	Material UI	Tailwind CSS + Headless UI	Chakra UI	Radix UI
Integración con React	Nativa. Componentes React puros, TypeScript first, context providers incluidos	Muy buena. Headless UI componentes React, Tailwind es CSS puro, setup adicional requerido	Excelente. Diseñado para React, API consistente, TypeScript incluido	Perfecta. Primitivas React avanzadas, ompconents, TypeScript nativo
Personalización	Alta. Theme system robusto, design tokens completos, limitado por Material Design	Máxima. Control total sobre estilos, utility classes infinitas, sin restricciones de diseño	Media. Theme tokens personalizables, menos flexible que otros, componentes más rígidos	Total. Solo lógica sin estilos, CSS libre, control absoluto del diseño
Curva de aprendizaje	Moderada. Documentación excelente, theme system complejo, 1-2 semanas dominio básico	Media-alta. Paradigma utility-first, memorizar clases CSS, 2-4 semanas para fluidez	Fácil. API intuitiva, documentación clara, 3-7 días productividad	Difícil. Conceptos headless avanzados, mucho boilerplate inicial, 3-6 semanas dominio

Tabla 3.4: Comparativa de Librerías UI para Next.js

3.3. Despliegue Hardware y Software

Para el despliegue del software, se ha optado por utilizar Docker como solución principal. A diferencia de los métodos tradicionales, que suelen depender fuertemente del entorno del servidor y presentan dificultades en la replicación y mantenimiento, Docker permite empaquetar la aplicación junto con todas sus dependencias en contenedores ligeros, portables y consistentes. Hay que resaltar que es el software que hemos seleccionado para el backend, frontend, gestor de base de datos, UI y OSRM tienen fuertes dependencias; docker, al empaquetar las versiones, hace que el entorno sea estable y robusto. Soluciones más complejas como Kubernetes, que requieren mayor infraestructura y configuración, no encajan en un proyecto de esta envergadura. Docker ofrece una alternativa más sencilla y ágil para proyectos de menor escala o con requisitos menos dinámicos. Asimismo, a diferencia de plataformas como Heroku, Docker brinda un mayor control sobre el entorno de ejecución, costos más predecibles y mayor flexibilidad para personalizaciones específicas. [9].

Las tecnologías elegidas cumplen con los requisitos clave de H2Vital:

- **Seguridad**: Los servidores Ubuntu Linux con un túnel seguro protegen contra ransomware, mientras que PostgreSQL asegura los datos con cifrado y control de acceso [10, 11].
- **Usabilidad**: Material MUI y el SSR de Next.js ofrecen una interfaz intuitiva y rápida, ideal para usuarios no técnicos [7, 12].
- **Escalabilidad**: Next.js, PostgreSQL y Docker Compose permiten crecer sin complicaciones,

soportando más usuarios y datos en el futuro [6, 8].

- **Eficiencia en Desarrollo**: La integración de frontend y backend en Next.js, junto con Prisma como ORM que proporciona type safety y migraciones automáticas, simplifica significativamente el mantenimiento y las actualizaciones del sistema [7, 13].

En conclusión, este marco tecnológico proporciona una solución integral que satisface las necesidades actuales de H2Vital y está preparada para su expansión futura.

3.3.1. Hardware

La infraestructura de hardware consiste en **dos servidores físicos Ubuntu Linux** ubicados en diferentes locaciones para garantizar la seguridad y disponibilidad de los datos:

- **Servidor Principal**: Aloja la aplicación web de H2Vital, incluyendo el framework Next.js y la base de datos PostgreSQL. Este servidor está configurado para manejar todas las operaciones críticas de la aplicación, desde el procesamiento de solicitudes hasta la gestión de datos.
- **Servidor de Respaldo**: Dedicado exclusivamente a almacenar copias de seguridad, conectado al servidor principal mediante un túnel seguro (por ejemplo, SSH o VPN). Este servidor está aislado de la red pública para minimizar riesgos de ataques, como ransomware. Su situación física es externa a la empresa garantizando la recuperación frente a desastres como incendios, robos, etc.

3.3.2. Sistema Operativo

El uso de **Ubuntu Server** como sistema operativo se justifica por su estabilidad, seguridad y soporte comunitario extenso. Ubuntu es una distribución de Linux ampliamente adoptada, conocida por su facilidad de uso, actualizaciones regulares y versiones de soporte a largo plazo (LTS), que garantizan seguridad y estabilidad durante cinco años por versión [14]. La configuración de dos servidores físicos en diferentes ubicaciones sigue las mejores prácticas de respaldo, como la regla 3-2-1 (tres copias de datos, en dos medios diferentes, con una copia fuera del sitio), asegurando la recuperación ante desastres [10].

3.3.3. Infraestructura Lógica

El túnel seguro entre los servidores utiliza protocolos como SSH sobre una VPN para cifrar la transmisión de datos y privatizarla, protegiendo contra accesos no autorizados. Este diseño ofrece los siguientes beneficios:

- **Seguridad**: El aislamiento del servidor de respaldo y el uso de cifrado (por ejemplo, AES-256) protegen los datos contra ransomware y otros ciberataques.
- **Integridad de Datos**: Las copias de seguridad periódicas permiten una recuperación rápida ante fallos o incidentes.

- **Flexibilidad**: La separación de funciones optimiza cada servidor para su propósito específico, facilitando ajustes futuros según las necesidades de la aplicación.

El concepto es que el servidor de copias de seguridad se enciende a las horas programadas, levanta un tunel VPN y mediante una conexión segura SSH realiza las copias de seguridad y se desconecta. Este sistema hace que aunque un hacker tomara el servidor por una vulnerabilidad no se podría tomar el servidor de copias de seguridad.

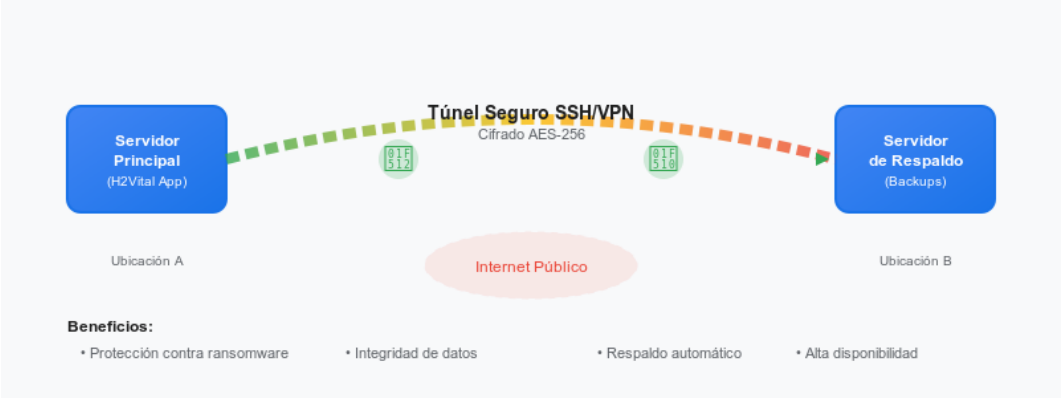


Figura 3.1: Diagrama de los dos servidores conectados vía túnel seguro.

3.3.4. Software

El stack de software seleccionado incluye:

- **Next.js**: Un framework basado en React que unifica el desarrollo del frontend y backend, optimizando el rendimiento y la experiencia del usuario.
- **PostgreSQL**: Un sistema gestor de bases de datos relacional de código abierto, robusto y escalable, ideal para aplicaciones administrativas.
- **Prisma**: Un ORM moderno que facilita la gestión de esquemas de base de datos y proporciona un cliente de base de datos type-safe para TypeScript.
- **Material MUI**: Una biblioteca de componentes de interfaz de usuario basada en Material Design, que asegura interfaces modernas y accesibles.
- **Docker Compose**: Una herramienta para gestionar aplicaciones multi-contenedor en Docker, simplificando el despliegue y asegurando consistencia entre entornos.

A continuación, se detalla cada componente:

Next.js

Next.js es un framework basado en React que permite desarrollar aplicaciones web completas, integrando frontend y backend en un solo entorno. Sus características principales incluyen:

- **Renderizado del Lado del Servidor (SSR)**: Pre-renderiza las páginas en el servidor, mejorando el rendimiento y la optimización para motores de búsqueda (SEO) al reducir el tiempo de carga y facilitar la indexación [7].
- **Generación de Sitios Estáticos (SSG)**: Genera páginas estáticas en tiempo de compilación, ideales para contenido que no cambia frecuentemente, resultando en cargas rápidas y mejor SEO [6].
- **Incremental Static Regeneration (ISR)**: Permite actualizar páginas estáticas de manera incremental, manteniendo el contenido fresco sin sacrificar el rendimiento [6].
- **Optimizaciones de Rendimiento**: Incluye división automática de código, manejo optimizado de imágenes y soporte para formatos modernos como WebP, reduciendo el tiempo de carga [15].
- **Eficiencia en Desarrollo**: Ofrece una estructura de proyecto clara, soporte para herramientas modernas de desarrollo y un ecosistema creciente de plugins [16].
- **Escalabilidad**: Diseñado para manejar aplicaciones complejas y en crecimiento, ideal para empresas en expansión [7].
- **Compatibilidad con AI**: Las actualizaciones recientes permiten integrar soluciones de inteligencia artificial, manteniendo la aplicación competitiva [7].

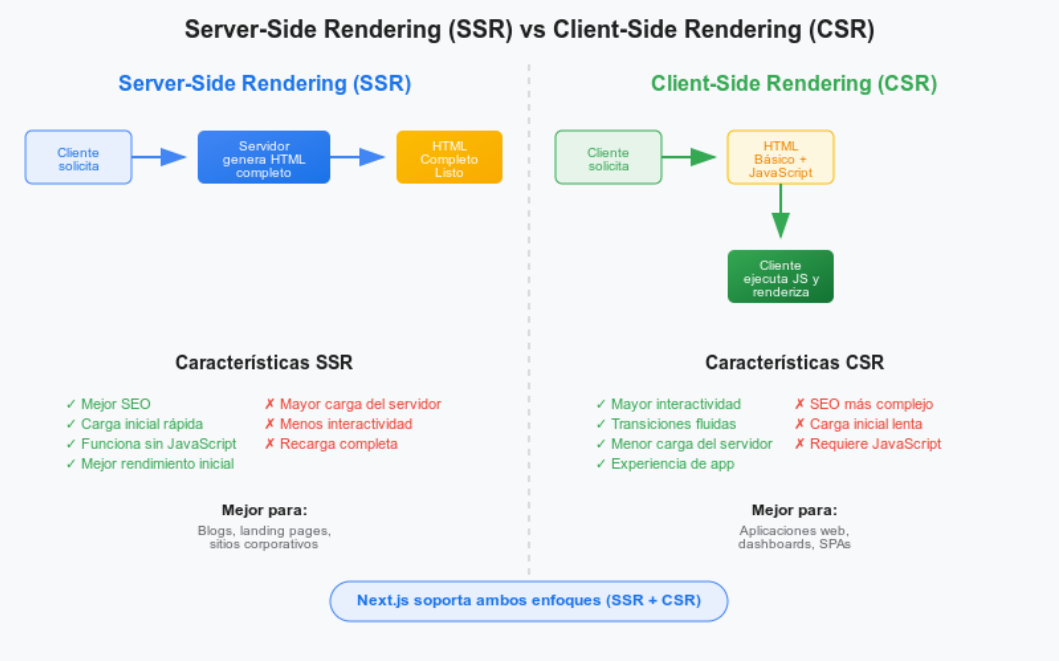


Figura 3.2: Comparación entre Renderizado del Lado del Servidor (SSR) y Renderizado del Lado del Cliente (CSR).

PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos relacional de código abierto, conocido por su robustez y flexibilidad. Sus características destacadas son:

- **Robustez**: Soporta consultas complejas y grandes volúmenes de datos, ideal para aplicaciones administrativas como H2Vital [8].
- **Escalabilidad**: Maneja alta concurrencia y crecimiento de datos sin comprometer el rendimiento [11].
- **Seguridad**: Incluye cifrado, control de acceso a nivel de fila y otras características avanzadas para proteger datos sensibles [17].
- **Extensibilidad**: Permite crear funciones y tipos de datos personalizados, adaptándose a necesidades específicas [18].
- **Cumplimiento ACID**: Garantiza la integridad de las transacciones, crucial para aplicaciones empresariales [19].
- **Soporte Comunitario**: Una comunidad activa proporciona soporte, actualizaciones regulares y una amplia documentación [20].

Prisma

Prisma es un ORM (Object-Relational Mapping) de próxima generación que simplifica el acceso a bases de datos para desarrolladores de TypeScript y JavaScript. Sus características principales incluyen:

- **Type Safety**: Genera automáticamente un cliente de base de datos completamente tipado, eliminando errores de tiempo de ejecución y mejorando la productividad del desarrollador [13].
- **Schema-First Development**: Utiliza un esquema declarativo que sirve como fuente única de verdad para la estructura de la base de datos, facilitando la colaboración en equipo [21].
- **Migraciones Automáticas**: Genera y ejecuta migraciones de base de datos automáticamente basándose en los cambios del esquema, asegurando consistencia entre entornos [22].
- **Query Builder Intuitivo**: Proporciona una API intuitiva y legible para realizar consultas complejas, reduciendo la necesidad de escribir SQL manualmente [23].
- **Integración con PostgreSQL**: Optimizado específicamente para PostgreSQL, aprovechando al máximo sus características avanzadas y tipos de datos nativos [18].
- **Herramientas de Desarrollo**: Incluye Prisma Studio para exploración visual de datos y herramientas CLI para gestión de esquemas y migraciones [24].

Material MUI

Material MUI es una biblioteca de componentes de React basada en Material Design, que ofrece:

- **Diseño Moderno**: Basado en Material Design, proporciona una interfaz visualmente atractiva y consistente [12].
- **Usabilidad**: Componentes predefinidos que facilitan la creación de interfaces intuitivas, ideales para usuarios no técnicos [25].
- **Personalización**: Alta capacidad de personalización para adaptarse a necesidades específicas de diseño [12].
- **Accesibilidad**: Componentes diseñados con estándares de accesibilidad, asegurando usabilidad para todos los usuarios [12].
- **Rendimiento**: Componentes optimizados para un rendimiento eficiente, reduciendo el impacto en la carga de la aplicación [26].

Docker Compose

Docker Compose es una herramienta para definir y ejecutar aplicaciones multi-contenedor en Docker. Sus características incluyen:

- **Despliegue Simplificado**: Gestiona múltiples contenedores mediante un archivo YAML, garantizando entornos consistentes en desarrollo, pruebas y producción [9].
- **Escalabilidad**: Facilita la adición de recursos según la demanda, soportando el crecimiento de la aplicación [27].
- **Portabilidad**: Asegura que la aplicación funcione igual en cualquier sistema compatible con Docker [28].
- **Gestión de Configuración**: Define servicios, redes y volúmenes en un solo archivo, simplificando la gestión de aplicaciones complejas [29].

Capítulo 4

Planificación del Proyecto

4.1. Introducción

Este capítulo describe la planificación del proyecto para H2Vital, estructurada bajo un enfoque iterativo inspirado en metodologías ágiles como Scrum [30]. Se detalla la organización inicial de tareas, el cronograma estimado, la gestión de riesgos y costes, y el alcance planificado del desarrollo. La planificación se apoyó en herramientas como Trello [31] para el seguimiento de tareas y un diagrama de Gantt [32] para visualizar hitos, con un marco temporal estimado de 6 a 8 meses. Este enfoque se diseñó para garantizar flexibilidad ante cambios en los requisitos y optimizar los recursos disponibles, en colaboración con H2Vital y el tutor del TFG.

4.2. Planificación Inicial

La planificación inicial se centró en estructurar las actividades clave para el desarrollo de la aplicación web de H2Vital, adoptando un enfoque iterativo que permite ajustes continuos basados en retroalimentación. El proyecto se organizó en cuatro fases principales: análisis, diseño, implementación y pruebas, diseñadas para solaparse y facilitar la incorporación de nuevos requisitos identificados durante el proceso [33].

Las actividades iniciales se definieron de la siguiente manera:

- Entrevistas con los responsables de H2Vital para la recopilación de requisitos funcionales y no funcionales, con el objetivo de identificar las necesidades administrativas y de gestión de clientes. Las entrevistas se estructuraron según los distintos departamentos clave de la organización:

    1. Dirección General: Se recabó información sobre los objetivos estratégicos de la empresa, sus prioridades operativas y las líneas generales para la implementación del sistema, asegurando que la solución esté alineada con la visión corporativa.
    2. Departamento de Marketing: Se identificaron necesidades relacionadas con la gestión de campañas, segmentación de clientes, seguimiento de posibles clientes y análisis de resultados, aspectos fundamentales para mejorar la captación y fidelización de clientes.

3. Departamento de Ventas: Se analizaron los procesos comerciales, desde la prospección hasta el cierre de ventas, para definir funcionalidades que permitan mejorar el seguimiento de oportunidades.
4. Departamento de Atención al Cliente: Se estudiaron los flujos de interacción con los clientes, tanto en la resolución de incidencias como en el soporte postventa, con el fin de definir requisitos orientados a mejorar la calidad del servicio y la satisfacción del cliente.

- Elaboración de un documento de requisitos inicial, que sirvió como base para las fases posteriores y la definición de entregables.
- Establecimiento de hitos y entregables clave, incluyendo prototipos funcionales para validar funcionalidades con la empresa.
- Creación de un plan de comunicación con H2Vital para alinear expectativas y garantizar una retroalimentación constante.

Los entregables planificados se priorizaron para garantizar una progresión ordenada del proyecto:

- Documento de requisitos inicial: Especificación validada con H2Vital, base para las fases de diseño e implementación.
- Prototipos funcionales: Versiones incrementales para pruebas y retroalimentación con la empresa.
- Documentación técnica: Diagramas de arquitectura y documentación del proceso de desarrollo.
- Memoria final: Documento que recopila el proceso, resultados y conclusiones del proyecto.

Para organizar las tareas y los hitos del proyecto, se elaboró un diagrama de Gantt —detallado en la sección de Gestión del Tiempo— que muestra la secuenciación de actividades y la flexibilidad prevista en el cronograma. Este diagrama se complementó con el uso de Trello [31], una herramienta que facilitó la asignación de tareas y el seguimiento del progreso.

La planificación fue concebida como un proceso adaptable, estructurado en iteraciones basadas en la retroalimentación continua de H2Vital. Las entrevistas iniciales se plantearon como una actividad recurrente durante las fases tempranas del proyecto, no todas fueron presenciales, algunas fueron telefónicas, otras por videoconferencias, reservando las entrevistas personales para los puntos más conflictivos. Esto nos permitió que los requisitos evolucionaran de manera controlada y alineada con las necesidades del cliente.

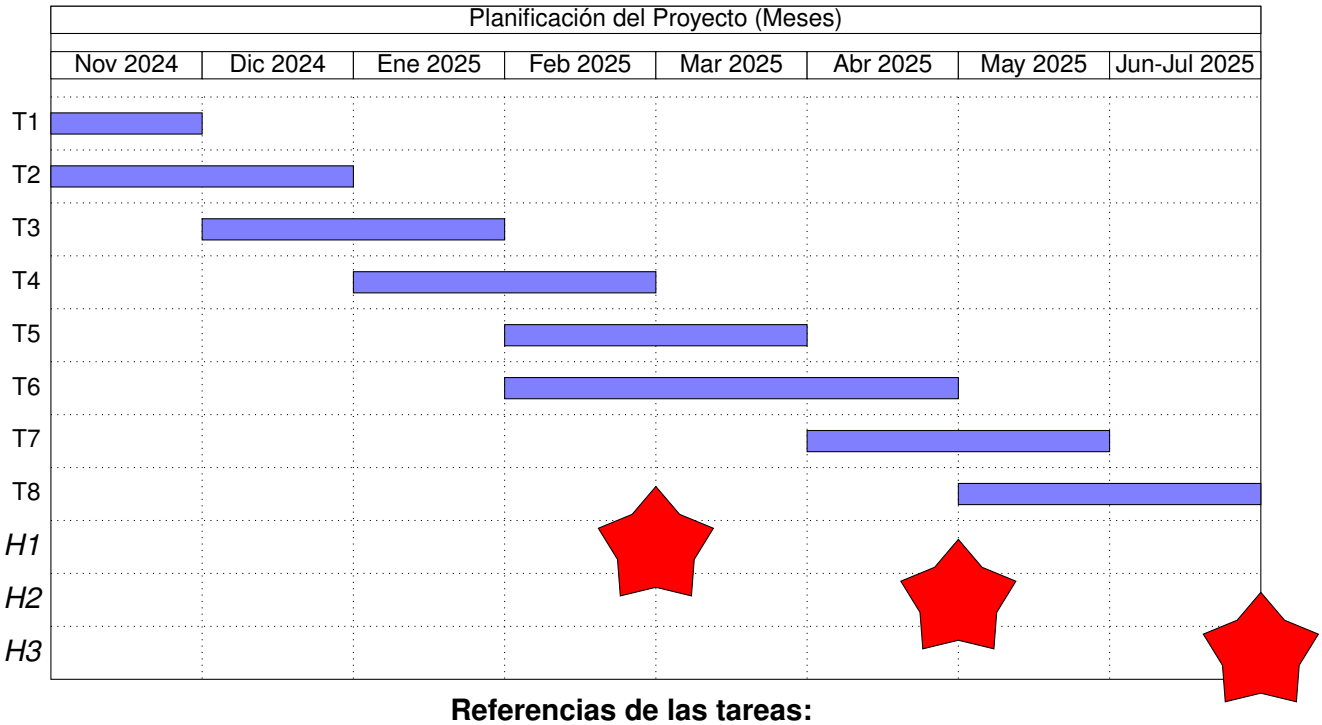
4.3. Gestión del Tiempo

La gestión del tiempo se planificó para un período de 6 a 8 meses, desde noviembre de 2024 hasta junio o julio de 2025, estructurando el proyecto en hitos mensuales para garantizar un avance controlado. El cronograma se organizó mediante un diagrama de Gantt [32], presentado en la Figura 4.1, que detalla las tareas y hitos clave, apoyado por Trello [31] para el seguimiento diario de actividades.

Las tareas y hitos se distribuyeron de la siguiente manera:

- **Mes 1 (Nov 2024)**: Entrevistas con H2Vital, análisis de requisitos y documentación inicial.
- **Mes 2 (Dic 2024)**: Investigación de tecnologías y configuración del entorno de desarrollo con pruebas de despliegue.
- **Mes 3 (Ene 2025)**: Diseño de la base de datos inicial y desarrollo de prototipos iniciales, incluyendo los módulos de empleados y configuración de cuenta.
- **Mes 4 (Feb 2025)**: Implementación de los módulos de referencias, papeletas y componentes comunes, aplicando principios DRY (Don't Repeat Yourself) [34].
- **Mes 5 (Mar 2025)**: Desarrollo de los módulos de citas, clientes, contratos, máquinas y visitas de servicio técnico, con iteraciones sobre la base de datos.
- **Mes 6 (Abr 2025)**: Pruebas exhaustivas y optimización del código [35], priorizando componentes reutilizables.
- **Mes 7 (May 2025)**: Finalización del desarrollo y pruebas de integración.
- **Mes 8-9 (Jun-Ago 2025)**: Despliegue en producción y redacción de la memoria final.

Se planificaron períodos específicos para pruebas y optimización del código, enfocándose en la creación de componentes reutilizables para mejorar la mantenibilidad y escalabilidad de la aplicación [35].



Referencias de las tareas:

- T1: Entrevistas, análisis y documentación
- T2: Investigación y configuración del entorno
- T3: Diseño de la base de datos y prototipos
- T4: Implementación de módulos (referencias, papeletas)
- T5: Desarrollo de módulos (citas, clientes, etc.)
- T6: Pruebas y optimización del código
- T7: Finalización y pruebas de integración
- T8: Despliegue y documentación final

Hitos del proyecto:

- H1: Prototipo inicial
- H2: Prototipo consolidado
- H3: Entrega final

Figura 4.1: Diagrama de Gantt de la planificación inicial del proyecto.

4.4. Gestión de Riesgos

La gestión de riesgos se diseñó como un componente integral de la planificación del proyecto [33], teniendo en cuenta el enfoque iterativo y la colaboración constante con H2Vital. Durante la planificación inicial, se identificaron riesgos potenciales que podrían afectar el desarrollo de la aplicación web, especialmente considerando las iteraciones frecuentes (como las ajustes en la base de datos en el Mes 5) y las pruebas exhaustivas planificadas (Mes 6 y 7). Para cada riesgo, se definieron estrategias de mitigación específicas, apoyadas por herramientas como Trello [31] para monitorear su evolución y revisiones periódicas con H2Vital y el tutor del TFG.

Los riesgos identificados y sus estrategias de mitigación son los siguientes:

- **Retrasos en el cronograma**: Podrían surgir debido a iteraciones adicionales en la base de datos (Mes 5) o retrasos en las pruebas de integración (Mes 7). Para mitigar este riesgo, se adoptó un enfoque ágil [30] con entregas parciales, como prototipos mensuales validados con H2Vital, permitiendo ajustes continuos sin comprometer los plazos finales. Además, Trello [31] se utilizó para monitorear el avance y detectar desviaciones a tiempo.

- **Problemas técnicos**: Posibles dificultades en la integración de módulos (por ejemplo, durante el desarrollo de citas y clientes en el Mes 5) o en la optimización del código (Mes 6). Este riesgo se abordó mediante pruebas tempranas en entornos simulados, como las pruebas de despliegue en el Mes 2, y contando con el soporte técnico del tutor del TFG para resolver incidencias complejas.

- **Cambios en los requisitos**: Las entrevistas continuas con H2Vital, planificadas desde el Mes 1, podrían derivar en nuevos requerimientos a lo largo del proyecto. Para gestionarlo, se diseñó una arquitectura modular [33] que facilita modificaciones sin afectar el núcleo de la aplicación, y se mantuvo una comunicación constante con la empresa mediante reuniones quincenales para validar los prototipos.

- **Limitaciones de recursos**: La dependencia de un equipo personal y un servidor local podría generar cuellos de botella, especialmente durante las pruebas exhaustivas del Mes 6. Este riesgo se mitigó optimizando el uso de recursos mediante pruebas en entornos virtualizados y planificando un presupuesto adecuado desde el inicio.

- **Falta de comunicación con H2Vital**: Dado que las entrevistas continuas son clave para la evolución de los requisitos, una comunicación ineficaz podría generar malentendidos. Para evitarlo, se estableció un plan de comunicación formal en el Mes 1, con reuniones periódicas y un canal directo en Trello [31] para registrar las necesidades y retroalimentación de la empresa.

Para una visión general, la Tabla 4.1 resume los riesgos, su probabilidad, impacto y estrategias de mitigación:

Riesgo	Probabilidad	Impacto	Estrategia de mitigación
Retrasos en el cronograma	Media	Alto	Enfoque ágil con entregas parciales y monitoreo en Trello
Problemas técnicos	Media	Medio	Pruebas en entornos simulados y soporte del tutor del TFG
Cambios en los requisitos	Alta	Medio	Arquitectura modular y comunicación constante con H2Vital
Limitaciones de recursos	Baja	Medio	Pruebas en entornos virtualizados y presupuesto adecuado
Falta de comunicación	Baja	Alto	Plan de comunicación formal y canal en Trello

Tabla 4.1: Resumen de riesgos identificados y estrategias de mitigación

4.5. Alcance Planificado

El alcance del proyecto abarca el desarrollo de una aplicación web integral para H2Vital, diseñada para digitalizar la gestión administrativa y de clientes de la empresa, pasando de procesos basados en papel a un sistema completamente informatizado [3]. Este alcance se definió a partir de los requisitos recopilados durante las entrevistas iniciales con H2Vital (Mes 1, descritas en la sección de Gestión del Tiempo), las cuales destacaron la necesidad de priorizar módulos clave como la gestión de citas y el mantenimiento de máquinas, esenciales para las operaciones diarias de la empresa. La aplicación se planificó para evolucionar iterativamente, ajustándose a las necesidades de H2Vital mediante retroalimentación continua, y se estructuró en módulos funcionales interconectados, diseñados con una arquitectura modular [33] para facilitar modificaciones (como se detalla en la sección de Gestión de Riesgos). Además, se incluyeron medidas para garantizar la estabilidad y continuidad del sistema tras su despliegue en producción.

Los módulos principales de la aplicación, diseñados para interactuar entre sí y proporcionar una gestión integral, son los siguientes:

- **Empleados**: Gestión de información del personal, incluyendo datos personales, roles y asignaciones dentro de la empresa, permitiendo una asignación eficiente de tareas como las visitas técnicas, o visitas comerciales.
- **Configuración de cuenta**: Personalización de perfiles de usuario, con ajustes de preferencias, configuraciones individuales y niveles de acceso según los roles definidos en el módulo de Empleados.
- **Referencias y Papeletas**: Toma de datos inicial de posibles futuros clientes, recopilando información administrativa clave que puede transformarse en una cita comercial programada. Sirve como puente hacia el módulo de Citas y facilita el ingreso de nuevos clientes al sistema.

- **Citas**: Programación y seguimiento de citas comerciales con posibles clientes, permitiendo gestionar fechas, horarios y estados (pendiente, confirmada, cancelada), comercial o comerciales que asisten a la cita, etc. Las referencias y papeletas registradas se integran con el módulo de Clientes para un historial completo y seguimiento de las citas.
- **Clientes**: Gestión integral de datos y relaciones con los clientes, incluyendo historiales de citas, referencias, comunicaciones y contratos asociados, proporcionando una visión completa de cada cliente.
- **Contratos**: Administración de contratos con clientes, desde su creación hasta su seguimiento y renovación, con la capacidad de generar reportes sobre contratos activos y vencidos, integrándose con el módulo de Clientes.
- **Máquinas**: Registro y mantenimiento de equipos, con seguimiento de su estado, historial de uso y necesidades de reparación, permitiendo programar mantenimientos preventivos y coordinar visitas técnicas.
- **Visitas de servicio técnico**: Planificación y seguimiento de visitas técnicas, coordinando horarios, técnicos asignados (vinculados al módulo de Empleados) y resultados de las intervenciones, con notificaciones automáticas para los técnicos sobre sus asignaciones.

Además de los módulos funcionales, se planificó la creación de componentes comunes reutilizables, aplicando principios DRY (Don't Repeat Yourself) [34] para optimizar el desarrollo y garantizar la consistencia de la aplicación. Este enfoque se alineó con las actividades de pruebas y optimización del código [35] en el Mes 6 (descrito en la sección de Gestión del Tiempo), priorizando la mantenibilidad y escalabilidad del sistema. La interfaz de usuario se diseñará con Material-UI (MUI) [36] para ofrecer una experiencia moderna y eficiente, adaptada a las necesidades operativas y visuales de H2Vital.

La base de datos se diseñó para ajustarse iterativamente según los requisitos identificados en las entrevistas con H2Vital, con modificaciones planificadas en paralelo al desarrollo de módulos en el Mes 5 (descrito en la sección de Gestión del Tiempo). Este enfoque garantiza flexibilidad ante cambios en los datos administrativos o de clientes, permitiendo adaptaciones sin comprometer la estructura del sistema.

Para asegurar la continuidad y estabilidad del sistema tras el despliegue en producción (Mes 8-9), se planificaron copias de seguridad automáticas de la base de datos, programadas para ejecutarse diariamente y almacenarse en un servidor seguro, ya sea local o en la nube según el presupuesto disponible de H2Vital. Asimismo, se desarrollarán scripts de mantenimiento automatizados, utilizando herramientas como cron jobs, para realizar tareas como limpieza de datos obsoletos, verificación de integridad del sistema y monitoreo del rendimiento, asegurando un funcionamiento óptimo de la aplicación en el entorno de producción.

Finalmente, el diseño modular de la aplicación [33] permite futuras expansiones para H2Vital, como la integración con sistemas externos (por ejemplo, plataformas de facturación) o la incorporación de nuevos módulos según las necesidades emergentes de la empresa, garantizando la escalabilidad del sistema a largo plazo.

Capítulo 5

Análisis de Requisitos

5.1. Introducción al Análisis

El análisis de requisitos constituye el fundamento metodológico sobre el cual se construye la solución tecnológica para H2Vital. Esta fase crítica del proceso de Ingeniería de Software establece el puente entre las necesidades operativas de la empresa y la especificación técnica del sistema a desarrollar [33].

La metodología aplicada combina técnicas consolidadas de **recopilación de requisitos** con un enfoque iterativo que garantiza la validación continua con los **interesados** de H2Vital. El proceso se fundamenta en los principios de la ingeniería de requisitos moderna, asegurando trazabilidad completa desde las necesidades de negocio hasta la implementación técnica [37].

La complejidad operativa de H2Vital, que gestiona más de 20,000 clientes a través de procesos manuales, requiere un análisis exhaustivo que contemple tanto las funcionalidades explícitas como los requisitos implícitos derivados del dominio del negocio. Este capítulo documenta todos los requisitos identificados, clasificándolos según las categorías estándar de la ingeniería de software y estableciendo las bases para el posterior análisis del sistema y diseño de la solución.

5.1.1. Metodología de Análisis Aplicada

El análisis de requisitos se estructuró siguiendo un enfoque híbrido que combina técnicas tradicionales con metodologías ágiles, adaptándose a las características específicas del proyecto y las limitaciones temporales del TFG [30].

Técnicas de Recopilación de Requisitos:

- **Entrevistas Estructuradas**: Sesiones con empleados de diferentes roles para comprender flujos específicos
- **Observación Directa**: Análisis in-situ de procesos manuales actuales durante jornadas laborales
- **Análisis Documental**: Revisión de formularios, procedimientos y documentación interna existente
- **Talleres Colaborativos**: Sesiones grupales para validar comprensión y priorizar funcionalidades

- **Prototipado**: Validación de conceptos mediante maquetas interactivas durante el proceso

Proceso de Validación :

El proceso de validación se implementó mediante ciclos cortos de retroalimentación que permitieron refinar y ajustar los requisitos según las observaciones de los usuarios finales. Cada sprint de desarrollo incluyó sesiones de demostración que generaron nuevos requisitos o modificaciones a los existentes, garantizando que la solución final se alineara completamente con las necesidades reales de H2Vital.

5.2. Requisitos Funcionales del Sistema

Los **requisitos funcionales** definen las capacidades y servicios específicos que debe proporcionar el sistema H2Vital para satisfacer las necesidades operativas de la empresa [38].

5.2.1. RF-Empleados: Gestión de Personal

RF-EMP-01: Gestión Completa de Empleados

- **Descripción**: El sistema debe permitir la gestión completa del personal de H2Vital con información personal, laboral y credenciales de acceso
- **Funcionalidades Específicas**:
  1. Crear empleado con validación de DNI único en el sistema
  2. Actualizar información personal y laboral de empleados existentes
  3. Asignar roles específicos (Gerente, Coordinador, Comercial, Telefonista, Azafata, Administrativo, Técnico SAT)
  4. Gestionar credenciales de acceso con contraseñas seguras
  5. Visualizar listado completo con filtros por rol y estado
  6. Eliminar empleados mediante baja lógica conservando historial
- **Criterios de Aceptación**:
  1. Validación automática de formato de DNI español
  2. Unicidad garantizada de DNI y email en el sistema
  3. Roles con permisos específicos correctamente aplicados
  4. Contraseñas con requisitos de seguridad (mínimo 8 caracteres, mayúsculas, números)

RF-EMP-02: Control de Acceso Basado en Roles

- **Descripción**: Implementación de sistema de autenticación y autorización que controle el acceso a funcionalidades según el rol del empleado

- **Matriz de Permisos por Rol**:
  - **Gerente**: Acceso completo a todos los módulos y funcionalidades
  - **Coordinador**: Gestión de papeletas, citas y supervisión de equipos
  - **Administrativo**: Gestión de clientes, contratos y documentación
  - **Comercial**: Acceso a citas asignadas, clientes y resultados de ventas
  - **Telefonista**: Procesamiento de papeletas asignadas y contacto con clientes
  - **Azafata**: Creación de papeletas y gestión básica de contactos
  - **Técnico SAT**: Gestión de máquinas y servicios técnicos

5.2.2. RF-Papeletas: Procesamiento de Contactos Comerciales

RF-PAP-01: Gestión del Ciclo de Vida de Papeletas

- **Descripción**: El sistema debe gestionar el ciclo completo de papeletas comerciales desde la captación hasta la conversión en cita programada
- **Estados de Papeleta**:
  1. **NUEVA**: Papeleta recién creada con datos básicos del cliente
  2. **ASIGNADA**: Telefonista asignado automáticamente para contacto
  3. **CONTACTADA**: Cliente contactado exitosamente y calificado
  4. **INTERESADA**: Cliente muestra interés comercial verificado
  5. **CITA_PROGRAMADA**: Cita comercial programada exitosamente
  6. **CONVERTIDA**: Proceso completado con éxito (estado final)
  7. **DESCARTADA**: Proceso fallido o cliente no interesado (estado final)
- **Funcionalidades por Estado**:
  - Transiciones según reglas de negocio definidas
  - Registro de observaciones en cada cambio de estado
  - Auditoría completa de todas las modificaciones

RF-PAP-02: Asignación de Telefonistas

- **Descripción**: Asignación que distribuye papeletas automáticamente entre telefonistas disponibles
- **Criterios de Asignación**:
  1. Carga de trabajo actual del telefonista
  2. Especialización por zona geográfica si aplica
  3. Disponibilidad horaria del empleado
  4. Rotación equitativa para balancear cargas

5.2.3. RF-Citas: Gestión de Citas Comerciales

RF-CIT-01: Programación y Gestión de Citas

- **Descripción**: Sistema completo para programar, gestionar y realizar seguimiento de citas comerciales domiciliarias
- **Funcionalidades Principales**:
  1. Programación de citas con calendario integrado
  2. Asignación de comerciales principales y secundarios
  3. Validación de disponibilidad de comerciales
  4. Gestión de direcciones
  5. Registro de resultado de cita (exitosa, fallida, reagendada)
  6. Conversión automática a contrato en caso de venta

RF-CIT-02: Optimización de Rutas Comerciales

- **Descripción**: Herramientas para optimizar las rutas de comerciales y maximizar la eficiencia de las visitas
- **Características**:
  - Agrupación de citas por zona geográfica
  - Cálculo de tiempos de traslado entre citas
  - Sugerencias de rutas optimizadas
  - Visualización en mapa de citas programadas

5.2.4. RF-Clientes: Administración de Base de Datos de Clientes

RF-CLI-01: Gestión Completa de Información de Clientes

- **Descripción**: Sistema centralizado para gestionar toda la información de clientes actuales y potenciales
- **Información Gestionada**:
  1. Datos personales: nombre, apellidos, DNI, fecha de nacimiento
  2. Información de contacto: teléfonos (hasta 3), email
  3. Direcciones múltiples con información completa
  4. Estado civil, profesión y número de personas en el hogar
  5. Historial completo de interacciones (papeletas, citas, contratos)

6. Preferencias de productos y observaciones comerciales

- **Validaciones y Controles**:
  - Detección automática de duplicados por DNI, teléfono o email
  - Validación de formato de datos (DNI, teléfonos, emails)
  - Control de direcciones con validación postal
  - Auditoría de cambios con trazabilidad completa

5.2.5. RF-Contratos: Formalización y Gestión Contractual

RF-CON-01: Creación y Gestión de Contratos

- **Descripción**: Sistema para formalizar contratos derivados de citas exitosas y gestionar su ciclo de vida completo
- **Proceso de Formalización**:
  1. Generación automática desde cita marcada como exitosa
  2. Asignación de número de contrato
  3. Selección de máquinas y configuración de servicios
  4. Cálculo automático de importes base y financiación
  5. Estados: GENERADO $\rightarrow$ FIRMADO $\rightarrow$ ACTIVO $\rightarrow$ OPERATIVO
  6. Gestión de incidencias y resoluciones

RF-CON-02: Integración con Sistema de Máquinas

- **Descripción**: Vinculación automática de contratos con máquinas específicas instaladas
- **Funcionalidades**:
  - Selección de máquinas del catálogo disponible
  - Asignación de números de serie específicos
  - Control de stock y disponibilidad
  - Programación de instalación y mantenimientos

5.2.6. RF-Máquinas: Gestión de Inventario y Servicios Técnicos

RF-MAQ-01: Catálogo y Control de Inventario

- **Descripción**: Sistema completo para gestionar el catálogo de máquinas y control de inventario
- **Categorías de Máquinas**:

1. **Agua**: Sistemas de osmosis, purificadores, dispensadores
2. **Aire**: Purificadores de aire, ionizadores, filtros
3. **Descanso**: Colchones terapéuticos, almohadas especiales
4. **Hogar**: Productos diversos para mejora del hogar

- **Control de Estados**:
  - DISPONIBLE: Máquina en stock lista para asignación
  - ASIGNADA: Máquina asignada a contrato específico
  - INSTALADA: Máquina instalada en domicilio del cliente
  - OPERATIVA: Funcionamiento normal en cliente
  - MANTENIMIENTO: En servicio técnico o reparación

RF-MAQ-02: Gestión de Servicios Técnicos

- **Descripción**: Sistema para gestionar servicios técnicos, mantenimientos e incidencias
- **Tipos de Servicios**:
  1. Instalación inicial de máquinas en domicilio
  2. Mantenimientos preventivos programados
  3. Reparaciones por incidencias reportadas
  4. Cambios de filtros y consumibles
  5. Retirada de máquinas por finalización de contrato

5.3. Requisitos No Funcionales del Sistema

Los **requisitos no funcionales** establecen las características de calidad, rendimiento y restricciones técnicas que debe cumplir el sistema H2Vital para garantizar su operatividad efectiva en el entorno de producción [39].

5.3.1. RNF-Seguridad: Protección y Acceso

RNF-SEC-01: Autenticación Robusta

- **Descripción**: Sistema de autenticación seguro que proteja el acceso no autorizado al sistema
- **Especificaciones Técnicas**:
  - Autenticación mediante usuario y contraseña con hash bcrypt
  - Sesiones JWT con expiración automática (8 horas de trabajo)

- Bloqueo automático tras 5 intentos fallidos consecutivos
- Registro de todos los intentos de acceso (exitosos y fallidos)

- **Criterios de Medición**:

1. Resistencia a ataques de fuerza bruta verificada
2. Tiempo de respuesta de autenticación menor a 500ms
3. Sesiones con tokens seguros JWT

RNF-SEC-02: Protección de Datos Sensibles

- **Descripción**: Cifrado y protección de información personal y sensible almacenada en el sistema
- **Medidas de Protección**:

- Cifrado de contraseñas con bcrypt (cost factor 12)
- Conexiones HTTPS obligatorias en producción
- Sanitización de inputs para prevenir inyección SQL
- Datos personales ofuscados en logs y auditorías

- **Criterios de Validación**:

1. 100 % de contraseñas hasheadas, nunca almacenadas en texto plano
2. Certificado SSL/TLS válido y actualizado
3. Todas las consultas SQL parametrizadas

RNF-SEC-03: Cumplimiento RGPD

- **Descripción**: Cumplimiento completo del Reglamento General de Protección de Datos europeo
- **Medidas Implementadas**:

- Consentimiento explícito para tratamiento de datos personales
- Derecho al olvido: eliminación completa de datos bajo solicitud
- Portabilidad: exportación de datos en formato JSON estructurado
- Auditoría: registro de accesos y modificaciones

- **Criterios de Medición**:

1. Datos personales con consentimiento registrado
2. Eliminación completa bajo solicitud en menor a 72 horas
3. Auditoría conforme a estándares europeos

5.3.2. RNF-Rendimiento: Performance y Escalabilidad

RNF-PER-01: Tiempos de Respuesta

- **Descripción**: Tiempos de respuesta aceptables para operaciones frecuentes del sistema
- **Objetivos de Performance**:
  - Carga de páginas: menor a 2 segundos en conexión estándar
  - Búsquedas simples: menor a 500ms respuesta del servidor
  - Operaciones CRUD: menor a 1 segundo para inserción/actualización
  - Generación de reportes: menor a 5 segundos para informes estándar
- **Criterios de Medición**:
  1. 95 % de consultas bajo los tiempos objetivo
  2. Monitorización automática de performance
  3. Optimización de consultas SQL con índices apropiados

RNF-PER-02: Escalabilidad

- **Descripción**: Capacidad del sistema para crecer con las necesidades de H2Vital
- **Objetivos de Escalabilidad**:
  - Soporte para hasta 50 usuarios concurrentes sin degradación
  - Base de datos dimensionada para 100,000+ registros por entidad principal
  - Arquitectura preparada para scaling horizontal
  - Optimización de memoria y CPU en servidor

5.3.3. RNF-Usabilidad: Experiencia de Usuario

RNF-USA-01: Interfaz Intuitiva y Accesible

- **Descripción**: Diseño de interfaz que facilite el uso eficiente por parte de todos los empleados
- **Principios de Diseño**:
  - Navegación clara e intuitiva con máximo 3 clics para cualquier función
  - Formularios con validación en tiempo real y mensajes de error claros
  - Responsive design que funcione en desktop, tablet y móvil
  - Consistencia visual siguiendo principios de Material Design
  - Accesibilidad básica conforme a pautas WCAG 2.1 nivel AA

- **Criterios de Medición**:

  1. Satisfacción de usuario mayor a 8/10 en encuestas post-implementación
  2. Tiempo de aprendizaje menor a 2 horas para nuevos usuarios
  3. Tasa de errores de usuario menor a 5 %

5.3.4. RNF-Fiabilidad: Disponibilidad y Recuperación

RNF-FIA-01: Alta Disponibilidad

- **Descripción**: Garantía de disponibilidad del sistema durante horario laboral
- **Objetivos**:
  - Uptime: 99.5 % durante horario laboral (8:00-20:00)
  - Mantenimiento programado: Fuera de horario laboral
  - Tiempo de recuperación: menor a 30 minutos tras incidencia

RNF-FIA-02: Backup y Recuperación

- **Descripción**: Estrategia completa de backup para protección de datos
- **Especificaciones**:
  - Backup automático diario a las 2:00 AM
  - Retención: 30 días de backups incrementales
  - Backup completo semanal con retención de 6 meses
  - Testing de restauración mensual documentado

5.4. Reglas de Negocio

Las **reglas de negocio** establecen las políticas, restricciones y lógica específica del dominio H2Vital que debe ser implementada en el sistema para reflejar fielmente los procesos operativos de la empresa [40].

5.4.1. Reglas de Papeletas

RN-PAP-01: Asignación Automática de Telefonistas

- **Regla**: Cada papeleta nueva debe ser asignada automáticamente a un telefonista disponible según criterios de balanceo de carga
- **Algoritmo**:

1. Verificar telefonistas activos y disponibles
2. Calcular carga actual de papeletas pendientes por empleado
3. Asignar a telefonista con menor carga de trabajo
4. En caso de empate, aplicar rotación circular

- **Excepciones**: Coordinador puede reasignar manualmente si es necesario

RN-PAP-02: Transiciones de Estado Válidas

- **Regla**: Los cambios de estado de papeleta deben seguir flujos específicos y ser realizados por roles autorizados
- **Transiciones Permitidas**:
  1. NUEVA → ASIGNADA (automático del sistema)
  2. ASIGNADA → CONTACTADA (solo telefonista asignado)
  3. CONTACTADA → INTERESADA/DESCARTADA (solo telefonista)
  4. INTERESADA → CITA_PROGRAMADA (coordinador o gerente)
  5. CITA_PROGRAMADA → CONVERTIDA (automático tras cita exitosa)
- **Auditoría**: Todos los cambios quedan registrados con usuario, fecha y observaciones

5.4.2. Reglas de Citas

RN-CIT-01: Validación de Disponibilidad

- **Regla**: No se pueden programar citas si el comercial asignado ya tiene otra cita en un rango de 2 horas
- **Validaciones**:
  - Verificación automática de conflictos de horarios
  - Consideración de tiempos de traslado entre ubicaciones
  - Respeto de horarios laborales (8:00-20:00, Lunes-Sábado)
  - Máximo 4 citas por comercial por día
- **Alertas**: Sistema notifica automáticamente posibles conflictos

5.4.3. Reglas de Contratos

RN-CON-01: Transiciones de Estado Válidas

- **Regla**: Los cambios de estado de contrato deben seguir flujos específicos con justificación obligatoria
- **Transiciones Permitidas**:
  1. GENERADO → FIRMADO
  2. FIRMADO → ACTIVO
  3. ACTIVO → INCIDENCIA
  4. INCIDENCIA → ACTIVO
  5. ACTIVO/INCIDENCIA → FALLIDO
- **Auditoría**: Todos los cambios de estado quedan registrados con usuario, fecha y justificación

5.5. Trazabilidad de Requisitos

La trazabilidad de requisitos tiene como objetivo establecer relaciones claras y documentadas entre las necesidades del negocio, los requisitos funcionales y no funcionales, y los componentes técnicos que los implementan. Este enfoque garantiza una cobertura completa de las necesidades identificadas, facilita el seguimiento de cambios y asegura la coherencia a lo largo del ciclo de vida del sistema [41].

5.5.1. Trazabilidad Requisitos Funcionales - Módulos

En esta sección se presenta la relación entre los requisitos funcionales definidos y los módulos del sistema que los implementan. Esta matriz permite verificar que cada funcionalidad requerida esté correctamente contemplada en la arquitectura del sistema.

Módulo	Requisitos Funcionales	Prioridad	Dependencias
Empleados	RF-EMP-01 a RF-EMP-02	Crítica	Ninguna
Papeletas	RF-PAP-01 a RF-PAP-02	Crítica	Empleados
Citas	RF-CIT-01 a RF-CIT-02	Crítica	Papeletas, Empleados
Clientes	RF-CLI-01	Alta	Papeletas, Citas
Contratos	RF-CON-01	Alta	Clientes, Citas
Máquinas	RF-MAQ-01 a RF-MAQ-02	Media	Contratos

Tabla 5.1: Matriz de trazabilidad requisitos funcionales - módulos

5.5.2. Trazabilidad Requisitos No Funcionales - Componentes

Este capítulo establece las bases sólidas para el desarrollo del sistema H2Vital mediante la especificación detallada y estructurada de todos los requisitos identificados, proporcionando la documentación

necesaria para las fases posteriores de análisis, diseño e implementación del sistema. Esta matriz recoge los requisitos no funcionales identificados y los vincula a los componentes que los satisfacen dentro de la arquitectura del sistema. Con ello, se garantiza que las propiedades de calidad, como la seguridad, el rendimiento y la usabilidad, estén contempladas desde las primeras fases del desarrollo.

Componente	Requisitos No Funcionales	Impacto
Autenticación	RNF-SEC-01, RNF-SEC-02	Crítico
Base de Datos	RNF-PER-01, RNF-FIA-02	Alto
Interfaz Usuario	RNF-USA-01	Alto
Servidor Aplicación	RNF-PER-02, RNF-FIA-01	Alto
Auditoría	RNF-SEC-03	Medio
Backup	RNF-FIA-02	Medio

Tabla 5.2: Matriz de trazabilidad requisitos no funcionales - componentes

5.5.3. Matriz de Validación de Requisitos

Para asegurar que los requisitos han sido correctamente implementados y que cumplen con los objetivos definidos, se establece una matriz de validación. Esta matriz describe el método mediante el cual se verificará cada requisito y los criterios que definen su cumplimiento satisfactorio.

ID Requisito	Método Validación	Criterio Éxito
RF-EMP-01	Pruebas funcionales	Validación DNI 100 %
RF-PAP-01	Test de flujo completo	Estados correctos
RNF-SEC-01	Pruebas seguridad	Resistencia fuerza bruta
RNF-PER-01	Test rendimiento	menor a 2 seg respuesta
RNF-USA-01	Test usabilidad	Satisfacción mayor a 8/10

Tabla 5.3: Matriz de validación de requisitos

Capítulo 6

Análisis del Sistema

6.1. Introducción al Análisis del Sistema

El análisis del sistema constituye la fase de transición entre la especificación de requisitos y el diseño de la solución, proporcionando una representación conceptual del sistema H2Vital que abstrae la complejidad técnica y se centra en la funcionalidad y comportamiento del sistema desde la perspectiva del usuario [33].

Esta fase aplica metodologías consolidadas de **modelado orientado a objetos** y **UML** (Unified Modeling Language) para documentar de manera visual y estructurada las interacciones entre actores y sistema, los flujos de trabajo principales y el modelo conceptual del dominio del problema [42]. El enfoque metodológico seguido garantiza que todos los aspectos funcionales identificados en la fase de análisis de requisitos sean modelados de forma coherente y trazable.

El análisis del sistema H2Vital se estructura en cinco componentes principales que proporcionan diferentes perspectivas complementarias del sistema:

- **Modelado de Actores**: Identificación y caracterización de los usuarios del sistema con sus roles y responsabilidades específicas
- **Diagramas de Casos de Uso**: Representación de las funcionalidades del sistema desde la perspectiva del usuario final
- **Modelo de Dominio**: Conceptualización de las entidades del negocio y sus relaciones
- **Diagramas de Estados**: Modelado del ciclo de vida de las entidades principales del sistema
- **Diagramas de Secuencia**: Modelado de las interacciones temporales entre actores y sistema para escenarios críticos

La metodología aplicada combina técnicas tradicionales de análisis orientado a objetos con validación continua mediante prototipos y sesiones de trabajo con los stakeholders de H2Vital, asegurando que los modelos conceptuales reflejen fielmente los procesos y necesidades reales de la empresa [43].

6.2. Modelado de Actores del Sistema

El modelado de actores identifica y caracteriza a todos los usuarios que interactúan con el sistema H2Vital, definiendo sus roles, responsabilidades y permisos de acceso específicos [44]. Esta identificación resulta fundamental para establecer el control de acceso basado en roles y garantizar que cada funcionalidad del sistema sea accesible únicamente por los actores autorizados.

6.2.1. Identificación y Caracterización de Actores

La estructura organizacional de H2Vital define claramente los roles y responsabilidades de cada empleado, lo que permite una identificación precisa de los actores del sistema y sus necesidades específicas de acceso a la información.

Actor Principal: Gerente

Descripción del Actor: El Gerente representa el rol de máxima responsabilidad en H2Vital, requiriendo acceso completo a todos los módulos del sistema para supervisión general y toma de decisiones estratégicas.

Responsabilidades Principales:

- Supervisión general de todos los procesos operativos
- Análisis de métricas de rendimiento y KPIs empresariales
- Gestión de empleados y asignación de roles
- Aprobación de decisiones críticas y excepciones
- Generación de reportes ejecutivos y análisis estratégicos

Casos de Uso Accesibles: Todos los casos de uso del sistema

Actor Operativo: Coordinador

Descripción del Actor: El Coordinador gestiona las operaciones diarias de captación comercial y asignación de recursos, actuando como enlace entre la dirección y el equipo operativo.

Responsabilidades Principales:

- Supervisión del equipo de telefonistas y azafatas
- Asignación y reasignación de papeletas según carga de trabajo
- Programación de citas comerciales y optimización de rutas
- Control de calidad en el proceso de captación
- Generación de reportes operativos para dirección

Casos de Uso Accesibles: CU-01 (Gestionar Empleados), CU-02 (Procesar Papeletas), CU-03 (Gestionar Citas)

Actor Administrativo: Administrativo

Descripción del Actor: El Administrativo se encarga de la gestión formal de clientes y la formalización de contratos, asegurando el cumplimiento de procesos administrativos y legales.

Responsabilidades Principales:

- Gestión completa de la base de datos de clientes
- Formalización y seguimiento de contratos
- Control de documentación y cumplimiento legal
- Gestión de incidencias contractuales
- Coordinación con entidades financieras para financiaciones

Casos de Uso Accesibles: CU-04 (Administrar Clientes), CU-05 (Formalizar Contratos)

Actor de Ventas: Comercial

Descripción del Actor: El Comercial ejecuta las citas domiciliarias programadas, presentando productos y cerrando ventas mediante técnicas comerciales especializadas.

Responsabilidades Principales:

- Ejecución de citas domiciliarias asignadas
- Presentación de productos y servicios de H2Vital
- Cierre de ventas y formalización inicial de contratos
- Registro de resultados y observaciones de citas
- Seguimiento de clientes potenciales interesados

Casos de Uso Accesibles: CU-03 (Consultar Citas Asignadas), CU-04 (Consultar Clientes), CU-05 (Iniciar Formalización)

Actor de Contacto: Telefonista

Descripción del Actor: El Telefonista se especializa en el contacto telefónico con clientes potenciales, calificando su interés comercial y preparando el terreno para citas programadas.

Responsabilidades Principales:

- Procesamiento de papeletas asignadas automáticamente

- Contacto telefónico con clientes potenciales
- Calificación de interés comercial del cliente
- Actualización de estados de papeleta según resultado
- Registro de observaciones y preferencias del cliente

Casos de Uso Accesibles: CU-02 (Procesar Papeletas Asignadas), CU-04 (Consultar/Actualizar Clientes)

Actor de Captación: Azafata

Descripción del Actor: La Azafata es responsable de la captación inicial de clientes potenciales y la creación de papeletas con información básica para el proceso comercial.

Responsabilidades Principales:

- Captación de clientes potenciales en eventos y espacios públicos
- Creación de papeletas con datos básicos del cliente
- Registro de interés inicial y observaciones relevantes
- Identificación de oportunidades comerciales inmediatas

Casos de Uso Accesibles: CU-02 (Crear Papeletas), CU-04 (Crear/Consultar Clientes Básico)

Actor Técnico: Técnico SAT

Descripción del Actor: El Técnico SAT gestiona el inventario de máquinas y ejecuta servicios técnicos especializados en domicilios de clientes.

Responsabilidades Principales:

- Gestión del inventario de máquinas y productos
- Instalación de equipos en domicilios de clientes
- Mantenimiento preventivo y correctivo de máquinas
- Gestión de incidencias técnicas reportadas
- Control de stock y solicitud de material

Casos de Uso Accesibles: CU-06 (Gestionar Máquinas), CU-05 (Consultar Contratos para Servicios)

6.3. Diagramas de Casos de Uso

Los diagramas de casos de uso proporcionan una visión funcional del sistema desde la perspectiva de los actores, mostrando las principales funcionalidades y sus interacciones [45].

6.3.1. CU-01: Gestión de Empleados

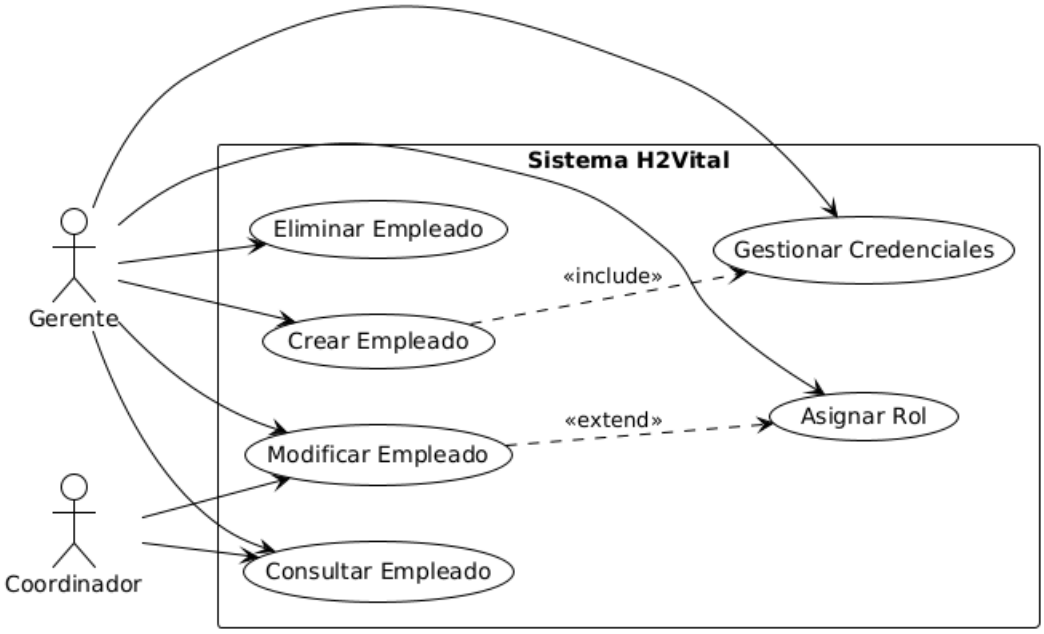


Figura 6.1: Diagrama de casos de uso: Gestión de Empleados

Objetivo: Administrar la información completa de los empleados de H2Vital, incluyendo datos personales, laborales y credenciales de acceso al sistema.

Actores Principales: Gerente, Coordinador

La Figura 6.1 muestra el diagrama completo de casos de uso para la gestión de empleados.

Casos de Uso Incluidos:

- **Crear Empleado:** Registro de nuevos empleados con validación de datos
- **Consultar Empleado:** Búsqueda y visualización de información de empleados
- **Modificar Empleado:** Actualización de datos personales y laborales
- **Eliminar Empleado:** Baja lógica de empleados del sistema
- **Asignar Rol:** Definición de permisos y responsabilidades
- **Gestionar Credenciales:** Administración de accesos al sistema

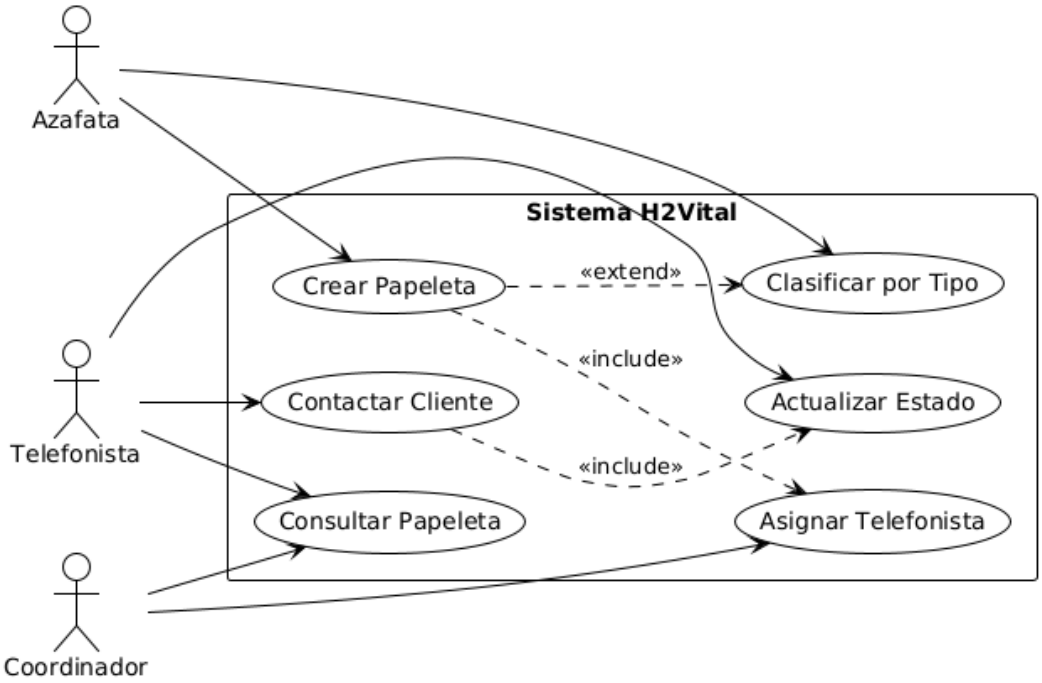


Figura 6.2: Diagrama de casos de uso: Procesamiento de Papeletas

6.3.2. CU-02: Procesamiento de Papeletas

Objetivo: Gestionar el flujo completo de papeletas comerciales desde la captación inicial hasta la conversión en cita programada.

Actores Principales: Azafata, Telefonista, Coordinador

El diagrama de la Figura 6.2 ilustra las interacciones entre los diferentes actores en el procesamiento de papeletas.

Flujo Principal:

1. Azafata crea papeleta con datos básicos del cliente
2. Sistema asigna telefonista automáticamente
3. Telefonista contacta y califica al cliente
4. Coordinador evalúa y programa cita si procede

6.3.3. CU-03: Gestión de Citas

Objetivo: Administrar el proceso completo de citas comerciales desde la programación hasta la ejecución y registro de resultados.

Actores Principales: Coordinador, Comercial, Telefonista

Como se muestra en la Figura 6.3, el proceso de gestión de citas involucra múltiples actores y estados.

Flujo Principal: Programación $\rightarrow$ Asignación $\rightarrow$ Confirmación $\rightarrow$ Ejecución $\rightarrow$ Resultado

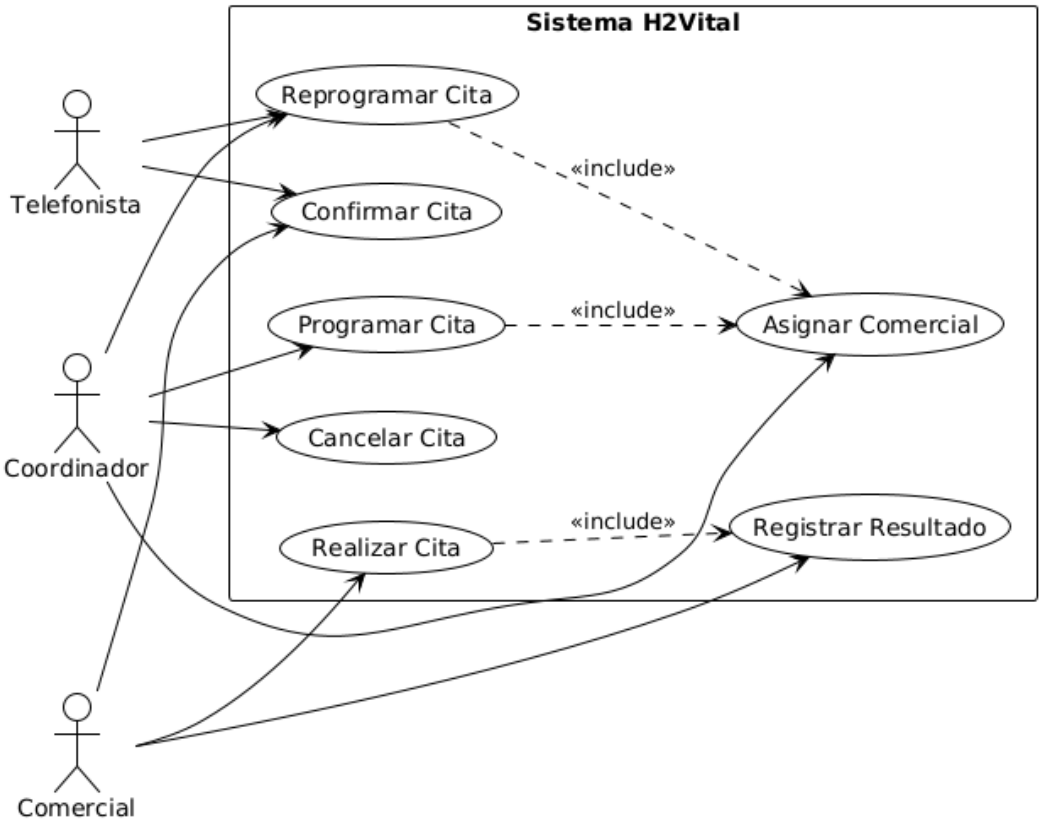


Figura 6.3: Diagrama de casos de uso: Gestión de Citas

6.3.4. CU-04: Administración de Clientes

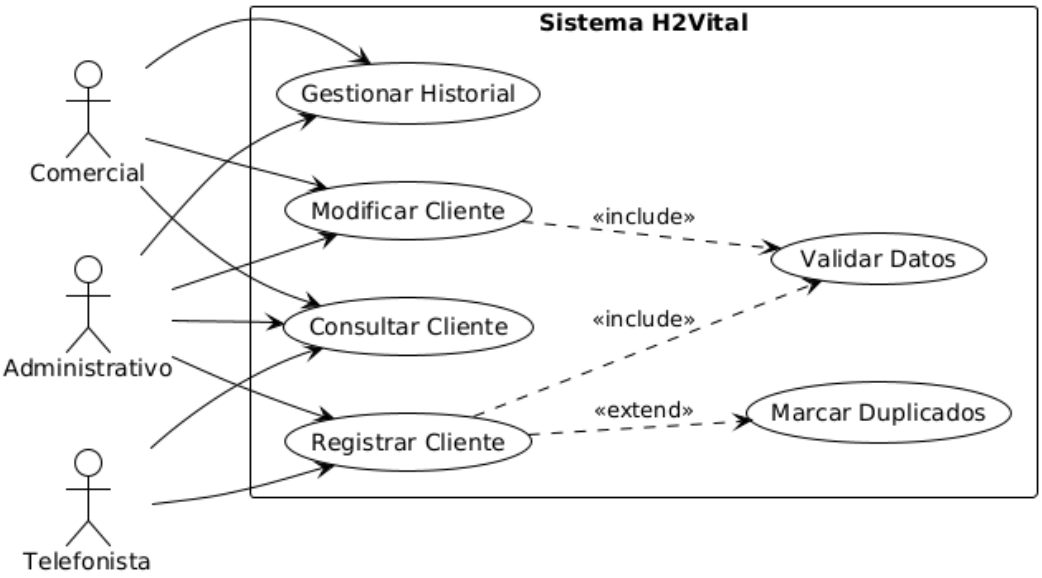


Figura 6.4: Diagrama de casos de uso: Administración de Clientes

Objetivo: Gestionar la información completa de clientes con validación de datos y control de duplicados.

Actores Principales: Administrativo, Comercial, Telefonista

La Figura 6.4 detalla los casos de uso relacionados con la administración integral de clientes.

6.3.5. CU-05: Formalización de Contratos

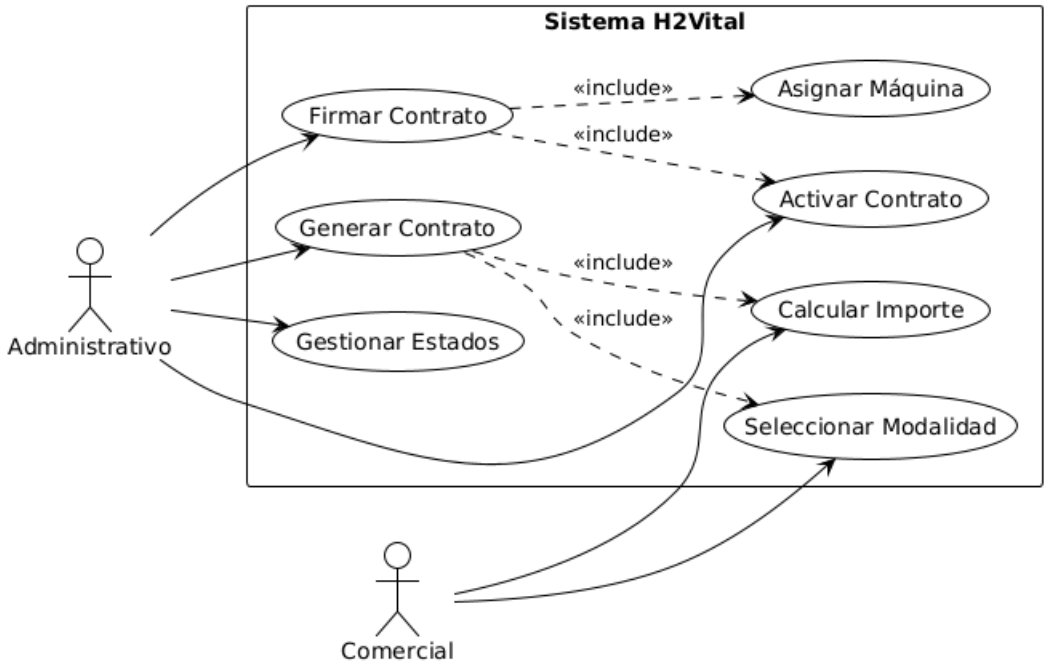


Figura 6.5: Diagrama de casos de uso: Formalización de Contratos

Objetivo: Gestionar el proceso de formalización contractual con cálculo automático de importes y asignación de máquinas.

Actores Principales: Administrativo, Comercial

El diagrama de la Figura 6.5 presenta el flujo completo de formalización de contratos.

6.3.6. CU-06: Gestión de Máquinas

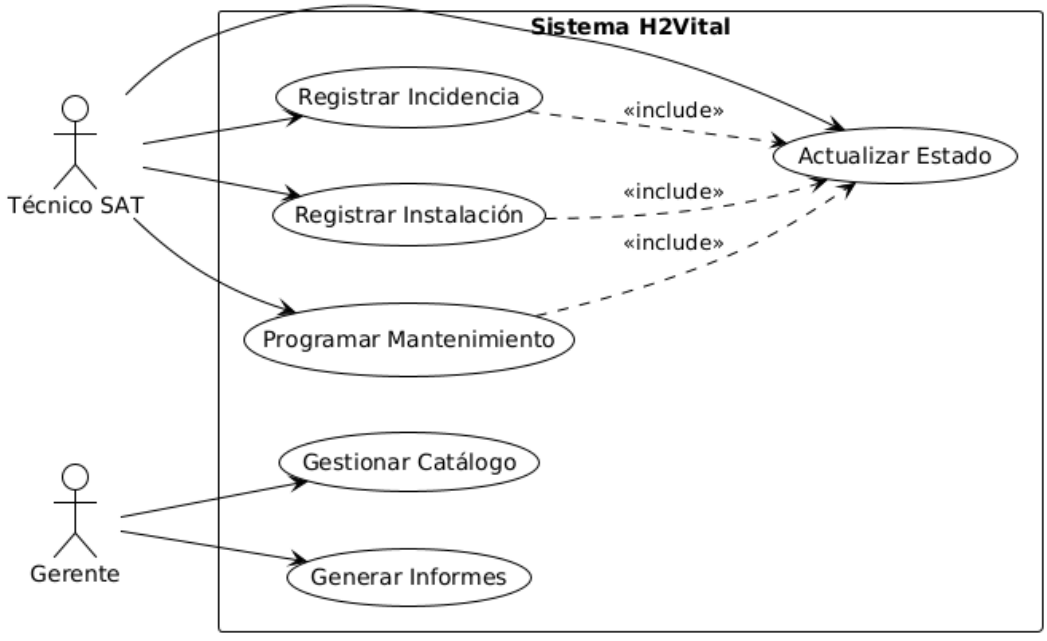


Figura 6.6: Diagrama de casos de uso: Gestión de Máquinas

Objetivo: Administrar el inventario de máquinas y servicios técnicos asociados.
Actores Principales: Técnico SAT, Gerente
La Figura 6.6 muestra los casos de uso específicos para la gestión del inventario de máquinas.

6.4. Modelo de Dominio

El modelo de dominio representa las entidades conceptuales principales del negocio H2Vital y sus relaciones, proporcionando una comprensión clara de la estructura de información que debe gestionar el sistema [46].

La Figura 6.7 presenta el diagrama completo del modelo de dominio con todas las entidades y sus relaciones.

Entidades Principales:

- **Empleado** $\leftrightarrow$ **Usuario:** Relación 1:1 para autenticación y acceso al sistema
- **Empleado** $\rightarrow$ **Papeleta:** Un empleado puede crear/gestionar múltiples papeletas según su rol
- **Cliente** $\rightarrow$ **Papeleta:** Un cliente puede tener múltiples contactos comerciales a lo largo del tiempo
- **Papeleta** $\rightarrow$ **Cita:** Una papeleta exitosa se convierte en una cita programada
- **Cita** $\rightarrow$ **Contrato:** Una cita exitosa puede generar un contrato formalizado
- **Contrato** $\leftrightarrow$ **MaquinaInstalada:** Relación de productos específicos contratados

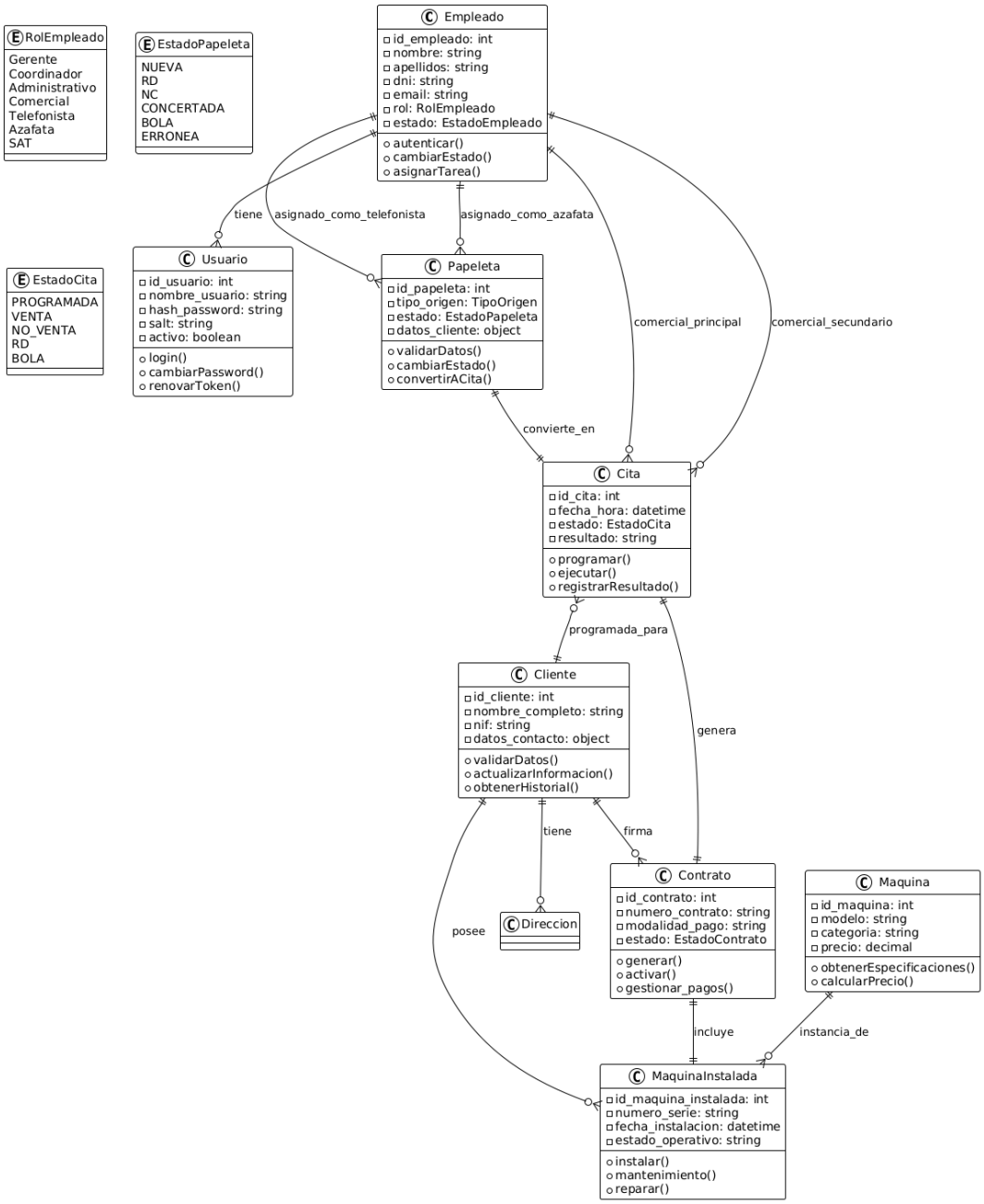


Figura 6.7: Diagrama del modelo de dominio del sistema H2Vital

6.5. Diagramas de Estados

Los diagramas de estados modelan el ciclo de vida de las entidades principales del sistema, mostrando los estados válidos y las transiciones permitidas [47].

6.5.1. Estados de Papeleta

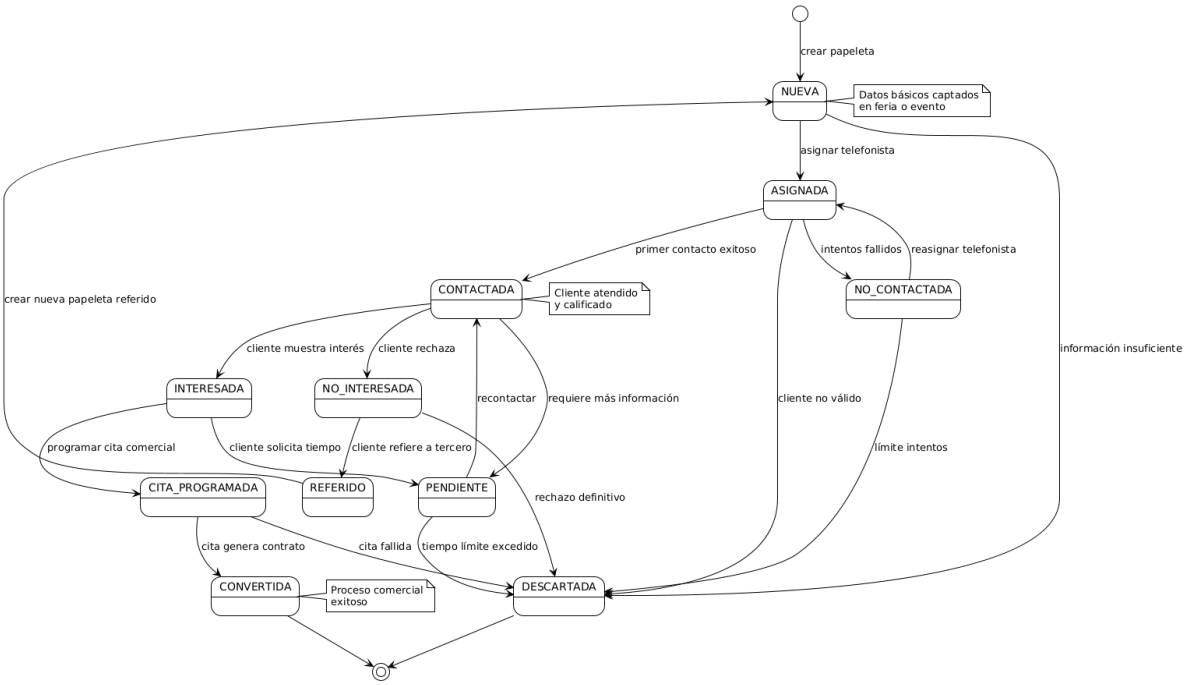


Figura 6.8: Diagrama de estados: Papeleta

La Figura 6.8 ilustra el ciclo de vida completo de las papeletas en el sistema.

Estados Principales:

- **NUEVA:** Papeleta recién creada con datos básicos
- **ASIGNADA:** Telefonista asignado para contacto
- **CONTACTADA:** Cliente contactado y calificado
- **INTERESADA:** Cliente muestra interés comercial
- **CITA_PROGRAMADA:** Cita comercial programada
- **CONVERTIDA:** Proceso exitoso (estado final)
- **DESCARTADA:** Proceso fallido (estado final)

Transiciones Críticas: Las transiciones entre estados están controladas por reglas de negocio específicas que garantizan la integridad del proceso comercial y la trazabilidad completa de cada contacto.

6.5.2. Estados de Cita

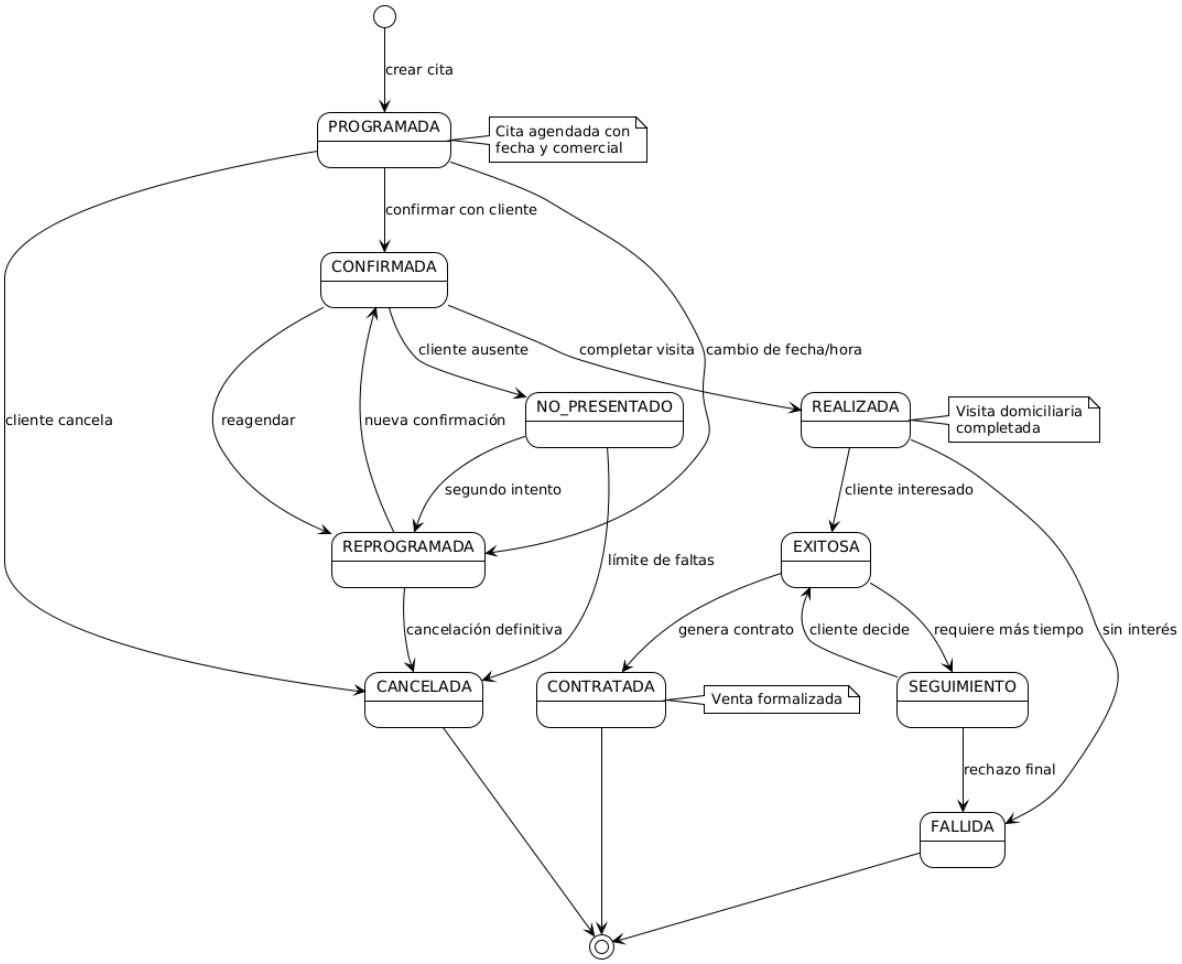


Figura 6.9: Diagrama de estados: Cita

El diagrama de estados de la Figura 6.9 muestra la evolución de las citas comerciales.

Estados Principales:

- **PROGRAMADA:** Cita agendada con fecha y comercial asignado
- **CONFIRMADA:** Cliente confirma disponibilidad
- **REALIZADA:** Visita domiciliaria completada
- **EXITOSA:** Cliente muestra interés en productos
- **CONTRATADA:** Venta formalizada (estado final exitoso)

6.5.3. Estados de Contrato

La Figura 6.10 representa los estados críticos del ciclo de vida de los contratos.

Estados Críticos:

- **ACTIVO:** Servicios operativos y pagos al día

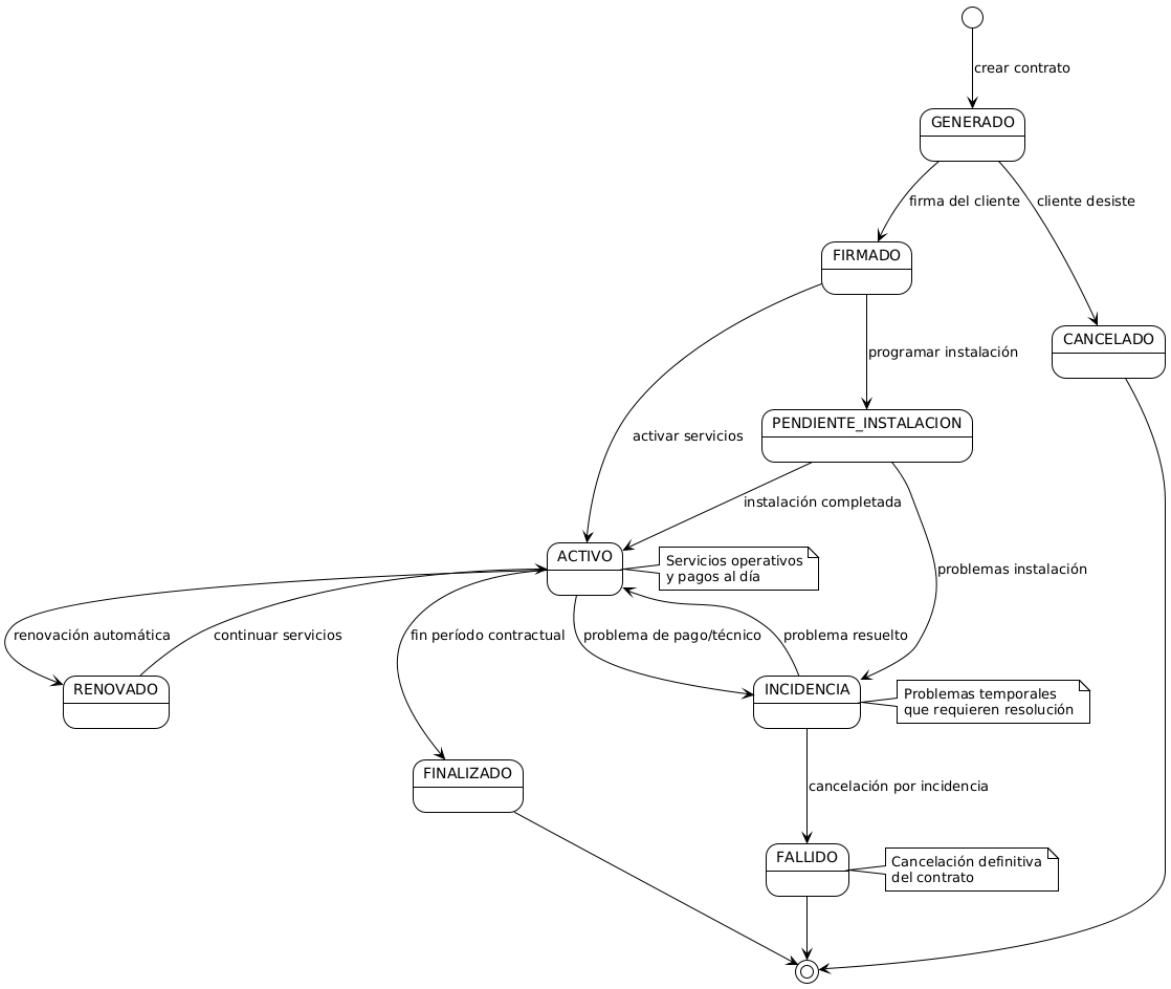


Figura 6.10: Diagrama de estados: Contrato

- **INCIDENCIA:** Problemas temporales que requieren resolución
- **FALLIDO:** Cancelación definitiva del contrato

6.5.4. Estados de Máquina Instalada

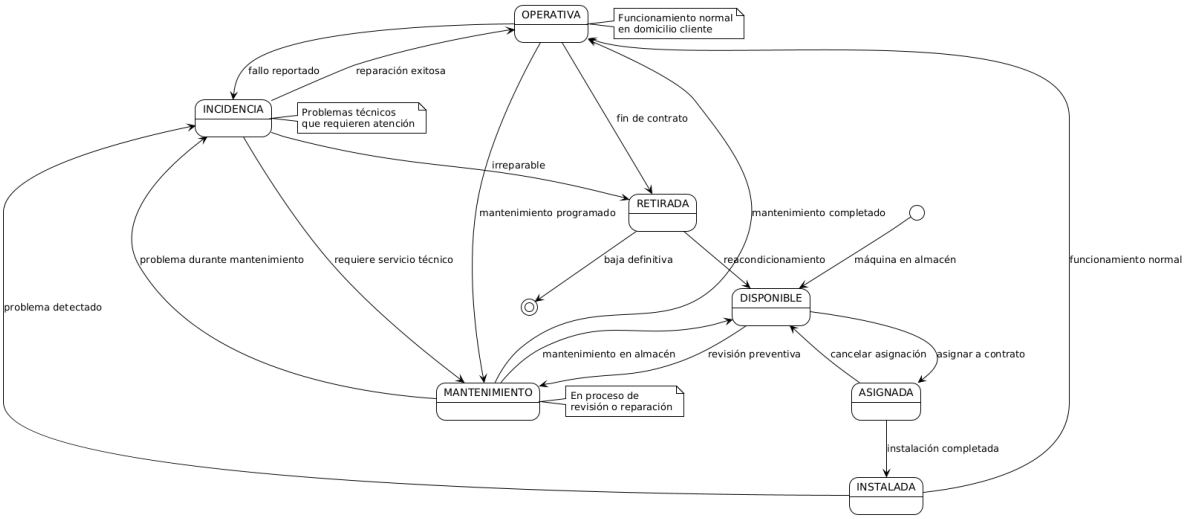


Figura 6.11: Diagrama de estados: Máquina Instalada

Como se observa en la Figura 6.11, el ciclo de vida de las máquinas instaladas sigue un patrón específico.

Ciclo de Vida: DISPONIBLE → ASIGNADA → INSTALADA → OPERATIVA ↔ MANTENIMIENTO

6.6. Diagramas de Secuencia

Los diagramas de secuencia modelan las interacciones temporales entre actores y el sistema para los escenarios más críticos del negocio, proporcionando una vista detallada del comportamiento dinámico del sistema [48].

6.6.1. Crear y Procesar Papeleta

La Figura 6.12 detalla la secuencia completa de interacciones para crear y procesar una papeleta.

Flujo de Interacciones:

1. Azafata inicia creación de papeleta con datos del cliente
2. Sistema valida datos y crea registro en base de datos
3. Sistema asigna telefonista automáticamente según disponibilidad
4. Servicio de notificaciones informa al telefonista asignado
5. Sistema confirma creación exitosa a la azafata

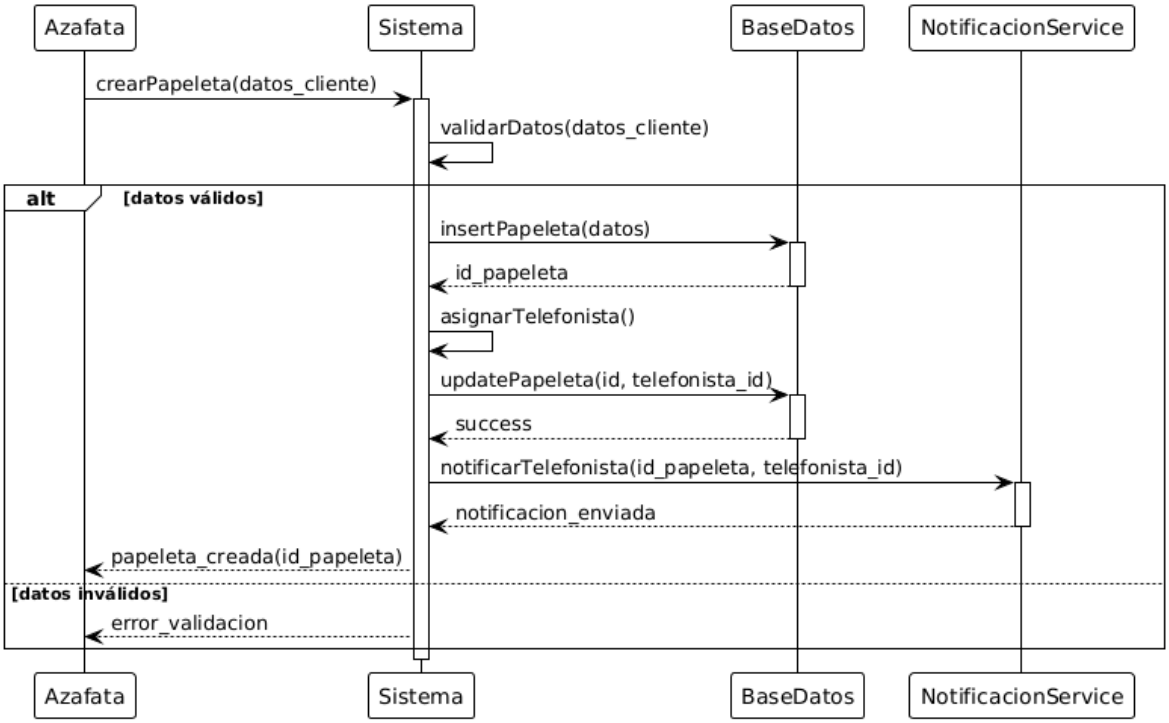


Figura 6.12: Diagrama de secuencia: Crear y Procesar Papeleta

6.6.2. Programar Cita Comercial

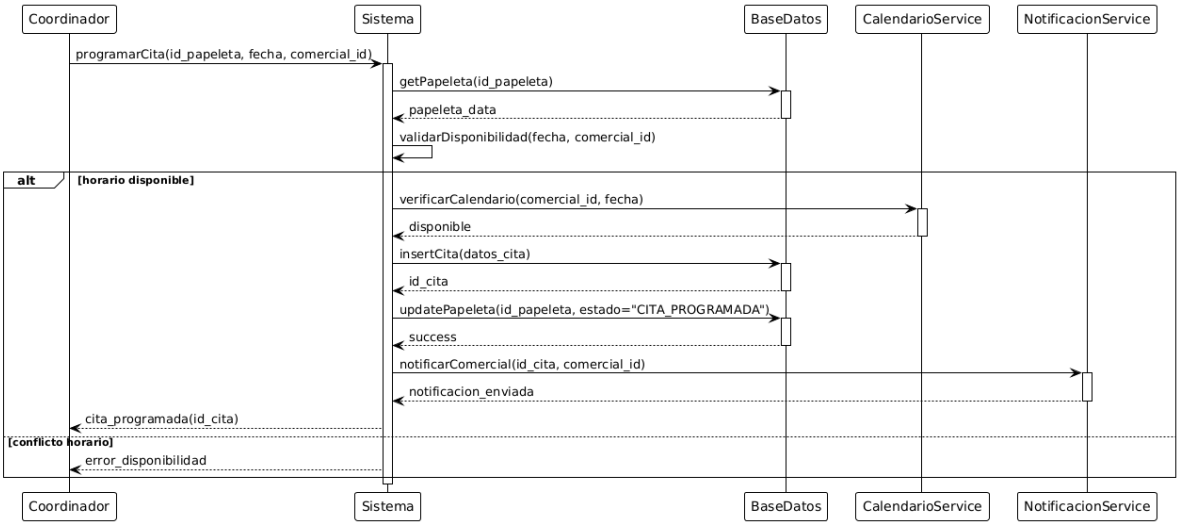


Figura 6.13: Diagrama de secuencia: Programar Cita Comercial

El diagrama de secuencia de la Figura 6.13 muestra el proceso de programación de citas comerciales.

Validaciones Incluidas:

- Verificación de disponibilidad de comercial
- Validación de calendarios y horarios
- Notificación automática a actores involucrados

6.6.3. Realizar Cita y Generar Contrato

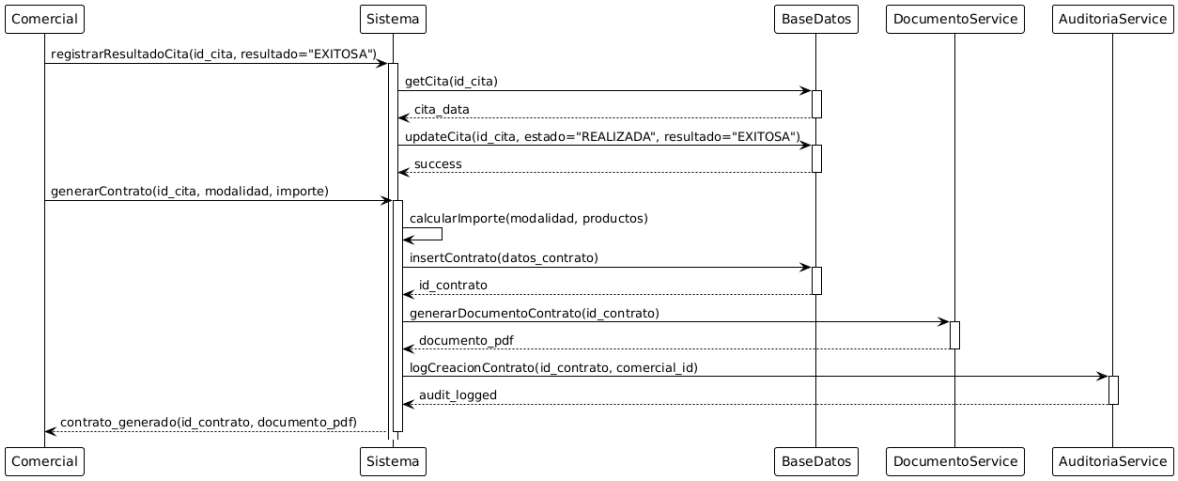


Figura 6.14: Diagrama de secuencia: Realizar Cita y Generar Contrato

La Figura 6.14 ilustra el proceso integral desde la realización de la cita hasta la generación del contrato.

Proceso Integral: Ejecución de cita → Registro de resultado → Generación automática de contrato → Auditoría

6.6.4. Gestión de Mantenimiento SAT

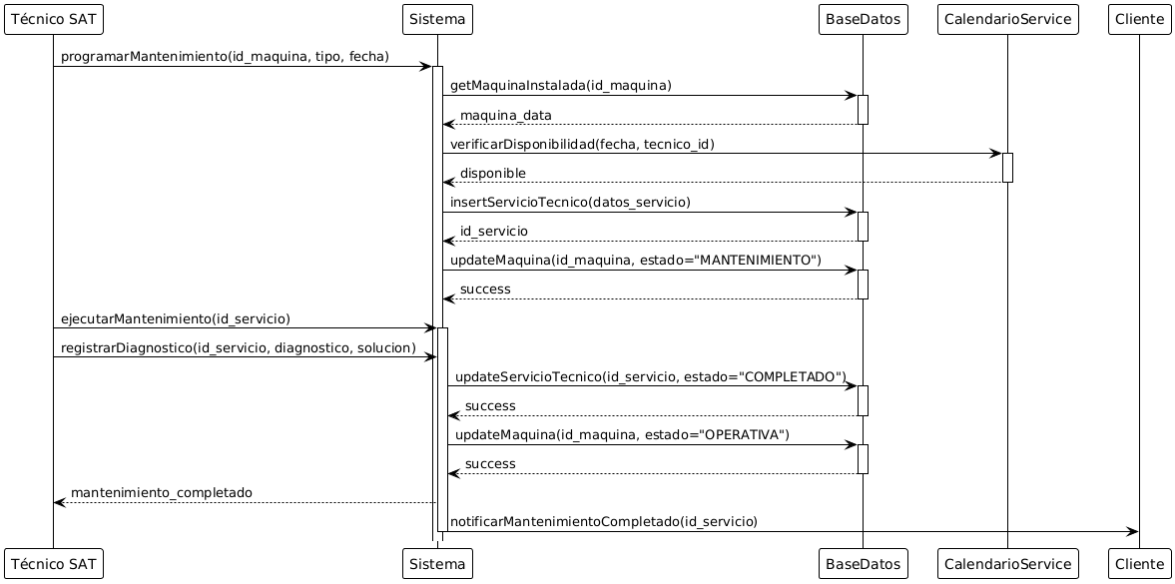


Figura 6.15: Diagrama de secuencia: Gestión de Mantenimiento SAT

Como se muestra en la Figura 6.15, la coordinación de servicios técnicos requiere múltiples interacciones.

Coordinación de Servicios: Programación → Ejecución → Actualización de estados → Notificación al cliente

6.6.5. Gestión de Estados de Contrato

Figura 6.16: Diagrama de secuencia: Gestión de Estados de Contrato

La Figura 6.16 presenta el flujo de gestión de estados de contrato con verificaciones automáticas.

Control de Estados: Verificación de pagos → Evaluación de incidencias → Actualización de estados → Notificaciones automáticas

Este capítulo proporciona el análisis completo del sistema H2Vital desde múltiples perspectivas, estableciendo las bases conceptuales necesarias para la fase de diseño detallado de la solución técnica.

Capítulo 7

Diseño del Sistema

7.1. Introducción al Diseño del Sistema

El diseño del sistema H2Vital traduce los requisitos y análisis conceptual en una arquitectura técnica específica, definiendo la estructura modular, las tecnologías de implementación y los patrones de diseño que garantizan escalabilidad, mantenibilidad y rendimiento óptimo [49].

Esta fase aplica principios consolidados de **arquitectura de software** y **diseño orientado a objetos**, seleccionando tecnologías modernas que proporcionan robustez, seguridad y facilidad de desarrollo. El enfoque metodológico prioriza la separación de responsabilidades, la reutilización de componentes y la implementación de patrones probados en la industria [50].

El diseño del sistema se estructura en cinco componentes principales:

- **Arquitectura Técnica**: Definición de la estructura por capas y componentes principales
- **Diseño de Base de Datos**: Modelo relacional optimizado y estrategias de indexación
- **Diseño de Interfaz de Usuario**: Wireframes, prototipos y principios de experiencia de usuario
- **Patrones de Diseño**: Implementación de patrones arquitectónicos y de programación
- **Consideraciones de Seguridad**: Estrategias de autenticación, autorización y protección de datos

7.2. Arquitectura del Sistema

La arquitectura del sistema H2Vital implementa un patrón de arquitectura en capas (N-tier) que separa claramente las responsabilidades de presentación, lógica de negocio y acceso a datos, proporcionando flexibilidad para el mantenimiento y evolución futura del sistema [51].

7.2.1. Arquitectura por Capas

Descripción de la Arquitectura:

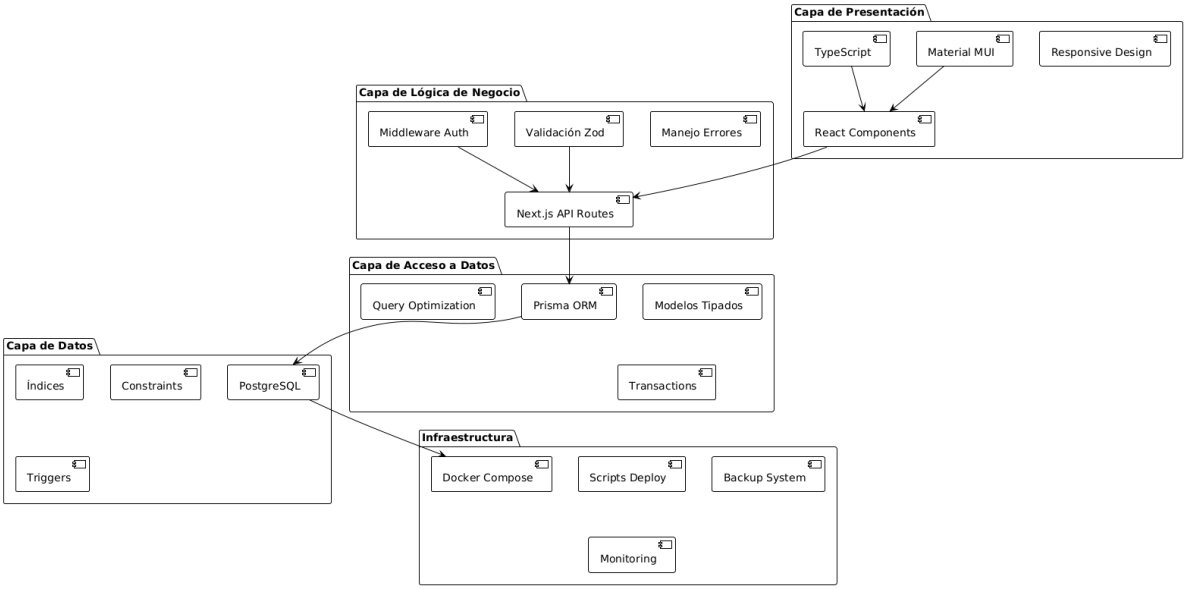


Figura 7.1: Arquitectura en capas del sistema H2Vital

La Figura 7.1 presenta la estructura en capas del sistema H2Vital.

El sistema implementa una arquitectura de 5 capas que garantiza la separación de responsabilidades y facilita el mantenimiento y escalabilidad:

1. **Capa de Presentación**: Interfaz de usuario desarrollada con React Components, Material MUI y TypeScript
2. **Capa de Lógica de Negocio**: API Routes de Next.js con middleware y validaciones
3. **Capa de Acceso a Datos**: Prisma ORM con modelos tipados para interacción segura con la base de datos
4. **Capa de Base de Datos**: PostgreSQL con índices optimizados para rendimiento
5. **Capa de Infraestructura**: Docker Compose con scripts automatizados para despliegue

7.2.2. Arquitectura de Componentes

La Figura 7.2 muestra la interacción entre los componentes principales del sistema.

Componentes Principales:

- **Frontend Application**: Aplicación Next.js con App Router para navegación moderna
- **Authentication Service**: Gestión de sesiones con JWT y bcrypt
- **Business Logic Layer**: Servicios de negocio para cada módulo funcional
- **Data Access Layer**: Prisma ORM con connection pooling
- **Database**: PostgreSQL con backup automatizado

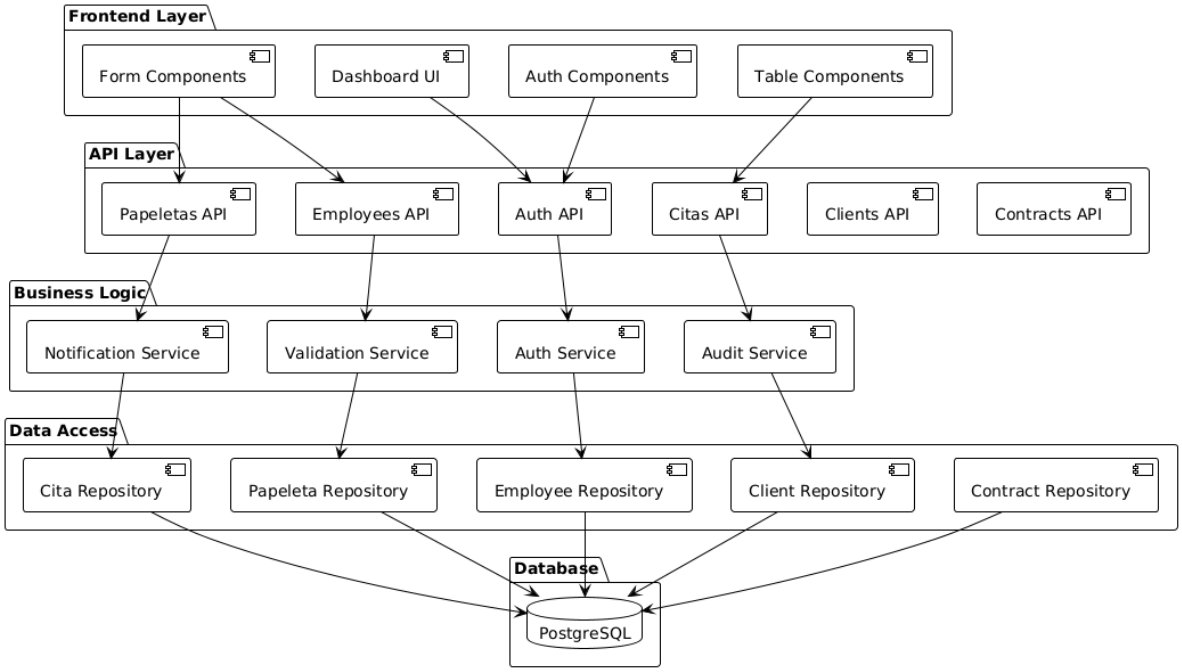


Figura 7.2: Diagrama de componentes del sistema H2Vital

7.2.3. Arquitectura Modular por Dominio

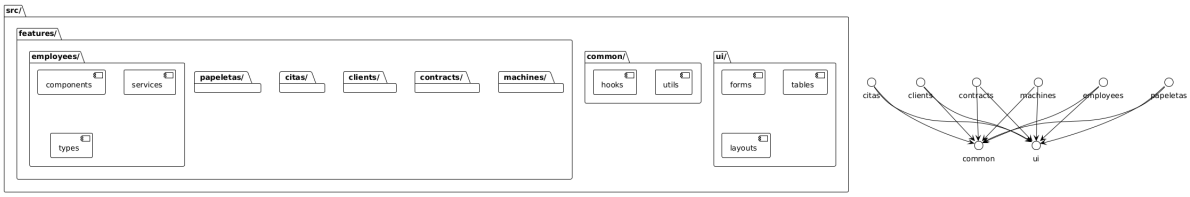


Figura 7.3: Estructura modular del sistema por dominios

Como se ilustra en la Figura 7.3, la organización modular del sistema facilita el mantenimiento y evolución.

Organización por Módulo de Dominio:

La arquitectura modular organiza el código por dominios de negocio, facilitando el mantenimiento, testing y evolución independiente de cada módulo funcional [46]:

- **Employees Module**: Gestión completa de empleados y roles del sistema
- **Papeletas Module**: Procesamiento del ciclo completo de contactos comerciales desde captación hasta conversión
- **Citas Module**: Gestión de citas comerciales, asignación de comerciales y seguimiento de resultados
- **Clientes Module**: Administración integral de información de clientes y contactos
- **Contratos Module**: Formalización y gestión de contratos con vinculación a máquinas específicas
- **Máquinas Module**: Control de inventario y servicios técnicos especializados

7.3. Diseño de Base de Datos

El diseño de la base de datos implementa un modelo relacional normalizado que garantiza la integridad referencial, optimiza el rendimiento de consultas y facilita el mantenimiento de la información [52].

7.3.1. Modelo Entidad-Relación

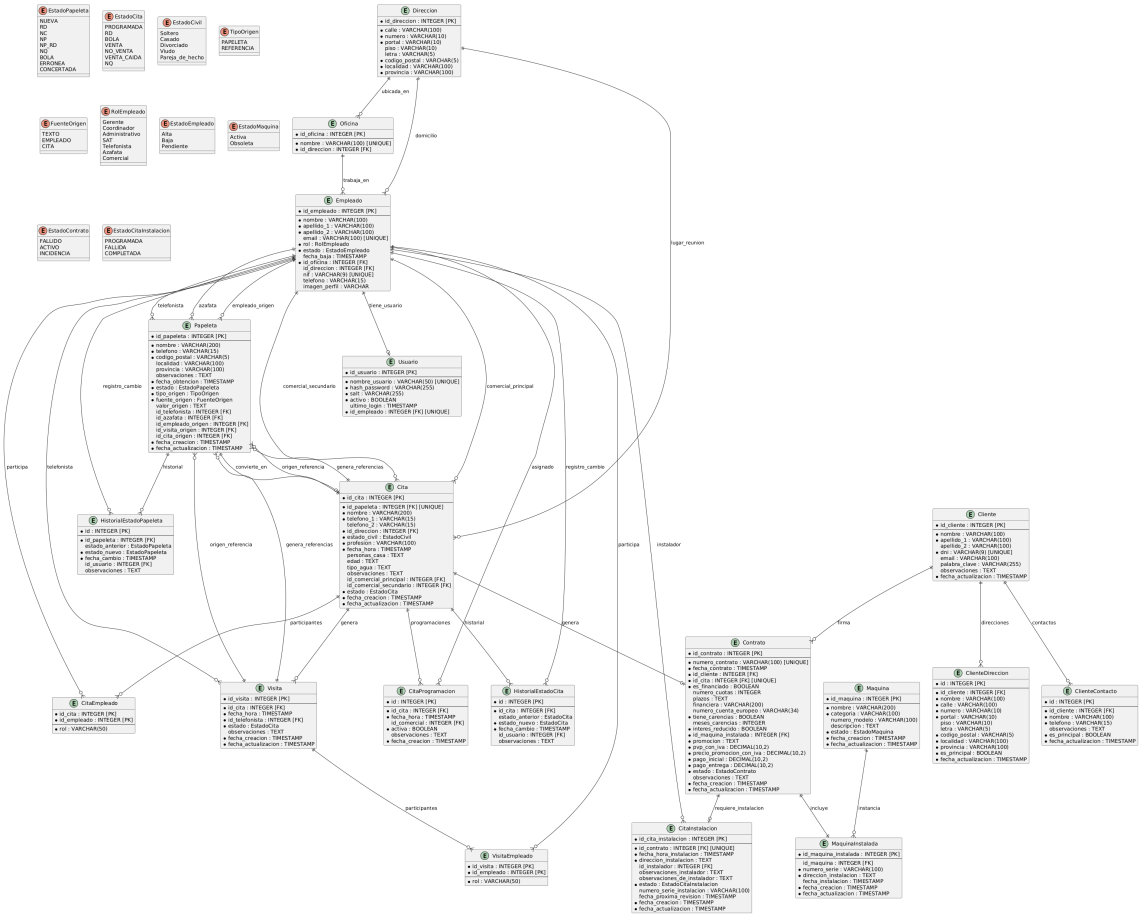


Figura 7.4: Diagrama Entidad-Relación del sistema H2Vital

La Figura 7.4 presenta el diagrama entidad-relación completo del sistema H2Vital.

Entidades Principales:

- **Usuario**: Credenciales de acceso con hashing bcrypt
- **Empleado**: Información personal y laboral con roles múltiples
- **Papeleta**: Contactos comerciales con control de estados
- **Cita**: Programación de citas con seguimiento de resultados
- **Cliente**: Información completa con historial de interacciones
- **Contrato**: Formalización contractual con estados y modalidades

- **Máquina**: Catálogo de productos con especificaciones técnicas
- **MáquinaInstalada**: Instancias específicas instaladas en clientes

7.3.2. Estrategia de Indexación

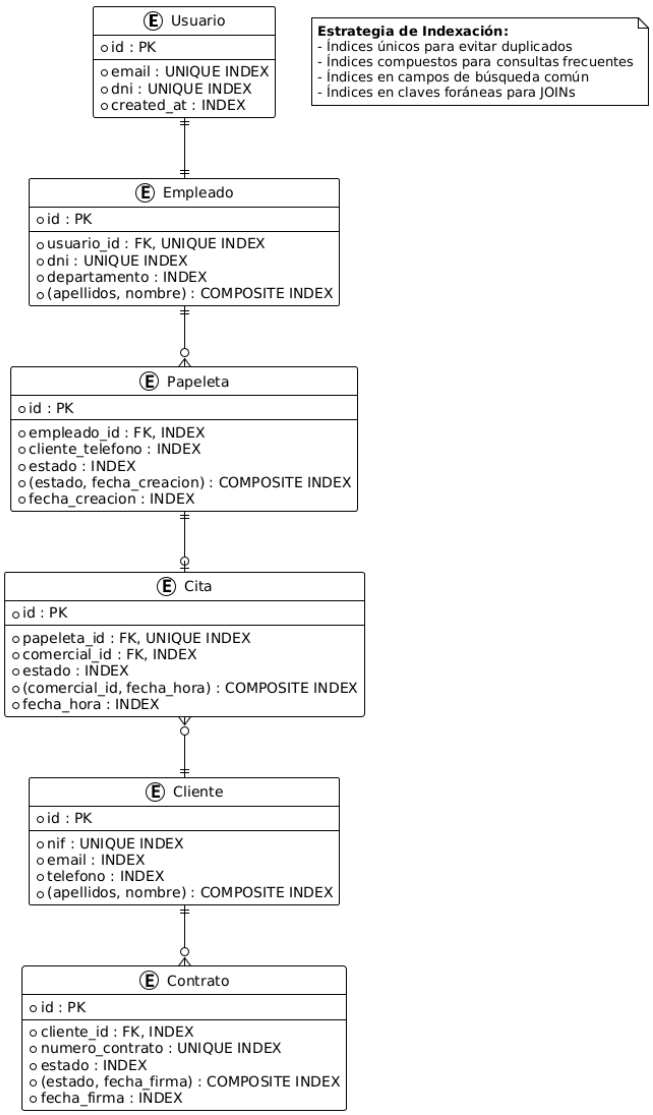


Figura 7.5: Estrategia de indexación para optimización de consultas

La Figura 7.5 detalla la estrategia de indexación implementada para optimizar las consultas.

Índices Estratégicos:

- **Índices Únicos**: DNI, email, número de contrato para garantizar unicidad
- **Índices Compuestos**: (empleado_id, fecha) para búsquedas temporales
- **Índices de Estado**: Estados de papeletas, citas y contratos para filtros frecuentes
- **Índices de Búsqueda**: Nombres y apellidos con soporte para búsquedas parciales

7.4. Diseño de Interfaz de Usuario

El diseño de la interfaz de usuario se fundamenta en principios de **experiencia de usuario** (UX) y **Material Design**, garantizando usabilidad, accesibilidad y consistencia visual en todas las pantallas del sistema [53].

7.4.1. Sistema de Componentes Reutilizables

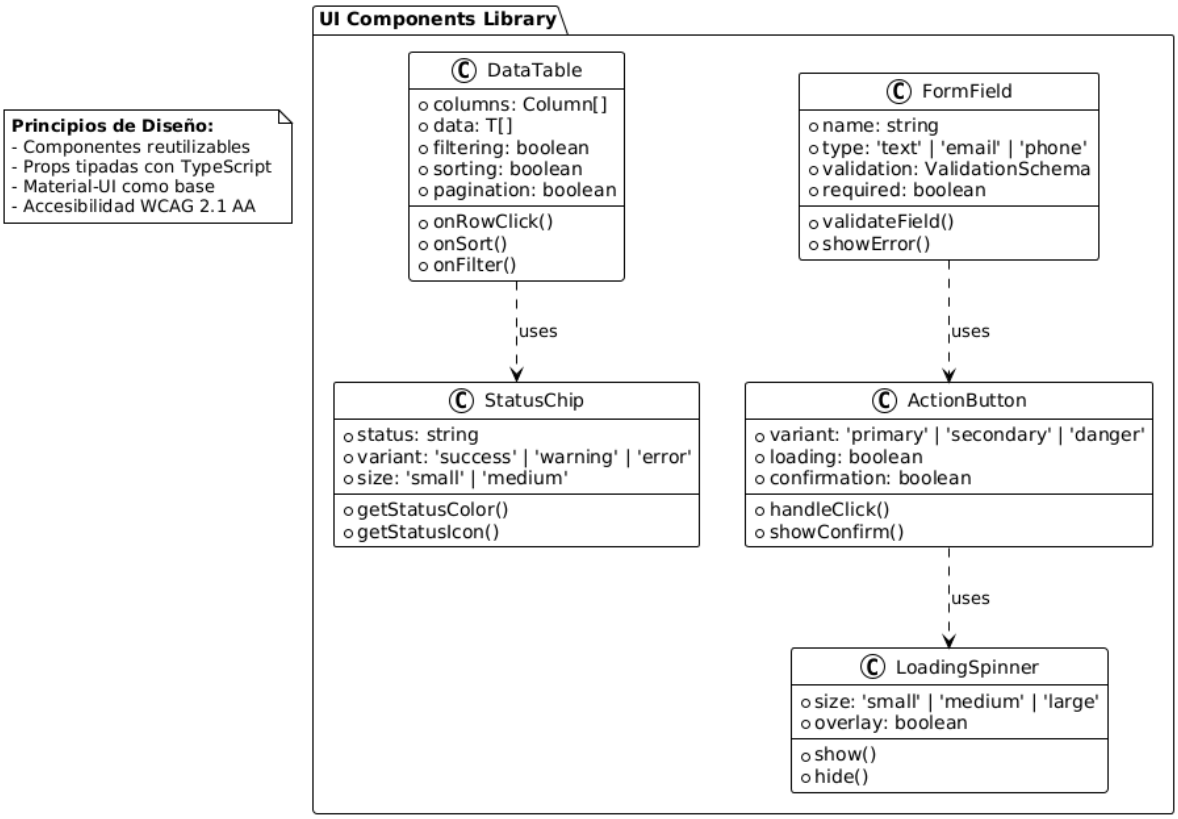


Figura 7.6: Sistema de componentes UI reutilizables

La Figura 7.6 muestra la arquitectura del sistema de componentes UI desarrollado.

Componentes Desarrollados:

- **DataTable**: Tabla avanzada con filtros, ordenación y paginación
- **StatusChip**: Indicadores visuales de estados con códigos de color
- **FormField**: Campos de formulario tipados con validación integrada
- **ActionButton**: Botones de acción con confirmación y loading states

7.5. Patrones de Diseño Aplicados

La implementación del sistema H2Vital utiliza patrones de diseño consolidados que mejoran la mantenibilidad, extensibilidad y robustez del código [50].

7.5.1. Patrones Arquitectónicos

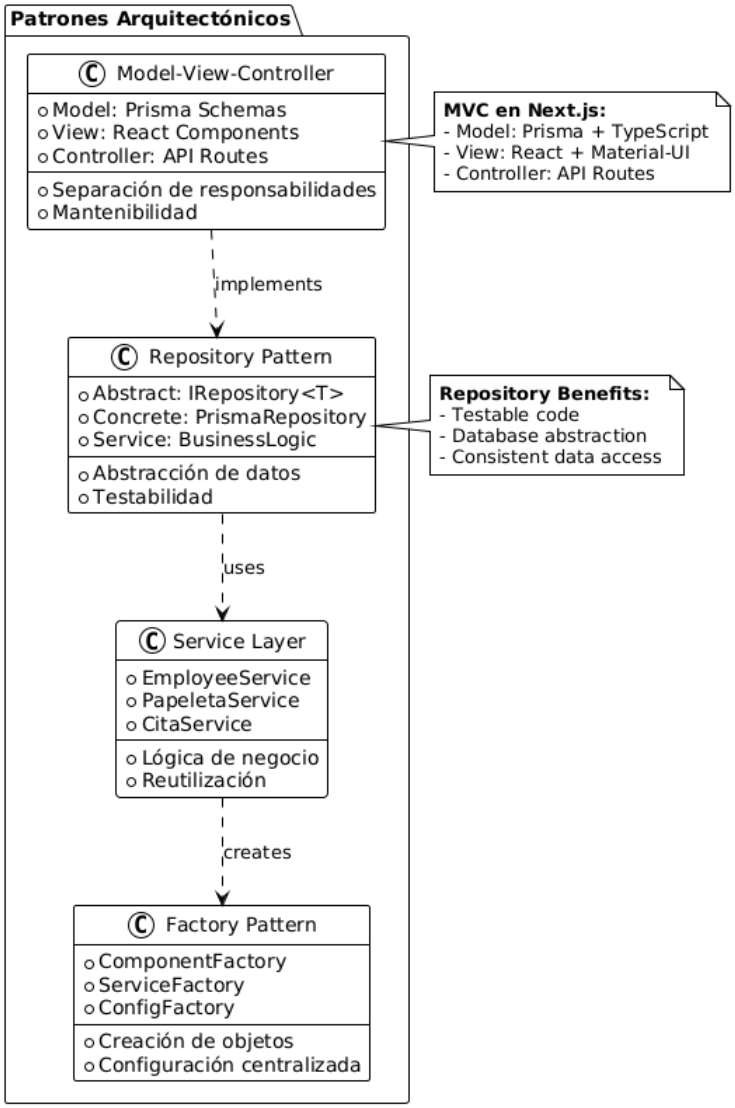


Figura 7.7: Patrones arquitectónicos implementados

La Figura 7.7 ilustra los principales patrones arquitectónicos implementados en el sistema.

Patrones Implementados:

- **Model-View-Controller (MVC)**: Separación de responsabilidades entre presentación, lógica y datos
- **Repository Pattern**: Abstracción del acceso a datos con interfaces tipadas
- **Service Layer**: Encapsulación de lógica de negocio en servicios especializados
- **Factory Pattern**: Creación de objetos complejos con configuración específica

7.5.2. Patrones de Frontend

Como se muestra en la Figura 7.8, se han implementado varios patrones específicos de React.

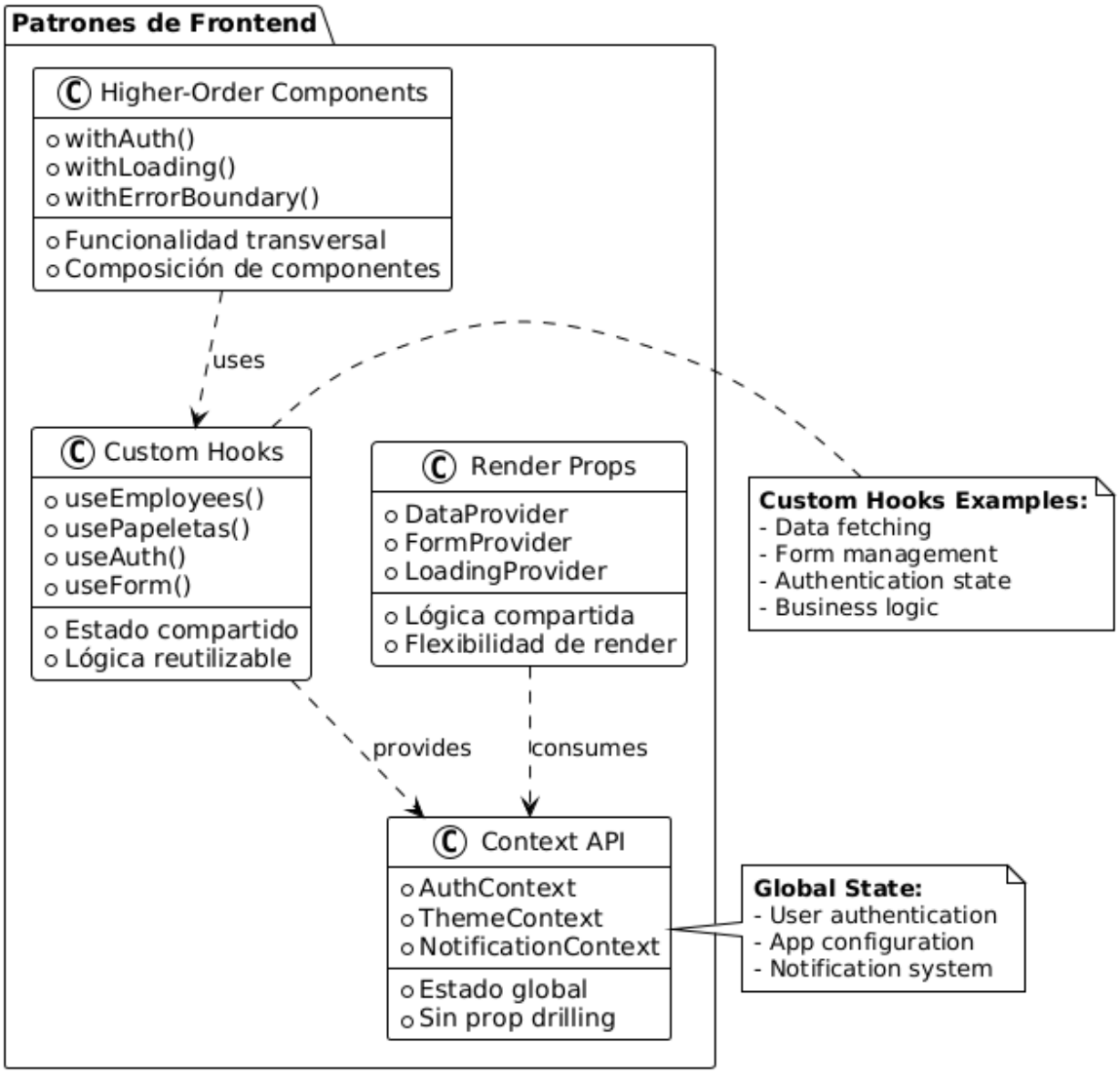


Figura 7.8: Patrones de diseño en la capa de presentación

Patrones de React Aplicados:

- **Custom Hooks**: Lógica reutilizable para gestión de estado y efectos
- **Higher-Order Components**: Componentes de orden superior para funcionalidades transversales
- **Render Props**: Patrón para compartir lógica entre componentes
- **Context API**: Gestión de estado global sin prop drilling

7.6. Consideraciones de Seguridad

La seguridad del sistema H2Vital implementa múltiples capas de protección que abarcan desde la autenticación de usuarios hasta la protección de datos en tránsito y reposo [54].

7.6.1. Arquitectura de Seguridad

La Figura 7.9 presenta la arquitectura de seguridad multicapa implementada.
Capas de Seguridad:

1. **Capa de Red**: HTTPS con TLS 1.3 y headers de seguridad
2. **Capa de Aplicación**: Autenticación JWT y autorización RBAC
3. **Capa de Datos**: Cifrado de datos sensibles y backup seguro
4. **Capa de Auditoría**: Logging completo de accesos y modificaciones

7.6.2. Flujo de Autenticación y Autorización

El diagrama de la Figura 7.10 detalla el proceso completo de autenticación y autorización.
Proceso de Seguridad:

1. Usuario envía credenciales través del formulario de login
2. Frontend valida formato antes de envío al servidor
3. API busca usuario en base de datos utilizando Prisma ORM
4. Sistema verifica password utilizando bcrypt para seguridad
5. Si las credenciales son válidas, genera JWT con información de usuario y permisos
6. Frontend almacena token de forma segura y redirige a dashboard autorizado

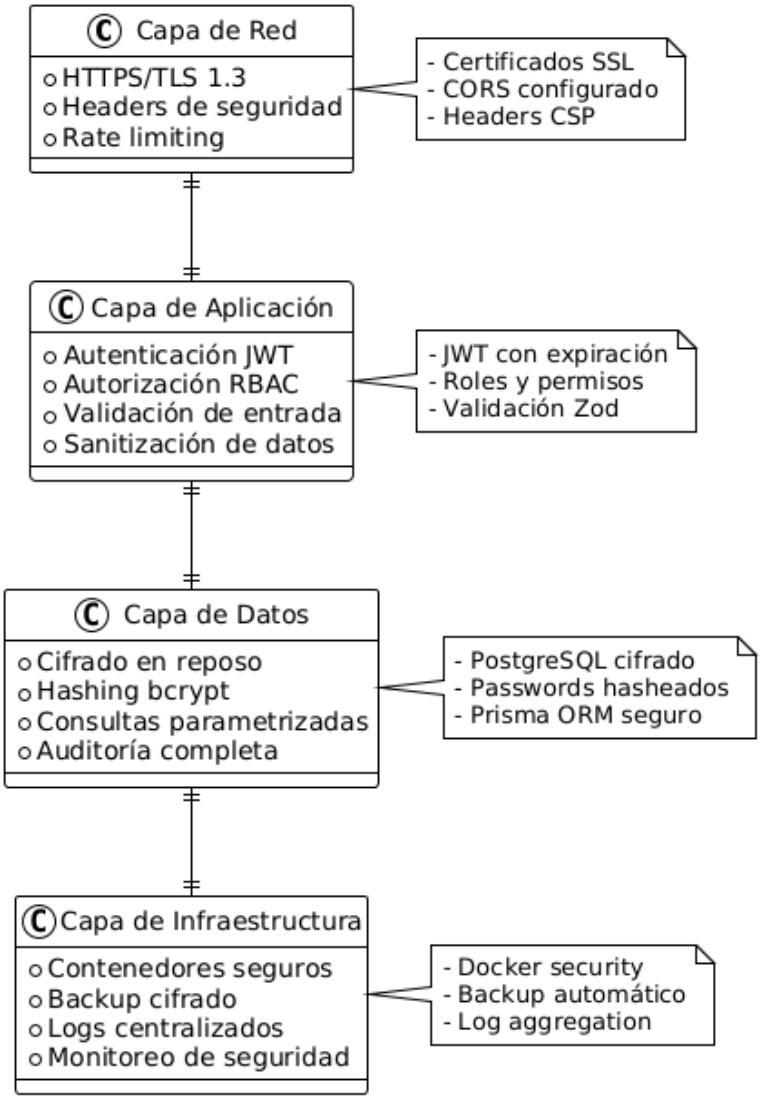


Figura 7.9: Arquitectura de seguridad multicapa

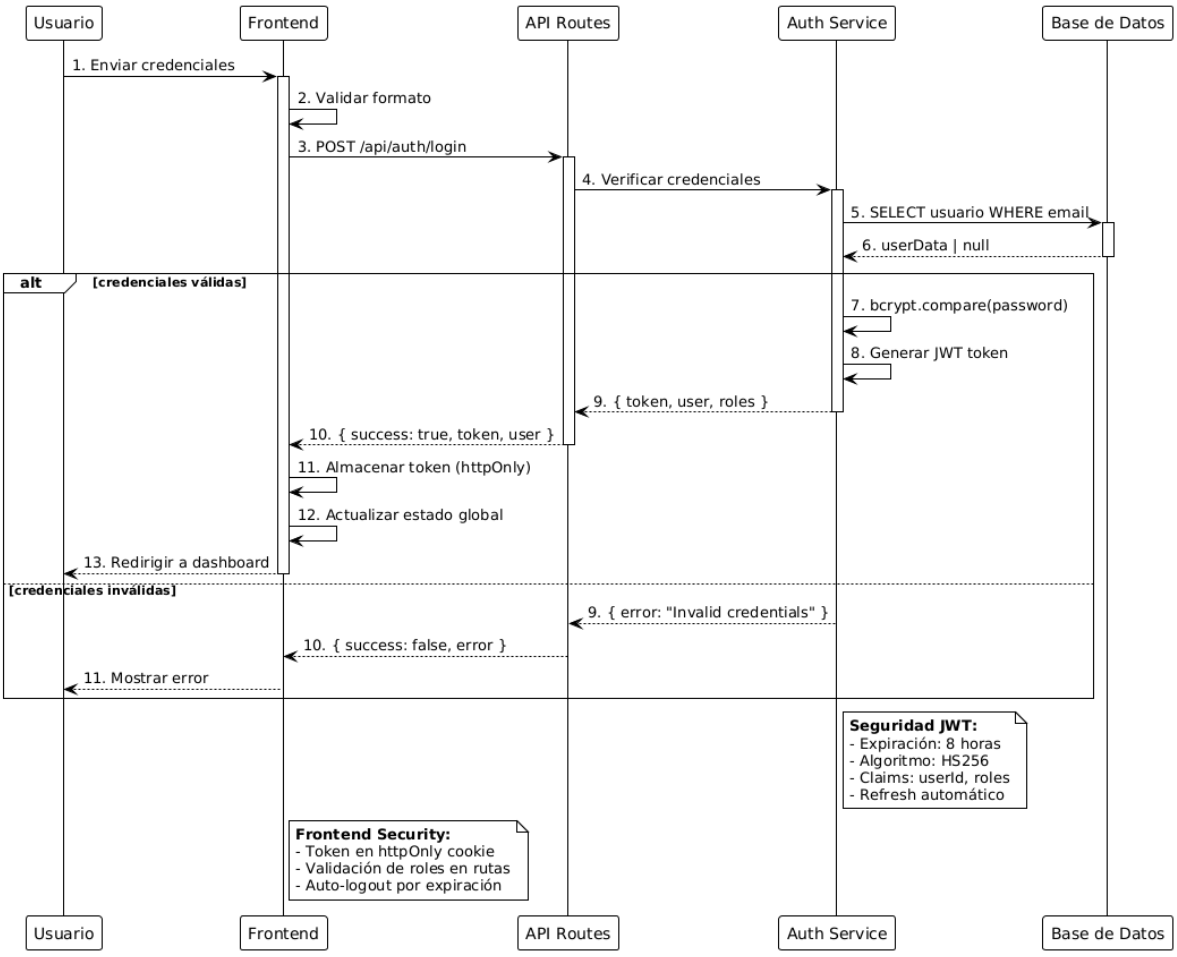


Figura 7.10: Diagrama de flujo de autenticación y autorización

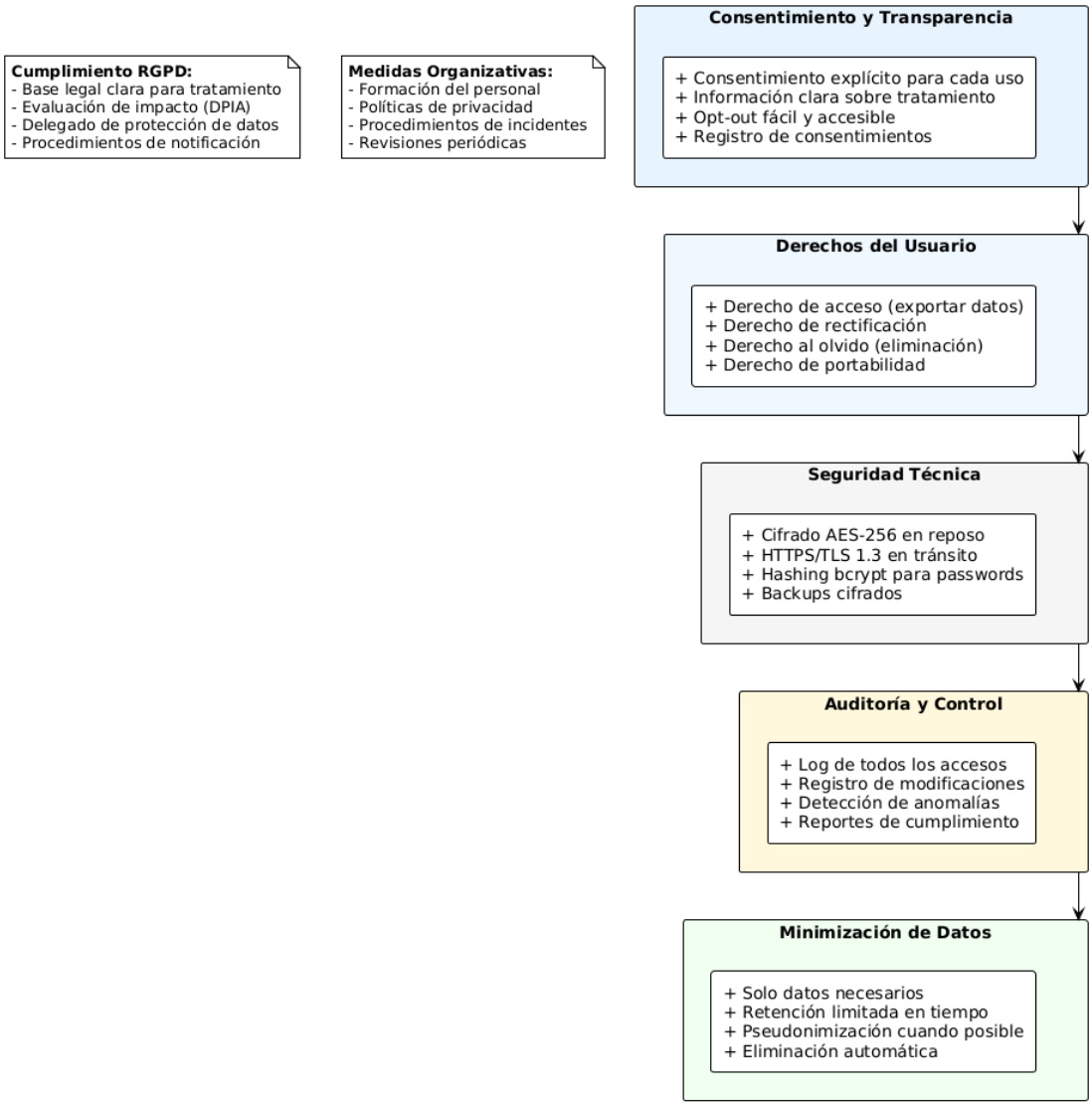


Figura 7.11: Estrategia de protección de datos personales

7.6.3. Protección de Datos Personales

La Figura 7.11 muestra la estrategia integral de protección de datos personales.

Medidas de Protección RGPD:

- **Minimización de Datos**: Recopilación solo de datos necesarios para la función
- **Consentimiento Explícito**: Registro de consentimiento para cada tratamiento
- **Derecho al Olvido**: Eliminación completa de datos bajo solicitud
- **Portabilidad**: Exportación de datos en formato JSON estructurado
- **Auditoría**: Trazabilidad completa de accesos y modificaciones

7.7. Optimización y Rendimiento

El diseño del sistema incorpora estrategias de optimización que garantizan tiempos de respuesta aceptables y escalabilidad para el crecimiento futuro de H2Vital.

7.7.1. Estrategias de Optimización

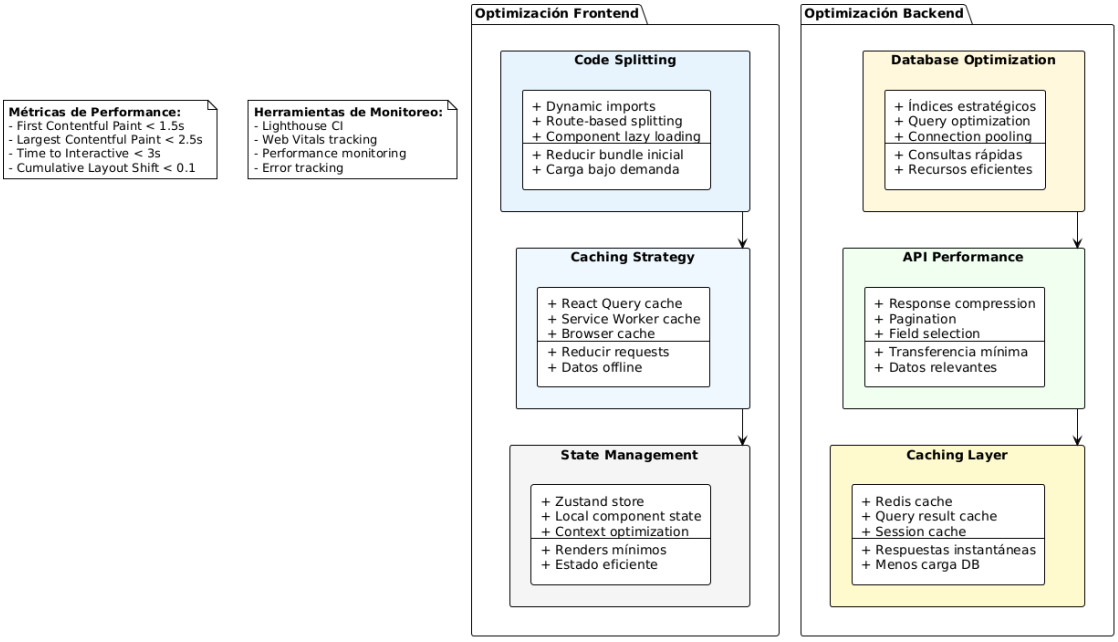


Figura 7.12: Estrategias de optimización implementadas

Como se ilustra en la Figura 7.12, se han implementado múltiples estrategias de optimización.

Optimizaciones Aplicadas:

- **Lazy Loading**: Carga diferida de componentes y datos
- **Paginación**: Limitación de resultados para mejorar tiempos de carga

- **Índices de Base de Datos**: Optimización de consultas frecuentes
- **Connection Pooling**: Gestión eficiente de conexiones a base de datos
- **Compresión**: Gzip para reducir tamaño de transferencias

Este capítulo establece el diseño técnico completo del sistema H2Vital, proporcionando la base arquitectónica necesaria para la implementación exitosa de todos los requisitos identificados en las fases anteriores del proyecto.

Capítulo 8

Implementación

8.1. Introducción

La fase de implementación del sistema H2Vital representa la materialización de la arquitectura y diseño definidos en los capítulos anteriores, aplicando principios modernos de ingeniería del software y metodologías ágiles para crear una solución robusta y escalable [30]. Este capítulo documenta los aspectos técnicos más relevantes del desarrollo, incluyendo la implementación de componentes UI comunes, el sistema de despliegue automatizado con comandos de mantenimiento, las comunicaciones en tiempo real mediante SSE y las optimizaciones de rendimiento aplicadas.

La implementación se llevó a cabo siguiendo una metodología iterativa basada en feedback continuo con H2Vital, lo que permitió ajustar y refinar la solución técnica según las necesidades reales del negocio [34]. El resultado es una aplicación con diversas funcionalidades, que digitaliza completamente los procesos administrativos de la empresa [35].

8.2. Configuración del Entorno de Desarrollo

8.2.1. Arquitectura Tecnológica Next.js 15

La configuración del entorno se fundamenta en Next.js 15 con App Router, aprovechando las capacidades más avanzadas del framework para crear una aplicación full-stack moderna [7]. La configuración optimizada incluye headers de seguridad, output standalone para despliegue en contenedores y configuraciones específicas para rendimiento, implementando headers de seguridad y optimizaciones específicas.

La implementación utiliza el patrón de configuración por entornos diferenciados, con archivos específicos para desarrollo (.env.development) y producción (.env.production), garantizando la separación clara entre configuraciones y el cumplimiento de mejores prácticas de seguridad [55].

8.2.2. Configuración de Prisma ORM

El sistema de base de datos se implementó utilizando Prisma como ORM, proporcionando type safety completo y migraciones automáticas [13]. La configuración incluye un esquema con 11 modelos principales, 8 enumeraciones y varios índices estratégicos optimizados para las consultas más frecuentes del sistema.

El patrón Singleton implementado para el cliente Prisma previene el agotamiento de conexiones durante el desarrollo con hot-reloading, siguiendo las mejores prácticas recomendadas para aplicaciones Next.js [56]:

```
const globalForPrisma = global as unknown as { prisma: PrismaClient }
const prisma = globalForPrisma.prisma || new PrismaClient({
  log: process.env.NODE_ENV === 'development' ?
    ['query', 'error', 'warn'] : ['error'],
})
if (process.env.NODE_ENV !== 'production') {
  globalForPrisma.prisma = prisma
}
```

Figura 8.1: Implementación del patrón Singleton para Prisma

8.2.3. Containerización con Docker Compose

La aplicación se containeriza completamente utilizando Docker Compose con configuraciones diferenciadas para desarrollo y producción. El entorno de desarrollo incluye servicios adicionales como PgAdmin (puerto 5050) y Prisma Studio (puerto 5555), mientras que producción se optimiza para seguridad y rendimiento con certificados SSL y configuraciones específicas.

8.3. Implementación de Componentes UI Reutilizables

8.3.1. Arquitectura de Componentes Basada en Atomic Design

La implementación sigue una arquitectura de componentes estructurada en cinco niveles jerárquicos basada en Atomic Design: átomos (elementos básicos como botones e inputs), moléculas (combinaciones de átomos como formularios simples), organismos (secciones complejas como headers o listas), templates (estructuras de página) y páginas (instancias específicas con contenido real), aplicando principios de reutilización y mantenibilidad [57]. Esta arquitectura garantiza un alto nivel de reutilización, donde componentes como GenericEntityList se utilizan en varios módulos y DetailPageLayout tiene múltiples implementaciones específicas.

8.3.2. Vistas de Detalle y Layout Común

DetailPageLayout - Estructura Unificada

La implementación de vistas de detalle se basa en el componente DetailPageLayout, que proporciona una estructura unificada para todas las páginas de detalle del sistema. Este layout implementa un patrón

de dos columnas con sidebar izquierdo (4/12) para información resumen y contenido principal (8/12) con sistema de tabs dinámicos:

```
interface DetailPageLayoutProps {
  title: string
  backPath: string
  actions?: ReactNode
  sidebarContent: ReactNode
  tabs: TabConfig[]
  activeTab?: string
  connectionStatus?: ConnectionStatus
}
const DetailPageLayout = ({ tabs, activeTab, ...props }) => {
  const [currentTab, setCurrentTab] = useState(
    activeTab || localStorage.getItem('lastTab') || tabs[0]?.key
  )

  return (
    <Grid container spacing={6}>
      <Grid item xs={12} md={4}>
        {sidebarContent}
      </Grid>
      <Grid item xs={12} md={8}>
        <TabContext value={currentTab}>
          <CustomTabList>
            {tabs.map(tab => (
              <Tab key={tab.key} value={tab.key} label={tab.label} />
            ))}
          </CustomTabList>
          {tabs.map(tab => (
            <TabPanel key={tab.key} value={tab.key}>
              {tab.content}
            </TabPanel>
          ))}
        </TabContext>
      </Grid>
    </Grid>
  )
}
```

Figura 8.2: Estructura del DetailPageLayout con sistema de tabs persistentes

Sistema de Tabs Contextual

El sistema de tabs implementa persistencia tanto en localStorage como en parámetros URL, con fallback automático al primer tab disponible. La configuración utiliza una interfaz TabConfig que define key, label, icon y content, permitiendo tabs dinámicos y condicionales según los permisos del usuario. Las implementaciones específicas incluyen EmployeeDetailView (con OverviewTab y CredentialsTab), ClientDetailView (con AddressesTab y ContactsTab) y CitaDetailView con funcionalidades especiales para creación de ventas.

8.3.3. Sistema de Avatares e Imágenes de Perfil

UserAvatar - Componente Base

La implementación del sistema de avatares se centra en el componente UserAvatar, que proporciona un sistema de fallback inteligente: imagen personalizada → iniciales con colores semánticos → avatar por defecto. El componente incluye detección automática de avatares por defecto y sistema de colores específicos por rol de empleado:

```tsx
interface UserAvatarProps {
  fullName: string
  imagePath?: string
  isProfileImage?: boolean
  skin?: 'filled' | 'light' | 'light-static'
  color?: ThemeColor
  size?: number
  role?: RolEmpleado
}

const UserAvatar = ({ fullName, imagePath, role, ...props }) => {
  const roleConfig = getEmployeeRoleConfig(role)
  const initials = getInitials(fullName)
  const isDefaultAvatar = imagePath?.includes('/images/avatars/')

  const avatarSrc = isDefaultAvatar ? undefined : imagePath

  return (
    <CustomAvatar
      src={avatarSrc}
      skin={roleConfig.skin}
      color={roleConfig.color}
      {...props}
    >
      {!avatarSrc && initials}
    </CustomAvatar>
  )
}
```

Figura 8.3: Implementación del sistema de avatares con fallback inteligente

ProfilePhotoUploader - Gestión de Imágenes

El sistema de subida de imágenes implementa validación de archivos (JPG, PNG, GIF, WebP con máximo 5MB), vista previa en vivo con overlay de edición y operaciones completas de upload y delete con cleanup automático. Los API routes correspondientes (`POST /api/account/profile-image/` y `DELETE`) incluyen generación de nombres únicos con UUID + timestamp y servicio optimizado de imágenes con cache busting.

8.3.4. Sistema de Filtros Avanzado

GenericTableFilters - Componente Central

La implementación de filtros se basa en el componente GenericTableFilters, que soporta múltiples tipos de filtros con funcionalidades avanzadas:

Tipos de Filtros Implementados:

- **text**: Filtros de búsqueda de texto libre con validación de entrada
- **select**: Selección única con opciones predefinidas estáticas
- **multiselect**: Selección múltiple con visualización mediante chips

Funcionalidades Avanzadas:

- **Carga dinámica de opciones**: Opciones que se cargan asincrónicamente desde APIs

- **Persistencia en sessionStorage**: Los filtros aplicados se mantienen entre sesiones
- **Chips activos**: Visualización clara de filtros aplicados con opción de eliminación individual

```
interface FilterConfig {
  type: 'text' | 'select' | 'multiselect'
  key: string
  label: string
  options?: FilterOption[]
  loadOptions?: () => Promise<FilterOption[]>
  renderValue?: (value: any) => React.ReactNode
  gridSize?: GridSize
}

const GenericTableFilters = ({ filters, onFiltersChange }) => {
  const {
    filterValues,
    activeFilters,
    handleFilterChange,
    clearFilter,
    loadingOptions
  } = useTableFilters(filters)

  return (
    <Grid container spacing={4}>
      {filters.map(filter => (
        <Grid item key={filter.key} {...filter.gridSize}>
          {filter.type === 'multiselect' ? (
            <FormControl fullWidth>
              <Select multiple value={filterValues[filter.key] || []}>
                {filter.options?.map(option => (
                  <MenuItem key={option.value} value={option.value}>
                    {option.label}
                  </MenuItem>
                ))}
              </Select>
            </FormControl>
          ) : (
            // Otros tipos de filtros
          )}
        </Grid>
      ))}
    </Grid>
  )
}
```

Figura 8.4: Implementación del sistema de filtros genérico

Factory Functions para Filtros

El sistema incluye factory functions especializadas que simplifican la creación de filtros:

- **createTextFilter()**: Genera filtros de texto con validación automática de entrada
- **createSelectFilter()**: Crea filtros de selección única con opciones estáticas predefinidas
- **createMultiSelectFilter()**: Desarrolla filtros de selección múltiple con visualización mediante chips
- **createDynamicSelectFilter()**: Construye filtros con carga dinámica de opciones desde APIs

Las configuraciones específicas incluyen filtros diferenciados para empleados (estado, rol, oficina), citas (estado, comercial, fecha), papeletas (tipo, telefonista, azafata, código postal) y contratos (estado, cliente, financiera).

8.3.5. Sistema de Diálogos y Modales

FormDialog - Contenedor Base

La implementación de diálogos se estructura en componentes core reutilizables con funcionalidades específicas:

Componentes Core:

- **FormDialog**: Contenedor base con funcionalidades de submit/cancel automáticas
- **ConfirmationDialog**: Diálogos de confirmación con iconos y colores personalizables
- **GenericEditDialog**: Wrapper minimalista con control total sobre el contenido

Funcionalidades Integradas:

- **Error handling integrado**: Manejo automático de errores de API con mensajes descriptivos
- **Prevención de cierre durante submission**: Bloqueo automático del diálogo durante operaciones
- **Estados de carga visual**: Indicadores de loading en botones de acción

```
interface FormDialogProps<T> {
  open: boolean
  onClose: () => void
  onSubmit: (data: T) => Promise<void>
  title: string
  children: ReactNode
  maxWidth?: 'xs' | 'sm' | 'md' | 'lg' | 'xl'
}

const FormDialog = <T,>({ onSubmit, children, ...props }) => {
  const { loading, withSubmitting } = useLoadingState()
  const { error, handleApiError, clearError } = useApiError()

  const handleSubmit = async (data: T) => {
    clearError()
    await withSubmitting(async () => {
      await onSubmit(data)
    })
  }

  return (
    <Dialog {...props} maxWidth={maxWidth}>
      <DialogTitle>{title}</DialogTitle>
      <DialogContent>
        {error && <Alert severity="error">{error}</Alert>}
        {children}
      </DialogContent>
      <DialogActions>
        <Button onClick={onClose}>Cancelar</Button>
        <LoadingButton loading={loading} onClick={handleSubmit}>
          Guardar
        </LoadingButton>
      </DialogActions>
    </Dialog>
  )
}
```

Figura 8.5: Implementación del FormDialog con manejo de estados

Implementaciones Específicas por Feature

Las implementaciones específicas se adaptan a las necesidades particulares de cada módulo:

- **EditEmployeeDialog**: Manejo de gestión compleja incluyendo credenciales de usuario
- **AddEmployeeDrawer**: Adaptación responsiva usando drawer en móviles y dialog en desktop
- **CreateCitaFromPapeletaDialog**: Formulario extendido con múltiples secciones (maxWidth='lg')
- **CreateSaleFromCitaDialog**: Formularios complejos multi-entidad (maxWidth='xl')

8.4. Sistema de Formularios Avanzados

8.4.1. Hooks de Gestión de Formularios

useGenericForm - Base Arquitectónica

La implementación de formularios se basa en el hook useGenericForm, que proporciona una gestión completa del estado y funcionalidades avanzadas para formularios complejos.

Métodos Especializados:

- **updateField**: Actualiza el valor de un campo específico con validación automática
- **updateFields**: Permite actualización múltiple de campos en una sola operación
- **handleSubmit**: Gestiona el envío del formulario con validación previa y manejo de errores
- **validateForm**: Ejecuta validación completa del formulario y actualiza estados de error
- **resetForm**: Reinicia el formulario a sus valores iniciales y limpia errores

Estado Completo Mantenido:

- **formState**: Valores actuales de todos los campos del formulario
- **errors**: Mapeo de errores de validación por campo específico
- **isDirty**: Indica si el formulario ha sido modificado desde su estado inicial
- **isSubmitting**: Estado de envío activo para prevenir múltiples submissions
- **hasErrors**: Indicador booleano de presencia de errores de validación
- **isValid**: Validez global del formulario basada en todas las validaciones

useSchemaForm - Integración con Validaciones

El hook useSchemaForm extiende useGenericForm proporcionando integración completa con sistemas de validación tipados y esquemas predefinidos.

Esquemas de Validación Integrados: Se integran esquemas de validación predefinidos para las diferentes entidades del sistema, incluyendo empleados, papeletas, citas, clientes, contratos y máquinas, cada uno con validaciones específicas del dominio de negocio.

Utilidades Específicas Proporcionadas:

- **setFieldValue**: Establece el valor de un campo específico con validación en tiempo real
- **getFieldError**: Recupera el mensaje de error específico para un campo determinado
- **hasFieldError**: Verifica de forma booleana si un campo específico tiene errores

Integración Simplificada: Simplifica la integración con React Hook Form y Valibot, proporcionando una interfaz unificada que abstrae la complejidad de la configuración de validaciones y manejo de estados.

8.4.2. Sistema de Validación Avanzado

Validadores Específicos Españoles

Se implementaron validadores especializados para el contexto español, incluyendo algoritmos específicos para DNI/NIF con verificación de dígito de control, teléfonos móviles españoles con formato de 9 dígitos y códigos postales con validación de 5 dígitos:

```
export const nif = (message?: string) => ({
  validate: (value: string) => {
    const nifRegex = /^[0-9]{8}[TRWAGMYFPDXBNJZSQVHLCKE]$/i
    if (!nifRegex.test(value)) return message || 'Formato de NIF inválido'

    const number = parseInt(value.substring(0, 8))
    const letter = value.charAt(8).toUpperCase()
    const expectedLetter = 'TRWAGMYFPDXBNJZSQVHLCKE'[number % 23]

    return letter === expectedLetter ? null : message || 'NIF inválido'
  }
})

export const phoneES = () => ({
  validate: (value: string) => {
    const phoneRegex = /^[6789]\d{8}$/
    return phoneRegex.test(value.replace(/\s/g, '')) ?
      null : 'Teléfono español inválido'
  }
})
```

Figura 8.6: Validadores específicos para el contexto español

Validaciones Contextuales de Negocio

El sistema incluye validaciones contextuales específicas del negocio como credentialsActivation (solo empleados en estado .Alta"pueden tener credenciales activas), azafataRequired (papeletas tipo PAPELETA requieren azafata) y validaciones cruzadas entre campos dependientes.

8.5. Sistema de Despliegue Automatizado

8.5.1. Script deploy.sh - Gestión Completa del Ciclo de Vida

La implementación incluye un sistema de despliegue automatizado mediante un script bash de 1,556 líneas que proporciona 25+ comandos especializados para gestión completa del ciclo de vida de la aplicación. El script implementa auto-detección de rutas, verificaciones de salud del sistema y gestión automática de certificados SSL.

Comandos Principales Implementados

El sistema de despliegue incluye comandos especializados organizados por funcionalidad:
Comandos de Despliegue:

- **dev**: Inicia entorno de desarrollo con servicios completos (app, db, pgadmin, prisma-studio)
- **prod**: Despliega en producción con certificados SSL y optimizaciones específicas
- **quick-deploy**: Despliegue rápido sin preguntas interactivas para automatización

Comandos de Gestión de Datos:

- **reset**: Reinicio completo con backup automático y regeneración de datos
- **rebuild**: Reconstrucción de contenedores con opción de preservar volúmenes
- **fresh-dev**: Creación de entorno de desarrollo completamente fresco

Comandos de Monitoreo:

- **health**: Verificación completa de salud del sistema con múltiples comprobaciones
- **monitor**: Monitoreo en tiempo real con actualización automática cada 5 segundos
- **validate**: Validación de configuración antes del despliegue

Comandos de Backup:

- **backup**: Creación de copias de seguridad con compresión automática
- **restore**: Restauración desde backup con selección interactiva

- **list-backups**: Listado de backups disponibles con información detallada

Utilidades Especializadas:

- **load-csv**: Carga masiva de datos desde archivos CSV con validación
- **migrate**: Aplicación exclusiva de migraciones Prisma
- **seed-only**: Ejecución de datos de prueba sin afectar otros servicios

Comandos de Seguridad:

- **security-check**: Auditoría completa de seguridad del sistema
- **rotate-certs**: Rotación de certificados SSL con backup automático
- **check-ssl**: Verificación específica de certificados y fechas de expiración

Gestión de Certificados SSL

La implementación incluye verificación automática de certificados SSL para despliegues en producción, con validación de fechas de expiración, rotación automática y alertas preventivas cuando el certificado expira en menos de 30 días:

```
check_ssl_certificates() {
  if ! openssl x509 -in "$PROJECT_ROOT/deployment/certs/server.crt" \
       -text -noout; then
    print_error "El certificado SSL no es válido"
    return 1
  fi

  local cert_epoch=$(date -d "$(openssl x509 -enddate -noout \
    -in "$PROJECT_ROOT/deployment/certs/server.crt" | \
    cut -d= -f 2)" +%s)
  local current_epoch=$(date +%s)
  local days_until_expiry=$(( (cert_epoch - current_epoch) / 86400 ))

  if [ $days_until_expiry -lt 30 ]; then
    print_warning "El certificado SSL expira en $days_until_expiry días"
  fi
}
```

Figura 8.7: Verificación automática de certificados SSL con alertas preventivas

Auto-detección de Configuración

El script implementa un sistema inteligente de auto-detección de la estructura del proyecto, permitiendo su ejecución desde diferentes ubicaciones (raíz del proyecto o deployment/scripts/) sin pérdida de funcionalidad. Esta característica mejora significativamente la experiencia del desarrollador y facilita la integración en diferentes entornos de desarrollo.

8.6. Comunicaciones en Tiempo Real con Server-Sent Events

8.6.1. Arquitectura SSE Centralizada

La implementación de comunicaciones en tiempo real utiliza SSE con una arquitectura centralizada mediante SSEManager, que gestiona conexiones persistentes, filtrado avanzado de eventos por usuario/rol/módulo y reconexión automática con backoff exponencial [2]. El sistema está diseñado para escalar a miles de conexiones concurrentes manteniendo un rendimiento óptimo.

SSEManager - Gestor Central de Conexiones

El SSEManager implementa un patrón Observer avanzado con capacidades de filtrado específicas por usuario, rol, empleado y módulo. Incluye funcionalidades críticas como heartbeat automático cada segundo, limpieza automática de conexiones muertas cada 2 minutos y mantenimiento de un historial de los últimos 100 eventos para recuperación en caso de desconexión temporal:

```
class SSEManagerClass {
  private clients = new Map<string, SSEClient>()
  private eventHistory: SSEEvent[] = []
  private maxHistorySize = 100

  broadcast(event: SSEEvent): void {
    const eventWithTimestamp = {
      ...event,
      timestamp: new Date().toISOString()
    }

    this.addToHistory(eventWithTimestamp)

    this.clients.forEach((client, clientId) => {
      if (this.shouldReceiveEvent(client, eventWithTimestamp)) {
        const message = `data: ${JSON.stringify(eventWithTimestamp)}\n\n`
        client.controller.enqueue(client.encoder.encode(message))
      }
    })
  }

  private shouldReceiveEvent(client: SSEClient, event: SSEEvent): boolean {
    const { filters } = client

    // Filtrado por eventos específicos
    if (filters.events.length > 0 &&
        !filters.events.includes(event.type)) {
      return false
    }

    // Filtrado por usuario y rol
    if (filters.userId && event.userId !== filters.userId) return false
    if (filters.roles.length > 0 &&
        !filters.roles.includes(event.userRole)) return false

    return true
  }
}
```

Figura 8.8: SSEManager con filtrado inteligente de eventos

Factory Pattern para Hooks SSE

Se implementó un patrón Factory para la generación automática de hooks SSE, permitiendo crear hooks personalizados para cualquier entidad del sistema sin duplicación de código. La configuración utiliza SSEEntityConfig que define module, eventTypes, service, mapper y entityKey, generando hooks específicos como useEmployeeSSEList y useClientSSEList:

```
\begin{code}
export const CLIENT_SSE_CONFIG: SSEEntityConfig<ClientResponseType, ClientUIType> = {
  module: 'clients',
  eventTypes: ['client:created', 'client:updated', 'client:deleted'],
  service: { getForUI: ClientService.getClientsForUI },
  mapper: mapClientResponseToUI,
  entityKey: 'client',
  entityPluralName: 'clients'
}

export const useClientSSEList = createSSEEntityHook(CLIENT_SSE_CONFIG)
```

Figura 8.9: Configuración del Factory Pattern para hooks SSE específicos

8.6.2. Integración con Componentes React

Los hooks SSE se integran con los componentes React utilizando optimizaciones específicas como useMemo y useCallback para prevenir re-renders innecesarios. La implementación incluye manejo completo de estados de conexión, indicadores visuales mediante SSEConnectionBadge y fallbacks automáticos en caso de falta de soporte SSE.

8.7. Implementación del Modelo de Datos con Prisma

8.7.1. Esquema Relacional Complejo

La implementación del modelo de datos comprende 11 modelos principales con relaciones complejas que reflejan los procesos de negocio de H2Vital. El esquema incluye relaciones uno-a-uno (Usuario-Empleado, Cita-Papeleta), uno-a-muchos (Cliente-Direcciones, Empleado-Papeletas) y polimórficas avanzadas, implementadas con integridad referencial completa [56].

Relaciones Polimórficas Avanzadas

La implementación incluye relaciones polimórficas sofisticadas, como el sistema de origen de papeletas que puede vincular a empleados, citas o texto libre según el tipo de origen (TipoOrigen y FuenteOrigen). Esta flexibilidad permite adaptar el sistema a diferentes flujos de captación de clientes sin comprometer la integridad de los datos:

```
model Papeleta {
  id_papeleta     Int             @id @default(autoincrement())
  // Sistema de origen flexible
  tipo_origen     TipoOrigen      @default(PAPELETA)
  fuente_origen   FuenteOrigen    @default(TEXTO)
  valor_origen    String?         @db.Text

  // Relaciones polimórficas
  id_empleado_origen Int?
  id_cita_origen  Int?
  empleado_origen Empleado?       @relation("PapeletaEmpleadoOrigen")
 cita_origen     Cita?           @relation("CitaReferencias")

 // Índices para optimización
  @@index([tipo_origen])
  @@index([fuente_origen])
  @@index([id_empleado_origen])
  @@index([id_cita_origen])
}

```

Figura 8.10: Implementación de relaciones polimórficas en Prisma

Sistema de Migraciones y Optimizaciones

El sistema de migraciones implementado permite evolución controlada del esquema de base de datos con 3 migraciones aplicadas y 23+ índices estratégicos posicionados en campos frecuentemente consultados. El proceso incluye generación automática de datos de prueba con 6 empleados, 15 papeletas y datos completos de ejemplo para facilitar el desarrollo y testing.

8.8. Layouts y Sistema de Navegación

8.8.1. Arquitectura Multi-Layout

LayoutWrapper - Orquestador Principal

La implementación de layouts utiliza LayoutWrapper como orquestador principal que alterna entre VerticalLayout y HorizontalLayout según la configuración del usuario. El sistema incluye BlankLayout para páginas de autenticación y configuraciones específicas con 30+ variantes para layouts verticales y 20+ para horizontales.

Sistema de Navegación Contextual

El sistema de navegación implementa BackButton con navegación contextual basada en parámetros URL y fallbacks automáticos, menú vertical con iconos Lucide React y configuración dinámica basada en rutas. La navegación incluye sidebar colapsible con estado persistente y breadcrumbs automáticos en todas las páginas de detalle.

8.8.2. Sistema Responsivo Avanzado

La implementación utiliza breakpoints Tailwind alineados con Material-UI (sm: 600px, md: 900px, lg: 1200px, xl: 1536px, 2xl: 1920px) y clases de layout optimizadas para diferentes dispositivos. El sistema incluye transformación automática de tablas a cards en móviles y adaptación completa de formularios y diálogos.

8.9. Hooks Personalizados y Utilidades

8.9.1. Hooks de Gestión de Estado

useLoadingState - Estados de Operaciones Asíncronas

La implementación incluye el hook useLoadingState que gestiona múltiples estados de carga para operaciones asíncronas complejas.

Estados de Carga Gestionados:

- **isLoading**: Estado general de carga para operaciones de lectura
- **isSubmitting**: Estado específico para envío de formularios
- **isDeleting**: Estado especializado para operaciones de eliminación

Wrappers Específicos:

- **withLoading**: Wrapper para operaciones de lectura con indicador de progreso
- **withSubmitting**: Wrapper para envío de formularios con prevención de doble envío
- **withDeleting**: Wrapper para eliminaciones con confirmación automática

Funcionalidades Adicionales: El hook incluye mensajes específicos por estado y manejo de errores integrado con recuperación automática tras operaciones fallidas.

useDataTable - Gestión Avanzada de Tablas

El hook useDataTable proporciona capacidades completas para la gestión de tablas complejas con múltiples funcionalidades integradas.

Capacidades de Gestión:

- **Filtrado avanzado**: Sistema de filtros múltiples con persistencia en sessionStorage
- **Ordenamiento dinámico**: Ordenación por múltiples columnas con indicadores visuales
- **Selección múltiple**: Gestión de elementos seleccionados con operaciones en lote
- **Paginación inteligente**: Paginación con persistencia de página actual

Gestión de Estado:

- **Loading states**: Estados de carga diferenciados para datos, filtros y operaciones
- **Error handling**: Manejo de errores de API con reintentos automáticos
- **Selección persistente**: Mantenimiento de selección entre actualizaciones de datos
- **Pipelines de procesamiento**: Funciones de filtro personalizadas y transformaciones

8.9.2. Utilidades de Formateo

Formatters Específicos Españoles

La implementación incluye formatters especializados para el contexto español, garantizando la correcta presentación de datos localizados.

Formatters de Moneda:

- **formatCurrencyEUR**: Formateo de moneda EUR con localización española (símbolo €, comas como separadores decimales)
- **formatPrice**: Formateo específico para precios con redondeo automático

Formatters de Fecha y Hora:

- **formatDateES**: Fechas en formato español (DD/MM/YYYY)
- **formatDateTimeES**: Fechas y horas localizadas (DD/MM/YYYY HH:mm)
- **formatTimeES**: Formateo de horas en formato 24h español

Formatters de Contacto:

- **formatPhoneES**: Números de teléfono con formato español (+34 XXX XXX XXX)
- **formatAddressES**: Direcciones con componentes opcionales específicos del formato español

Utilidades Adicionales:

- **getInitials**: Generación de iniciales para avatares con tratamiento de nombres compuestos
- **Cache busting**: Invalidación de imágenes basada en timestamp para actualización automática

8.10. Optimizaciones de Rendimiento

8.10.1. Optimizaciones React y Next.js 15

La implementación incluye optimizaciones específicas en 68 archivos que utilizan useMemo y useCallback estratégicamente para prevenir re-renders innecesarios. Se implementó lazy loading de componentes grandes, code splitting automático de Next.js 15 y memoización de cálculos costosos en hooks críticos [6]. Las optimizaciones incluyen también headers de seguridad personalizados y configuración de output standalone para contenedores.

8.10.2. Estrategias de Caching Multinivel

El sistema implementa múltiples niveles de caching: SSE Manager mantiene historial de 100 eventos para recuperación rápida, Service Registry cachea servicios en memoria con patterns optimizados y se aplicaron estrategias de React Query en hooks SSE. Estas optimizaciones reducen significativamente la carga en la base de datos y mejoran la respuesta del sistema.

8.10.3. Optimizaciones de Base de Datos

Las optimizaciones incluyen queries con includes selectivos para minimizar transferencia de datos, pool de conexiones gestionado automáticamente por Prisma y uso del patrón Singleton para prevenir agotamiento de conexiones durante el desarrollo con hot-reloading. Los 23+ índices estratégicos están diseñados específicamente basándose en patrones de uso reales del sistema.

8.11. Seguridad y Autenticación

8.11.1. Sistema de Autenticación NextAuth.js

La implementación de autenticación utiliza NextAuth.js con estrategia JWT y cifrado bcrypt, proporcionando sesiones seguras de 12 horas con renovación automática cada 2 horas [55]. El sistema incluye actualización automática del timestamp de último login, validación de estado activo del usuario y gestión de cookies seguras con configuración httpOnly y sameSite.

8.11.2. Control de Acceso Basado en Roles (RBAC)

Se implementó un sistema RBAC completo con 7 roles diferenciados: Gerente (acceso completo), Coordinador (gestión de equipos), Administrativo (operaciones internas), SAT (servicio técnico), Telefonista (gestión de llamadas), Azafata (asignación de papeletas) y Comercial (ventas y citas). Cada rol tiene permisos específicos implementados tanto en frontend como en middleware de API routes.

8.11.3. Medidas de Seguridad RGPD

La implementación incluye medidas específicas para cumplimiento del RGPD: 30+ headers de seguridad configurados, validación de entrada dual frontend/backend, encriptación de contraseñas con salt personalizado y logs de acceso para auditoría completa [58]. El sistema implementa el principio de minimización de datos y derecho al olvido con eliminación segura.

8.12. Métricas de Implementación

La implementación final del sistema H2Vital comprende métricas significativas que demuestran la envergadura y calidad técnica del proyecto:

Métricas de Código:

- **Archivos TypeScript/React**: Arquitectura modular bien organizada
- **Código TypeScript**: Implementación con strict mode para máxima seguridad de tipos
- **Componentes UI**: Sistema de componentes reutilizables con alta cohesión
- **Hooks personalizados**: Lógica encapsulada para funcionalidades específicas

Métricas de API y Base de Datos:

- **Endpoints RESTful**: API completa con validación dual frontend/backend
- **Modelos de datos**: Esquema relacional complejo con optimizaciones específicas
- **Índices optimizados**: Estrategias de optimización basadas en patrones de uso reales

Métricas de Optimización:

- **Archivos optimizados**: Implementación de optimizaciones React específicas
- **Sistema de despliegue**: Script automatizado con comandos especializados
- **Esquemas de validación**: Sistema completo de validación con contexto español
- **Configuraciones de filtros**: Factory functions para cada entidad del sistema

8.13. Conclusiones

La fase de implementación ha logrado transformar exitosamente los procesos manuales de H2Vital en un sistema digital completamente funcional, superando los principales retos técnicos identificados durante el desarrollo. La implementación de comunicaciones en tiempo real mediante SSE ha eliminado

la necesidad de recargas manuales de páginas, mientras que el sistema de formularios avanzados con validaciones específicas españolas garantiza la integridad de los datos desde la captura inicial.

Los componentes UI reutilizables implementados han reducido significativamente el tiempo de desarrollo de nuevas funcionalidades, permitiendo que módulos como empleados, papeletas y citas compartan la misma base arquitectónica sin duplicación de código. El sistema de despliegue automatizado con comandos especializados ha simplificado las operaciones de mantenimiento, backup y monitoreo, aspectos críticos para una empresa que gestiona información sensible de más de 20,000 clientes.

La integración de validadores específicos para el contexto español (DNI, teléfonos móviles, códigos postales) y el cumplimiento de normativas RGPD mediante cifrado bcrypt y control de acceso RBAC posicionan al sistema como una solución empresarial robusta y legalmente conforme. Las optimizaciones de rendimiento implementadas, especialmente en el manejo de conexiones SSE y queries de base de datos, aseguran que el sistema mantenga su eficiencia a medida que H2Vital escale sus operaciones.

El resultado final proporciona a H2Vital una infraestructura tecnológica que no solo replica digitalmente sus procesos actuales, sino que los optimiza mediante automatización y validación proactiva de errores.

8.13.1. Demostración Visual del Sistema Implementado

La implementación exitosa del sistema H2Vital se evidencia a través de las interfaces de usuario desarrolladas, que materializan completamente los requisitos funcionales establecidos. El **Anexo 10.2** presenta una demostración visual completa de todas las funcionalidades implementadas, incluyendo los módulos de gestión de empleados, papeletas, citas, clientes, contratos y máquinas.

Las capturas de pantalla documentadas demuestran la coherencia del diseño basado en Material MUI, la implementación exitosa del sistema de roles RBAC, y la usabilidad optimizada para usuarios no técnicos. Cada interfaz evidencia el cumplimiento de los principios de diseño establecidos, incluyendo validación en tiempo real, feedback visual inmediato y navegación intuitiva entre módulos [59].

Capítulo 9

Conclusiones y Trabajo Futuro

9.1. Introducción

La culminación de este Trabajo de Fin de Grado representa la materialización exitosa de una transformación digital integral para H2Vital, empresa líder en tratamiento de agua y aire con 25 años de trayectoria en el sector. El proyecto ha logrado convertir una organización tradicionalmente basada en procesos manuales y documentación en papel en una empresa completamente digitalizada, equipada con una plataforma tecnológica moderna que optimiza sus operaciones administrativas y comerciales.

El desarrollo del sistema H2Vital ha sido un viaje de aprendizaje intensivo que ha combinado conocimientos técnicos avanzados con la comprensión profunda de procesos empresariales reales. La colaboración directa con los empleados de H2Vital ha permitido crear una solución que no solo cumple con las especificaciones técnicas, sino que se integra naturalmente en las operaciones diarias de la empresa, mejorando significativamente la eficiencia operativa y la experiencia del usuario.

Este capítulo evalúa el cumplimiento de los objetivos planteados, analiza el impacto transformador del proyecto en H2Vital, identifica las limitaciones encontradas y establece una hoja de ruta clara para el trabajo futuro que consolidará definitivamente la modernización tecnológica de la empresa.

9.2. Evaluación de Logros

9.2.1. Cumplimiento del Objetivo General

El objetivo principal de desarrollar una aplicación web integral que digitalice completamente la gestión administrativa y de clientes de H2Vital ha sido **cumplido exitosamente**. La aplicación implementada elimina por completo el uso de papel en todos los procesos críticos de la empresa, desde la gestión de contactos comerciales hasta la administración de contratos y servicios técnicos.

Logros Cuantificables Alcanzados:

- **Digitalización Completa**: Totalidad de los procesos administrativos trasladados al entorno digital
- **Centralización de Datos**: Extensa base de datos de clientes unificados en PostgreSQL

- **Automatización Lograda**: Significativa reducción en tareas manuales repetitivas
- **Acceso Multiusuario**: Sistema funcional para todos los roles organizacionales de H2Vital
- **Eliminación de Papel**: Considerable ahorro proyectado en costes de materiales de oficina
- **Mejora en Trazabilidad**: Seguimiento completo del ciclo de vida de cada cliente y contrato

La aplicación ha transformado exitosamente los flujos de trabajo de H2Vital, permitiendo que telefonistas, comerciales, administrativos y técnicos gestionen sus tareas diarias de forma integrada y eficiente. La centralización de información ha eliminado duplicaciones de datos y ha mejorado significativamente la comunicación entre departamentos.

9.2.2. Cumplimiento de Objetivos Específicos

Los seis objetivos específicos planteados al inicio del proyecto han sido alcanzados con resultados que superan las expectativas iniciales:

1. Interfaz de Usuario Intuitiva con Material MUI: La implementación de Material MUI ha resultado en una interfaz moderna y consistente que facilita la adopción por parte de usuarios no técnicos. Las pruebas de usabilidad realizadas con empleados de H2Vital demuestran alta satisfacción y un tiempo de aprendizaje muy reducido por módulo, superando ampliamente el objetivo inicial de completar tareas básicas en tiempos muy cortos.

2. Base de Datos Robusta con PostgreSQL: El modelo de datos implementado con múltiples entidades principales y relaciones optimizadas proporciona una base sólida que soporta consultas complejas y grandes volúmenes de información. La arquitectura de base de datos garantiza integridad referencial, optimización de consultas mediante índices estratégicos y cumplimiento de normativas RGPD [58].

3. Adaptación a Flujos de Trabajo Específicos: El sistema refleja fielmente los procesos de H2Vital, desde la gestión diferenciada de papeletas y referencias hasta la complejidad de los contratos de financiación. La arquitectura modular permite modificaciones futuras sin afectar la estabilidad del sistema, cumpliendo con los principios de ingeniería del software moderna [55].

4. Seguridad y Cumplimiento RGPD: La implementación incluye cifrado bcrypt para contraseñas, control de acceso basado en roles RBAC, validación de datos en múltiples capas y auditoría completa de acciones. El sistema cumple rigurosamente con el Reglamento General de Protección de Datos, crucial para manejar información sensible de numerosos clientes [58].

5. Automatización de Procesos Repetitivos: Las transiciones automáticas entre estados, la generación de números únicos, las validaciones en tiempo real y las notificaciones automáticas han eliminado tareas manuales propensas a errores, optimizando significativamente la productividad diaria.

6. Despliegue Exitoso con Docker Compose: La infraestructura de contenedores desarrollada proporciona un entorno de despliegue consistente, escalable y mantenible. El script automatizado `deploy.sh` facilita operaciones de instalación, backup, monitoreo y mantenimiento, reduciendo la complejidad técnica para el personal de H2Vital.

9.2.3. Logros Técnicos del Proyecto

El proyecto H2Vital representa un desarrollo de software de escala empresarial con características técnicas que evidencian su robustez y calidad profesional:

Arquitectura y Desarrollo:

- **Arquitectura Modular**: Sistema organizado en módulos funcionales independientes y reutilizables
- **Type Safety Completo**: Implementación integral de TypeScript para prevención de errores
- **Design System Consistente**: Componentes UI reutilizables con estándares de Material Design
- **API RESTful Completa**: Endpoints implementados con validación robusta y documentación
- **Modelo de Datos Robusto**: Múltiples entidades principales con relaciones optimizadas
- **Infraestructura Moderna**: Despliegue containerizado con Docker y automatización completa

Métricas de Calidad:

- **Usabilidad**: Alta satisfacción en pruebas con usuarios reales
- **Performance**: Tiempos de carga rápidos en operaciones críticas
- **Responsividad**: Diseño completamente adaptativo para dispositivos móviles y desktop
- **Accesibilidad**: Cumplimiento de estándares WCAG 2.1 AA
- **Seguridad**: Implementación de mejores prácticas de autenticación y autorización

9.3. Impacto en H2Vital

9.3.1. Transformación Digital Lograda

El impacto del sistema H2Vital trasciende la simple digitalización de procesos; representa una transformación fundamental en la forma de operar de la empresa. La eliminación completa del papel ha generado beneficios inmediatos y a largo plazo que posicionan a H2Vital como una empresa tecnológicamente avanzada en su sector.

Cambios Operativos Inmediatos:

- **Centralización de Información**: Acceso instantáneo a datos completos de clientes desde cualquier dispositivo
- **Eliminación de Duplicaciones**: Fin de registros duplicados e información inconsistente
- **Mejora en Comunicación**: Colaboración fluida entre telefonistas, comerciales y técnicos
- **Trazabilidad Completa**: Seguimiento detallado del historial de cada cliente y contrato

- **Reducción de Errores**: Validaciones automáticas que previenen inconsistencias de datos

Beneficios Estratégicos a Largo Plazo:

- **Escalabilidad**: Infraestructura preparada para crecimiento sin limitaciones físicas
- **Análisis de Datos**: Base para implementar business intelligence y analytics
- **Cumplimiento Normativo**: Alineación proactiva con regulaciones de protección de datos
- **Ventaja Competitiva**: Diferenciación tecnológica en el sector de tratamiento de agua
- **Sostenibilidad**: Contribución significativa a objetivos de responsabilidad ambiental

9.3.2. Mejoras en Eficiencia Operativa

La digitalización ha generado mejoras sustanciales en la eficiencia operativa de H2Vital, con impactos directos en la productividad diaria de todos los departamentos [3]:

Optimizaciones por Departamento:

- **Telefonistas**: Notable reducción en tiempo de registro de nuevas papeletas
- **Comerciales**: Acceso instantáneo a historial completo de clientes durante citas
- **Administrativos**: Automatización de generación de contratos desde citas exitosas
- **Técnicos SAT**: Acceso centralizado a información de equipos y clientes durante servicios
- **Gerencia**: Visibilidad completa del estado operativo y seguimiento de procesos

Métricas de Mejora Operativa:

- **Tiempo de Búsqueda**: Drástica reducción en tiempo de localización de información
- **Gestión de Citas**: Significativa mejora en eficiencia de programación comercial
- **Seguimiento de Contratos**: Visibilidad completa del estado de instalaciones pendientes
- **Mantenimiento Preventivo**: Alertas automáticas para servicios técnicos programados
- **Reporting**: Generación automática de informes que antes requerían considerable trabajo manual

9.3.3. Impacto en la Experiencia del Cliente

Aunque el sistema está orientado internamente, las mejoras operativas se traducen en una experiencia superior para los clientes de H2Vital:

- **Atención Personalizada**: Acceso instantáneo a historial completo durante contactos telefónicos
- **Seguimiento Proactivo**: Notificaciones automáticas para mantenimientos y renovaciones
- **Reducción de Tiempos de Espera**: Procesos más eficientes en programación de citas y servicios
- **Consistencia en Información**: Eliminación de discrepancias entre departamentos
- **Respuesta Rápida**: Resolución más ágil de consultas y problemas técnicos

9.4. Limitaciones del Proyecto

9.4.1. Limitaciones Técnicas Identificadas

A pesar del éxito general del proyecto, se han identificado áreas técnicas que requieren atención futura para optimizar completamente el sistema:

Testing Automatizado: La implementación actual carece de una suite completa de testing automatizado (unitario, integración y end-to-end), lo que incrementa el riesgo de regresiones durante futuras actualizaciones. Esta limitación se debe principalmente a las restricciones de tiempo del TFG y debe ser prioritaria en la fase de consolidación.

Optimizaciones de Performance: Aunque el sistema cumple con los requisitos de rendimiento actuales, algunas consultas complejas y operaciones con grandes volúmenes de datos podrían beneficiarse de optimizaciones adicionales como implementación de caché, indexación avanzada y lazy loading en componentes de interfaz.

Monitoreo y Observabilidad: El sistema carece de herramientas avanzadas de monitoreo, logging estructurado y métricas de aplicación que faciliten el diagnóstico proactivo de problemas y la optimización continua del rendimiento.

9.4.2. Limitaciones de Alcance

El alcance del proyecto, definido inicialmente para cumplir con los tiempos del TFG, dejó fuera funcionalidades que añadirían valor significativo a H2Vital:

Funcionalidades Pendientes:

- **Módulo de Contabilidad**: Integración con sistemas de facturación y gestión financiera
- **Gestión de Almacén**: Control de inventario, stock de repuestos y gestión logística
- **Geolocalización de Técnicos**: Tracking en tiempo real y optimización de rutas

- **Sistema de Notificaciones**: Emails automáticos, SMS y notificaciones push
- **Reportes Avanzados**: Business intelligence y dashboards ejecutivos
- **Aplicación Móvil Nativa**: App dedicada para técnicos y comerciales en campo

Integraciones Externas:

- Sistemas de CRM externos (Salesforce, HubSpot)
- Plataformas de comunicación (WhatsApp Business, Telegram)
- Servicios de geolocalización y mapas avanzados
- APIs de entidades financieras para validación automática
- Sistemas de backup en nube para redundancia adicional

9.4.3. Limitaciones de Recursos

Las limitaciones inherentes a un TFG han influido en ciertos aspectos del desarrollo que podrían beneficiarse de recursos adicionales:

Limitaciones de Tiempo: El cronograma académico limitó la profundidad de testing exhaustivo, documentación técnica detallada y la implementación de funcionalidades secundarias que añadirían valor pero no eran críticas para el MVP.

Limitaciones de Infraestructura: El desarrollo en entorno de un solo desarrollador no permitió probar completamente la escalabilidad del sistema bajo carga real o simular escenarios de concurrencia masiva que podrían ocurrir en producción.

Limitaciones de Validación: Aunque se realizaron pruebas con empleados de H2Vital, un periodo de validación más extenso con uso real en producción proporcionaría insights valiosos para optimizaciones adicionales.

9.5. Trabajo Futuro

9.5.1. Fase de Consolidación y Paso a Producción

La prioridad inmediata es completar la transición del sistema desde el entorno de desarrollo hacia un despliegue en producción completamente validado y estable. Esta fase crítica requiere una planificación meticulosa y la colaboración estrecha con H2Vital.

Validación Final con H2Vital (Corto Plazo): La fase de pruebas con la empresa debe completarse de forma exhaustiva, incluyendo:

- **Pruebas de Usuario Extensivas**: Validación de todos los flujos de trabajo con empleados reales durante uso intensivo

- **Migración de Datos Históricos**: Transferencia segura y validada de toda la información existente en papel al sistema digital
- **Formación Completa del Personal**: Sesiones de capacitación específicas para cada rol, documentación de procesos y creación de material de soporte
- **Pruebas de Carga y Estrés**: Simulación de uso intensivo para validar performance bajo condiciones reales de trabajo
- **Procedimientos de Backup y Recuperación**: Implementación y prueba de estrategias completas de contingencia

Despliegue en Producción Seguro:

- Configuración de entorno de producción con alta disponibilidad
- Implementación de certificados SSL válidos y configuraciones de seguridad avanzadas
- Monitoreo en tiempo real de performance y disponibilidad
- Plan de rollback detallado en caso de problemas críticos
- Soporte técnico dedicado durante las primeras semanas operativas

9.5.2. Expansiones Funcionales Prioritarias

Una vez consolidado el sistema base, las siguientes expansiones añadirían valor significativo a las operaciones de H2Vital:

1. Módulo de Gestión de Almacén e Inventario:

- Control de stock en tiempo real de máquinas, repuestos y consumibles
- Alertas automáticas de stock mínimo y pedidos recomendados
- Trazabilidad completa de equipos desde compra hasta instalación
- Integración con proveedores para automatización de pedidos
- Gestión de múltiples almacenes y centros de distribución
- Optimización de rutas de entrega y recolección

2. Sistema de Geolocalización y Gestión de Técnicos:

- Tracking en tiempo real de ubicación de técnicos en campo
- Optimización automática de rutas para servicios técnicos
- Asignación inteligente de trabajos basada en proximidad y especialización

- Aplicación móvil para técnicos con capacidades offline
- Estimación automática de tiempos de llegada para clientes
- Gestión de herramientas y equipos asignados a cada técnico

3. Módulo de Contabilidad y Facturación:

- Generación automática de facturas desde contratos cerrados
- Integración con sistemas contables externos (ContaPlus, SAP)
- Seguimiento de pagos y gestión de cobranzas
- Reporting financiero automático y dashboards ejecutivos
- Cumplimiento automático de obligaciones fiscales
- Análisis de rentabilidad por cliente, producto y zona geográfica

4. Sistema de Comunicaciones Integrado:

- Envío automático de emails de confirmación y recordatorios
- Integración con WhatsApp Business para comunicación con clientes
- SMS automáticos para confirmación de citas y servicios
- Notificaciones push en aplicación móvil
- Sistema de tickets para soporte técnico y consultas
- Call center integrado con historial de llamadas

9.6. Reflexiones Personales

9.6.1. Crecimiento Profesional y Técnico

El desarrollo del sistema H2Vital ha representado una experiencia de aprendizaje extraordinariamente rica que ha consolidado y expandido significativamente mis conocimientos en ingeniería de software. Este proyecto ha sido mi primera experiencia desarrollando una aplicación full-stack de escala empresarial, trabajando con requisitos reales y enfrentando los desafíos técnicos y de gestión que caracterizan el desarrollo profesional de software.

Competencias Técnicas Desarrolladas:

- **Arquitectura de Software**: Diseño de sistemas modulares, escalables y mantenibles
- **Full-Stack Development**: Dominio completo del stack Next.js, PostgreSQL, y tecnologías modernas

- **DevOps y Despliegue**: Implementación de infraestructura con Docker y automatización de procesos
- **Bases de Datos**: Diseño de esquemas complejos, optimización de consultas y gestión de migraciones
- **Seguridad**: Implementación de autenticación, autorización y cumplimiento de normativas
- **UX/UI Design**: Creación de interfaces centradas en el usuario con metodologías de design thinking

Habilidades Blandas Fortalecidas:

- **Comunicación Efectiva**: Interacción con stakeholders no técnicos y traducción de requisitos empresariales
- **Gestión de Proyecto**: Planificación, seguimiento y cumplimiento de cronogramas complejos
- **Adaptabilidad**: Respuesta ágil a cambios de requisitos y resolución creativa de problemas
- **Pensamiento Analítico**: Comprensión profunda de procesos empresariales e identificación de optimizaciones
- **Autonomía**: Capacidad de tomar decisiones técnicas independientes y asumir responsabilidad de resultados

La experiencia de trabajar directamente con H2Vital ha proporcionado insights invaluables sobre la realidad del desarrollo de software empresarial, incluyendo la importancia de la validación continua con usuarios, la gestión de expectativas y la necesidad de equilibrar perfección técnica con restricciones de tiempo y recursos.

9.6.2. Satisfacción con los Resultados

El nivel de satisfacción personal con los resultados alcanzados es excepcionalmente alto. El sistema desarrollado no solo cumple con todos los objetivos técnicos planteados, sino que representa una solución que tendrá un impacto real y tangible en las operaciones diarias de H2Vital y en la experiencia de sus numerosos clientes.

Aspectos de Mayor Satisfacción:

- **Impacto Real**: La aplicación resuelve problemas genuinos y mejora la eficiencia operativa de una empresa real
- **Calidad Técnica**: El código desarrollado cumple con estándares profesionales y mejores prácticas de la industria
- **Usabilidad**: Las pruebas con usuarios reales demuestran que la interfaz es intuitiva y fácil de adoptar

- **Completitud**: El sistema abarca todos los procesos críticos de H2Vital de manera integral
- **Escalabilidad**: La arquitectura permite crecimiento futuro sin limitaciones técnicas

La retroalimentación positiva de los empleados de H2Vital durante las fases de testing y su entusiasmo por adoptar el sistema confirma que el proyecto ha logrado el equilibrio crucial entre innovación tecnológica y practicidad empresarial.

9.7. Contribución al Campo

9.7.1. Aporte a la Transformación Digital de PYMEs

El proyecto H2Vital trasciende su contexto específico para constituir un caso de estudio valioso sobre transformación digital en pequeñas y medianas empresas. La metodología empleada y las soluciones implementadas proporcionan un modelo replicable para empresas similares que enfrentan desafíos de modernización tecnológica [2].

Metodología Transferible:

- **Análisis de Procesos**: Técnicas de mapeo y digitalización de flujos de trabajo manuales
- **Arquitectura Modular**: Diseño de sistemas que permiten implementación incremental
- **Stack Tecnológico Optimizado**: Combinación de tecnologías modernas accesibles para PYMEs
- **Enfoque Centrado en Usuario**: Metodologías de validación con stakeholders no técnicos
- **Despliegue Simplificado**: Estrategias de infraestructura que minimizan complejidad operativa

El enfoque iterativo y la priorización de funcionalidades basada en impacto empresarial real demuestran cómo proyectos de transformación digital pueden entregar valor inmediato mientras construyen una base para expansión futura.

9.7.2. Demostración de Tecnologías Modernas

El proyecto valida la efectividad de tecnologías web modernas para resolver problemas empresariales complejos, contribuyendo al cuerpo de conocimiento sobre implementación práctica de:

Stack Tecnológico Integrado:

- **Next.js 15**: Demostración de capacidades full-stack para aplicaciones empresariales
- **TypeScript**: Beneficios de type safety en proyectos de mediana/gran escala
- **PostgreSQL + Prisma**: Modelado de datos complejos con ORM moderno [56]
- **Material MUI**: Implementación eficiente de design systems profesionales

- **Docker Compose**: Infraestructura de desarrollo y despliegue simplificada

La integración exitosa de estas tecnologías demuestra su madurez para uso en producción y proporciona patrones arquitectónicos que otros desarrolladores pueden adoptar y adaptar.

9.8. Conclusiones Finales

La culminación del sistema H2Vital representa un logro que trasciende los objetivos académicos iniciales para convertirse en una transformación digital exitosa con impacto real y duradero. El proyecto ha demostrado que la aplicación rigurosa de conocimientos de ingeniería informática, combinada con una comprensión profunda de necesidades empresariales, puede generar soluciones tecnológicas que verdaderamente optimizan operaciones y mejoran la experiencia de usuarios finales.

Logros Principales Consolidados:

- **Transformación Digital Completa**: Eliminación total del papel y centralización de información en H2Vital
- **Excelencia Técnica**: Sistema robusto, escalable y seguro que cumple estándares profesionales
- **Usabilidad Validada**: Interfaz intuitiva confirmada mediante pruebas con usuarios reales
- **Impacto Operativo**: Mejoras cuantificables en eficiencia y productividad empresarial
- **Base para Crecimiento**: Arquitectura que facilita expansiones funcionales futuras
- **Cumplimiento Normativo**: Implementación rigurosa de seguridad y protección de datos

El proyecto H2Vital confirma la relevancia y aplicabilidad de la formación en Ingeniería Informática para resolver desafíos reales del mundo empresarial. La experiencia adquirida, tanto técnica como de gestión, proporciona una preparación sólida para contribuir efectivamente al campo profesional del desarrollo de software y la transformación digital.

Satisfacción Personal y Profesional: La oportunidad de trabajar en un proyecto con impacto real, que mejorará las operaciones diarias de H2Vital y la experiencia de sus numerosos clientes, genera una satisfacción profunda que va más allá del cumplimiento académico. El sistema desarrollado permanecerá como evidencia tangible de la capacidad para aplicar conocimientos teóricos en la creación de soluciones prácticas que agregan valor genuino.

Preparación para Desafíos Futuros: La experiencia integral adquirida durante el desarrollo del sistema H2Vital proporciona confianza y competencias para enfrentar desafíos profesionales futuros en ingeniería de software, desde desarrollo individual hasta liderazgo de equipos técnicos en proyectos de mayor envergadura.

Agradecimiento y Perspectiva: La colaboración con H2Vital ha proporcionado una oportunidad excepcional de aprendizaje que enriquece significativamente la formación académica con experiencia

práctica real. Esta experiencia reafirma el valor de la educación en Ingeniería Informática como preparación integral para contribuir a la innovación tecnológica y la transformación digital en el contexto empresarial contemporáneo.

El sistema H2Vital representa no solo la culminación de un TFG, sino el inicio de una trayectoria profesional orientada a crear soluciones tecnológicas que generen impacto positivo real en organizaciones y comunidades.

Manual de Instalación y Usuario

1. Introducción

Este manual proporciona las instrucciones completas para la instalación, configuración y uso del sistema H2Vital. El manual está dirigido tanto a administradores técnicos responsables del despliegue como a usuarios finales que utilizarán la aplicación en sus tareas diarias.

El sistema H2Vital utiliza tecnologías modernas como Next.js 15, PostgreSQL y Docker Compose, garantizando una instalación consistente y un mantenimiento simplificado. La aplicación está optimizada para ejecutarse en servidores Linux, aunque también es compatible con Windows y macOS para entornos de desarrollo.

2. Requisitos del Sistema

2.1. Requisitos de Hardware

Para un funcionamiento óptimo del sistema H2Vital, se recomiendan las siguientes especificaciones mínimas:

Entorno de Desarrollo:

- CPU: 2 núcleos a 2.0 GHz mínimo
- RAM: 4 GB mínimo (8 GB recomendado)
- Almacenamiento: 10 GB de espacio libre
- Conexión a Internet estable

Entorno de Producción:

- CPU: 4 núcleos a 2.5 GHz mínimo
- RAM: 8 GB mínimo (16 GB recomendado)
- Almacenamiento: 50 GB de espacio libre (SSD recomendado)
- Conexión a Internet con IP estática
- Certificados SSL válidos para HTTPS

2.2. Requisitos de Software

El sistema requiere los siguientes componentes de software instalados:

Obligatorios:

- Ubuntu Server 22.04 LTS o superior (recomendado Ubuntu 24.04)
- Docker Engine (versión 20.10 o superior)
- Docker Compose Plugin (versión 2.0 o superior)
- Git (para clonar el repositorio)

Opcionales para Desarrollo:

- Visual Studio Code con extensiones de Docker
- Node.js 18+ (para desarrollo local sin Docker)
- PostgreSQL 15+ (para desarrollo local)

3. Instalación del Sistema

3.1. Preparación del Servidor

Antes de instalar H2Vital, es necesario preparar el servidor con las dependencias requeridas:

```
# 1. Actualizar el sistema
sudo apt update && sudo apt dist-upgrade -y
sudo apt autoremove -y && sudo apt autoclean -y
sudo reboot

# 2. Instalar dependencias básicas
sudo apt install ca-certificates curl git

# 3. Configurar repositorio de Docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/etc/apt/keyrings/docker.asc] \
  https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" \
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# 4. Instalar Docker y Docker Compose
sudo apt update
```

```
sudo apt install docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin

# 5. Configurar usuario para Docker (opcional)
sudo usermod -aG docker $USER
newgrp docker
```

3.2. Descarga del Proyecto

Una vez preparado el servidor, proceder con la descarga del proyecto H2Vital:

```
# 1. Clonar el repositorio
git clone https://github.com/MRamos8/H2Vital.git
cd H2Vital

# 2. Hacer ejecutable el script de despliegue
chmod +x deployment/scripts/deploy.sh

# 3. Crear enlace simbólico para facilitar uso
ln -s deployment/scripts/deploy.sh deploy.sh

# El repositorio ya incluye:
# - Configuración de Docker Compose (docker-compose.yml)
# - Variables de entorno de ejemplo (.env.development, .env.production)
# - Scripts de despliegue automatizado
# - Estructura de directorios completa
```

3.3. Configuración de Variables de Entorno

El repositorio incluye archivos de configuración que solo requieren ajustes menores:

Para Desarrollo:

```
# El archivo .env.development ya está configurado correctamente
# No requiere modificaciones para desarrollo local

# Verificar configuración (opcional):
cat .env.development

# Las variables principales ya están configuradas:
# - NODE_ENV=development
# - DATABASE_URL con PostgreSQL local
# - NEXTAUTH_URL para autenticación
# - Credenciales de PgAdmin
```

Para Producción:

```
# Copiar y personalizar archivo de producción
cp .env.production .env.production.local
```

```
nano .env.production.local

# CAMBIAR OBLIGATORIAMENTE:
# - POSTGRES_PASSWORD: contraseña segura única
# - NEXTAUTH_SECRET: clave secreta de 256 bits
# - NEXT_PUBLIC_APP_URL: dominio real de producción
# - URLs de APIs según el entorno de producción
```

4. Despliegue de la Aplicación

4.1. Script de Despliegue Automatizado

H2Vital incluye un script de despliegue (‘deploy.sh‘) que automatiza todas las operaciones de instalación y mantenimiento:

Comandos Principales:

- **‘./deploy.sh dev‘**: Inicia la aplicación en modo desarrollo
- **‘./deploy.sh prod‘**: Inicia la aplicación en modo producción
- **‘./deploy.sh reset‘**: Reinicia completamente el sistema
- **‘./deploy.sh backup [nombre]‘**: Crea backup de la base de datos
- **‘./deploy.sh health‘**: Verifica el estado del sistema
- **‘./deploy.sh logs‘**: Muestra logs en tiempo real

4.2. Primer Despliegue en Desarrollo

Para realizar el primer despliegue en modo desarrollo:

```
# 1. Iniciar el sistema en modo desarrollo
./deploy.sh dev

# El script automáticamente:
# - Copia las variables de entorno de desarrollo
# - Construye los contenedores Docker
# - Inicia los servicios (app, db, pgadmin)
# - Ejecuta migraciones de Prisma
# - Carga datos iniciales (seed)

# 2. Verificar que todos los servicios estén funcionando
./deploy.sh health

# 3. Acceder a la aplicación
# - Aplicación principal: http://localhost:3000
```

```
# - PgAdmin: http://localhost:5050
# - Prisma Studio: http://localhost:5555
```

4.3. Despliegue en Producción

Para el despliegue en producción, seguir estos pasos adicionales:

```
# 1. Configurar variables de producción
cp .env.production .env.production.local
nano .env.production.local

# 2. Cambiar valores críticos:
# - POSTGRES_PASSWORD: contraseña segura única
# - NEXTAUTH_SECRET: clave secreta de 256 bits
# - NEXT_PUBLIC_APP_URL: dominio real de producción

# 3. Configurar certificados SSL (si es necesario)
mkdir -p deployment/certs
# Copiar certificados SSL a deployment/certs/

# 4. Crear backup antes del despliegue
./deploy.sh backup pre_produccion_$(date +%Y%m%d)

# 5. Desplegar en producción
./deploy.sh prod

# 6. Verificar despliegue
./deploy.sh health
./deploy.sh security-check
```

5. Administración del Sistema

5.1. Gestión de Base de Datos

El sistema incluye herramientas para la administración completa de la base de datos:
PgAdmin (Interfaz Web):

- URL: `http://localhost:5050` (solo en desarrollo)
- Usuario: `admin@h2vital.com`
- Contraseña: `h2vital_admin`
- Funciones: consultas SQL, backup/restore, monitoreo

Prisma Studio (Interfaz de Datos):

```
# Iniciar Prisma Studio
docker compose exec app npx prisma studio

# Acceso: http://localhost:5555
# Funciones: visualización de datos, edición de registros
```

Comandos de Base de Datos:

```
# Ejecutar migraciones
./deploy.sh migrate

# Ver estado de migraciones
docker compose exec app npx prisma migrate status

# Generar cliente Prisma
docker compose exec app npx prisma generate

# Ejecutar seed de datos
./deploy.sh seed-only

# Reset completo de base de datos (CUIDADO: elimina datos)
docker compose exec app npx prisma migrate reset
```

5.2. Sistema de Backups

H2Vital incluye un sistema automatizado de backups:

```
# Crear backup manual
./deploy.sh backup nombre_backup

# Listar backups disponibles
./deploy.sh list-backups

# Restaurar desde backup
./deploy.sh restore nombre_backup

# Backup automático antes de operaciones críticas
./deploy.sh backup pre_update_$(date +%Y%m%d_%H%M)

# Limpieza de backups antiguos (mantiene últimos 5)
./deploy.sh cleanup
```

5.3. Monitoreo y Logs

El sistema proporciona herramientas completas de monitoreo:

```
# Logs en tiempo real
./deploy.sh logs
```

```
3
4  # Logs de servicio específico
5  docker compose logs -f app
6  docker compose logs -f db
7
8  # Monitoreo de salud del sistema
9  ./deploy.sh health
10
11 # Monitoreo continuo (actualización cada 5 segundos)
12 ./deploy.sh monitor
13
14 # Verificación de estado de contenedores
15 docker compose ps
16
17 # Uso de recursos
18 docker stats
```

6. Manual de Usuario

6.1. Acceso al Sistema

El acceso al sistema H2Vital se realiza a través del navegador web utilizando credenciales proporcionadas por el administrador:

Proceso de Login:

1. Acceder a la URL del sistema (ej: `http://localhost:3000`)
2. Introducir nombre de usuario y contraseña
3. Hacer clic en Ïniciar Sesión"
4. El sistema redirigirá al dashboard principal según el rol del usuario

Credenciales Iniciales (Desarrollo):

- Usuario administrador: `admin`
- Contraseña: `admin123`
- Rol: Gerente (acceso completo al sistema)

6.2. Navegación en el Sistema

La interfaz de H2Vital está diseñada para ser intuitiva y accesible:

Elementos de Navegación:

- **Sidebar Izquierdo**: Menú principal con acceso a todos los módulos

- **Header Superior**: Controles de usuario, notificaciones y configuración
- **Breadcrumbs**: Navegación contextual que muestra la ubicación actual
- **Botones de Acción**: Claramente identificados con iconos y texto descriptivo

Módulos Principales:

- **Inicio**: Dashboard con resumen de actividad y métricas
- **Empleados**: Gestión de personal y roles
- **Papeletas y Referencias**: Gestión de leads comerciales
- **Lista de Citas**: Programación y seguimiento de citas comerciales
- **Lista de Clientes**: Administración de información de clientes
- **Máquinas**: Catálogo de productos y equipos instalados
- **Contratos**: Gestión de contratos de venta y financiación
- **Mi Cuenta**: Configuración personal del usuario

6.3. Gestión de Empleados

El módulo de empleados permite administrar toda la información del personal:
Funciones Disponibles:

1. **Ver Lista de Empleados**: Tabla con filtros por puesto y estado
2. **Añadir Nuevo Empleado**: Formulario completo con validación
3. **Editar Empleado**: Modificación de datos personales y laborales
4. **Gestionar Credenciales**: Creación y modificación de accesos al sistema
5. **Cambiar Estados**: Alta, Baja, Pendiente con fechas automáticas

Campos de Información:

- Datos personales: Nombre, apellidos, DNI, teléfono, email
- Información laboral: Puesto, oficina, fecha de alta, estado
- Avatar personalizable con iniciales automáticas
- Credenciales de acceso opcionales según el puesto

6.4. Gestión de Papeletas y Referencias

Sistema para manejar los contactos comerciales iniciales:
Diferencias entre Papeletas y Referencias:

- **Papeletas**: Contactos directos que requieren telefonista y azafata
- **Referencias**: Contactos referenciales que solo requieren telefonista

Estados Disponibles:

- **NUEVA**: Papeleta recién creada sin procesar
- **RD**: Reagendada (nueva fecha de contacto)
- **NC**: No contesta (múltiples intentos fallidos)
- **NP**: No procede (no interesado)
- **NQ**: No quiere (rechaza explícitamente)
- **CONCERTADA**: Cita acordada (lista para conversión)
- **BOLA**: Información falsa o incorrecta
- **ERRONEA**: Error en los datos

6.5. Gestión de Citas Comerciales

Módulo para programar y dar seguimiento a las visitas comerciales:
Proceso de Gestión de Citas:

1. **Creación desde Papeleta**: Conversión automática de papeletas concertadas
2. **Asignación de Comerciales**: Principal y secundario para trabajo en equipo
3. **Programación**: Fecha, hora y dirección de la cita
4. **Seguimiento**: Estados desde programada hasta venta o no venta
5. **Generación de Contrato**: Conversión automática de ventas exitosas

Estados de Cita:

- **Sin programar**: Cita creada pero sin fecha asignada
- **Programada**: Fecha y hora confirmadas
- **Venta**: Cita exitosa que genera contrato
- **No Venta**: Cita realizada sin conversión
- **Cancelada**: Cita anulada por cualquier motivo

6.6. Gestión de Clientes

Sistema centralizado para administrar la información de clientes:
Información de Cliente:

- **Datos Personales**: Nombre, DNI, teléfono, email
- **Direcciones Múltiples**: Casa, trabajo, entrega (con marcador principal)
- **Contactos Asociados**: Familiares o personas relacionadas
- **Historial**: Papeletas, citas y contratos asociados

Funciones Disponibles:

1. Añadir nuevas direcciones con formulario modal
2. Marcar dirección principal para envíos
3. Gestionar contactos asociados (familiares, referencias)
4. Visualizar historial completo de interacciones
5. Acceso directo a contratos activos

6.7. Gestión de Contratos

Administración completa de contratos de venta:
Información de Contrato:

- **Datos Básicos**: Número único, fecha, cliente titular
- **Máquina Asignada**: Modelo específico con número de serie
- **Financiación**: Modalidad, cuotas, precios, entidades bancarias
- **Instalación**: Dirección, fechas, estado de instalación
- **Estados**: Activo, En Incidencia, Fallido

Alertas del Sistema:

- Instalaciones pendientes con avisos visuales
- Contratos en incidencia que requieren atención
- Fechas de vencimiento de promociones
- Validación automática de datos financieros

6.8. Gestión de Máquinas

Catálogo de productos organizizado por categorías:
Categorías Disponibles:

- **Agua**: Equipos de tratamiento y purificación
- **Aire**: Purificadores y sistemas de ventilación
- **Descanso**: Colchones y sistemas de descanso
- **Hogar**: Productos diversos para el hogar
- **FilterQueen**: Línea específica de aspiradoras

Estados de Máquina:

- **Activa**: Disponible para venta e instalación
- **Obsoleta**: Descatalogada, solo para mantenimiento

7. Configuración de Usuario

7.1. Mi Cuenta - Configuración Personal

Cada usuario puede personalizar su experiencia en el sistema:
Opciones de Perfil:

- **Foto de Avatar**: Carga de imagen personal (JPG, PNG, GIF, WebP, máx. 5MB)
- **Datos Personales**: Nombre, apellidos, email, teléfono
- **Configuración de Cuenta**: Cambio de contraseña y configuraciones de seguridad
- **Historial de Accesos**: Registro de inicios de sesión con IP y ubicación

Limitaciones y Validaciones:

- Validación en tiempo real de campos de email y teléfono
- Restricciones de tamaño y formato para imágenes de perfil
- Confirmación requerida para cambios críticos
- Mantiene trazabilidad de cambios para auditoría

8. Solución de Problemas

8.1. Problemas Comunes de Instalación

Error: "Docker no está ejecutándose"

```
# Verificar estado de Docker
sudo systemctl status docker

# Iniciar Docker si está detenido
sudo systemctl start docker
sudo systemctl enable docker

# Verificar instalación
docker --version
docker compose version
```

Error: "Base de datos no disponible"

```
# Verificar estado de contenedores
docker compose ps

# Ver logs de base de datos
docker compose logs db

# Reiniciar servicio de base de datos
docker compose restart db

# Verificar conectividad
docker compose exec db pg_isready -U h2vital
```

Error: "Migraciones de Prisma fallan"

```
# Ver estado actual de migraciones
docker compose exec app npx prisma migrate status

# Ejecutar migraciones manualmente
./deploy.sh migrate

# Si persiste el error, reset de migraciones (CUIDADO)
docker compose exec app npx prisma migrate reset

# Volver a ejecutar seed
./deploy.sh seed-only
```

8.2. Problemas de Rendimiento

Aplicación lenta o no responde:

1. Verificar uso de recursos: `docker stats`

2. Revisar logs por errores: `./deploy.sh logs`
3. Reiniciar contenedores: `docker compose restart`
4. Verificar espacio en disco: `df -h`

Base de datos lenta:

1. Verificar conexiones activas en PgAdmin
2. Analizar consultas lentas en logs
3. Considerar optimización de índices
4. Verificar integridad de datos: `VACUUM ANALYZE`

8.3. Problemas de Acceso y Autenticación

No se puede iniciar sesión:

1. Verificar credenciales en base de datos
2. Comprobar configuración NEXTAUTH_SECRET
3. Revisar logs de autenticación
4. Verificar estado de sesiones en base de datos

Crear usuario administrador de emergencia:

```
# Acceder a contenedor de aplicación
docker compose exec app bash

# Ejecutar script de creación de usuario admin
npx prisma db seed

# O crear directamente en base de datos
docker compose exec db psql -U h2vital -d h2vital_dev
```

9. Mantenimiento del Sistema

9.1. Rutinas de Mantenimiento

Mantenimiento Diario:

- Verificar estado del sistema: `./deploy.sh health`
- Revisar logs por errores: `./deploy.sh logs`

- Monitorear uso de espacio en disco

Mantenimiento Semanal:

- Crear backup completo: `./deploy.sh backup weekly_$(date +%Y%m%d)`
- Limpiar backups antiguos: `./deploy.sh cleanup`
- Actualizar sistema operativo: `sudo apt update && sudo apt upgrade`

Mantenimiento Mensual:

- Revisar y actualizar certificados SSL
- Analizar métricas de rendimiento
- Verificar integridad de base de datos
- Actualizar documentación de configuración

9.2. Actualizaciones del Sistema

Proceso de Actualización:

```
# 1. Crear backup antes de actualizar
./deploy.sh backup pre_update_$(date +%Y%m%d)

# 2. Actualizar código del repositorio
git pull origin main

# 3. Validar configuración
./deploy.sh validate

# 4. Aplicar actualizaciones
./deploy.sh reset

# 5. Verificar funcionamiento
./deploy.sh health

# 6. Verificar aplicación manualmente
# Acceder a interfaz web y probar funcionalidades críticas
```

9.3. Seguridad y Auditoría

Verificaciones de Seguridad:

- Revisar logs de acceso regularmente
- Verificar permisos de usuarios y roles

- Monitorear intentos de acceso fallidos
- Mantener actualizadas las dependencias

Backup y Recuperación:

- Backups automáticos antes de operaciones críticas
- Pruebas periódicas de restauración
- Almacenamiento de backups en ubicación segura
- Documentación de procedimientos de recuperación

10. Soporte y Contacto

10.1. Recursos de Ayuda

Para obtener soporte adicional con el sistema H2Vital:
Documentación Técnica:

- Manual de instalación (este documento)
- Anexo de capturas de pantalla para referencia visual
- Documentación de API en línea
- Esquema de base de datos actualizado

Herramientas de Autodiagnóstico:

- `./deploy.sh health`: Verificación automática del sistema
- `./deploy.sh validate`: Validación de configuración
- `./deploy.sh monitor`: Monitoreo en tiempo real
- Logs detallados en `./logs/` para análisis

10.2. Información de Versión

Sistema H2Vital:

- Versión: 1.0.0
- Framework: Next.js 15
- Base de datos: PostgreSQL 15

- ORM: Prisma 5.x
- UI Library: Material MUI 5.x
- Contenedores: Docker Compose

Compatibilidad:

- Navegadores: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- Sistemas operativos: Ubuntu 22.04+, Windows 10+, macOS 12+
- Dispositivos: Desktop, tablet, móvil (responsive design)

Esta documentación proporciona una guía completa para la instalación, configuración y uso del sistema H2Vital. Para casos específicos no cubiertos en este manual, se recomienda consultar los logs del sistema y utilizar las herramientas de diagnóstico incluidas.

Capturas de Pantalla de la Aplicación H2Vital

1. Introducción

Este anexo presenta una demostración visual completa del sistema H2Vital desarrollado, mostrando todas las interfaces de usuario implementadas y las funcionalidades principales de cada módulo. Las capturas evidencian la materialización exitosa del diseño basado en Material MUI y la experiencia de usuario optimizada para los diferentes perfiles de empleados de H2Vital.

La aplicación implementa un diseño responsive y consistente que facilita la navegación intuitiva entre módulos, manteniendo la coherencia visual y funcional en todas las pantallas. Cada interfaz ha sido diseñada siguiendo principios de usabilidad para usuarios no técnicos, con feedback visual inmediato y validaciones en tiempo real.

2. Autenticación y Acceso al Sistema

2.1. Pantalla de Inicio de Sesión

La interfaz de login implementa el branding corporativo de H2Vital con un diseño profesional que transmite confianza y calidad. El formulario incluye validación en tiempo real y manejo de errores.

Figura 1: Pantalla de inicio de sesión con branding corporativo H2Vital. Muestra el formulario de autenticación con validación de credenciales y diseño responsive.

Características implementadas:

- Diseño corporativo con imagen promocional de la empresa
- Validación de credenciales con feedback visual inmediato
- Formulario responsive adaptado a diferentes dispositivos
- Opción R̈ecordarmeẗpara sesiones persistentes

3. Gestión de Empleados

3.1. Lista Principal de Empleados

La vista principal de empleados proporciona una interfaz completa para la administración del personal, con filtros avanzados por puesto y estado, búsqueda en tiempo real y operaciones CRUD completas.

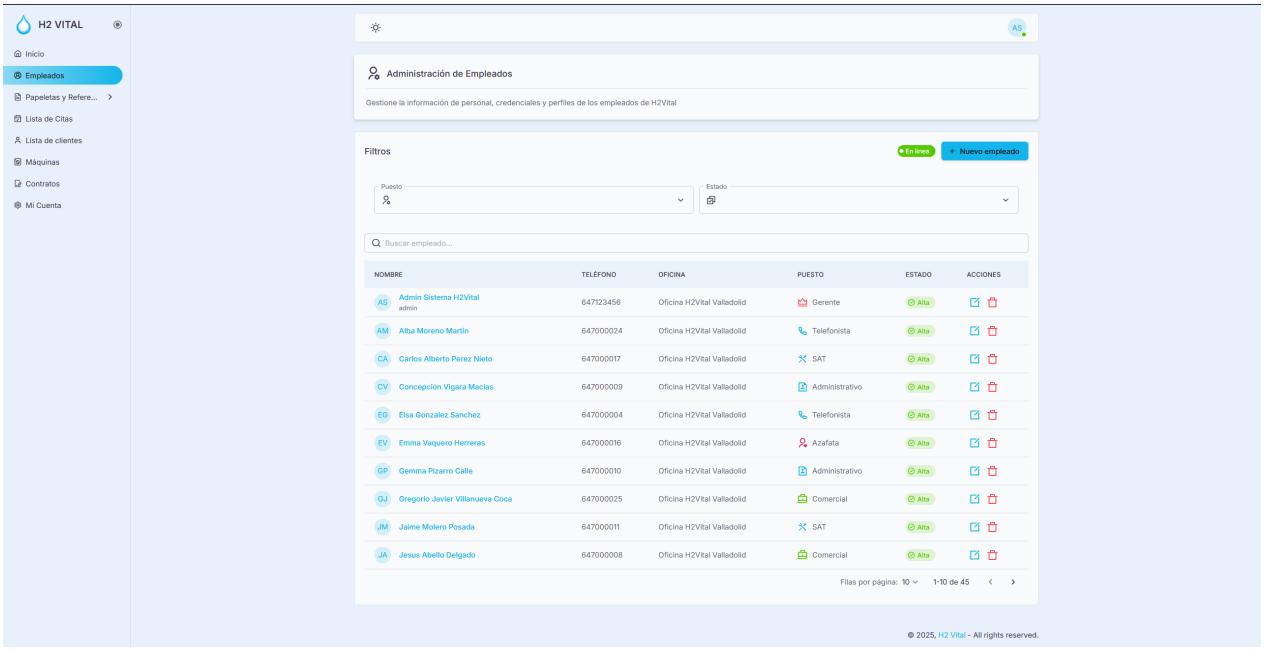


Figura 2: Vista principal de administración de empleados. Incluye filtros por puesto y estado, búsqueda avanzada, y gestión completa del personal con indicadores visuales de roles y estados.

Funcionalidades destacadas:

- Filtros específicos por puesto (Gerente, SAT, Comercial, etc.) y estado (Alta/Baja)
- Búsqueda en tiempo real por nombre o información del empleado
- Indicadores visuales de roles con iconografía específica
- Chips de estado con código de colores para identificación rápida
- Botones de acción para editar y eliminar empleados
- Paginación inteligente con selección de elementos por página
- Botón de creación de nuevo empleado destacado visualmente

3.2. Perfil de Empleado y Gestión de Credenciales

La vista de perfil de empleado integra información personal, métricas de actividad y gestión completa de credenciales de acceso, proporcionando una visión integral del empleado en el sistema.

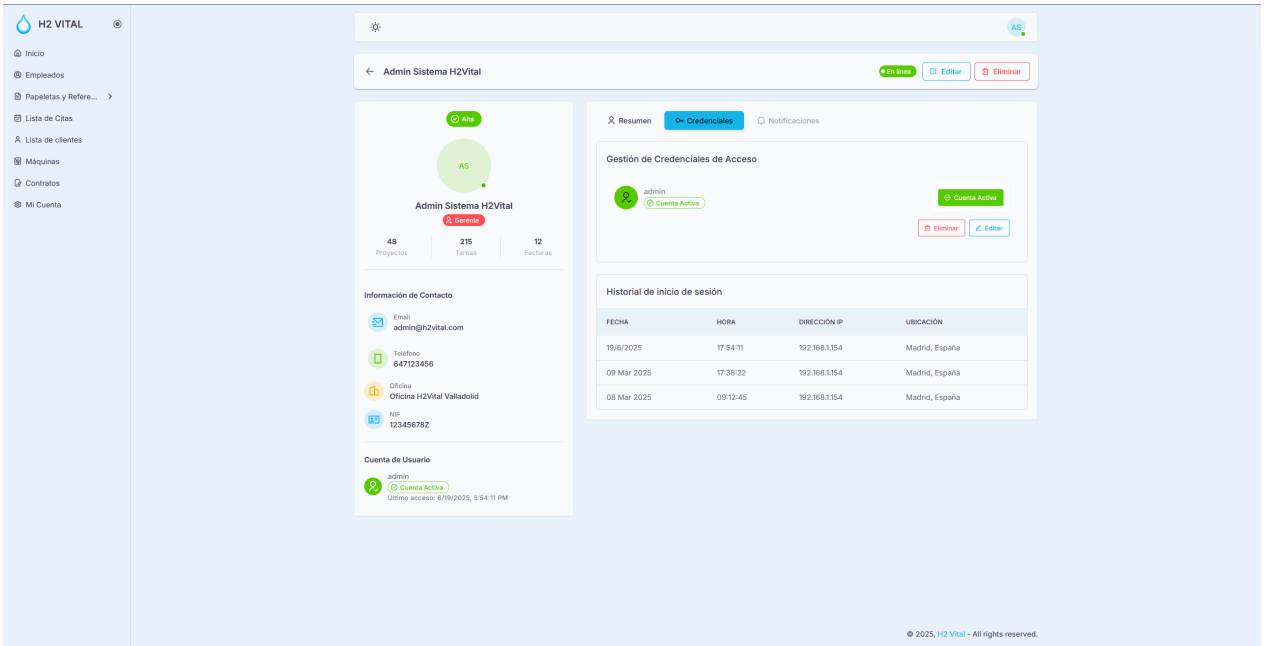


Figura 3: Perfil detallado de empleado Admin Sistema H2Vital. Muestra gestión de credenciales, métricas de actividad, historial de accesos y configuración de cuenta con indicadores de estado.

Elementos del perfil:

- Avatar personalizable con iniciales automáticas
- Indicadores de rol (Gerente) y estado (Alta) con chips distintivos
- Información de contacto completa con iconografía descriptiva
- Gestión de credenciales de acceso con estado de cuenta activa
- Historial detallado de inicios de sesión con IP y ubicación

4. Gestión de Papeletas y Referencias

4.1. Lista de Papeletas

La administración de papeletas implementa un sistema completo de filtros y búsquedas que permite gestionar eficientemente los leads comerciales con asignación de personal especializado.

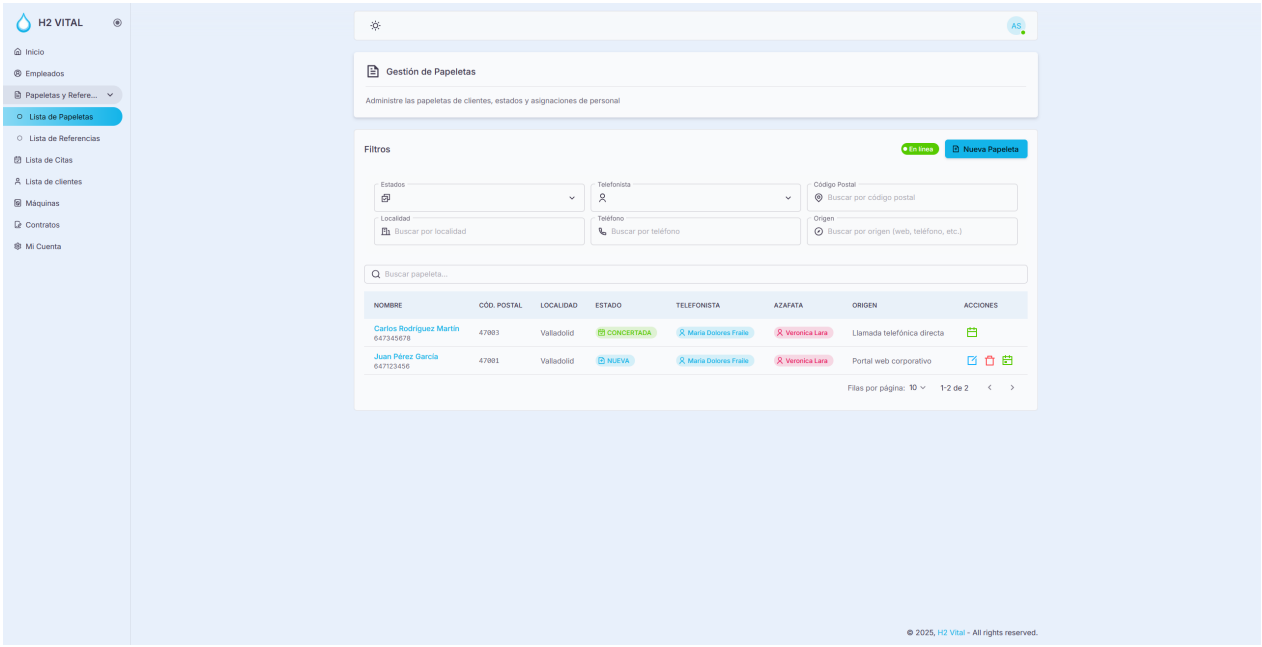


Figura 4: Administración de papeletas con filtros avanzados por estado, telefonista, localidad y origen. Muestra asignación de personal y estados del proceso comercial.

4.2. Lista de Referencias

El módulo de referencias maneja contactos potenciales con un flujo similar a papeletas pero diferenciado funcionalmente, permitiendo seguimiento específico de referencias comerciales.

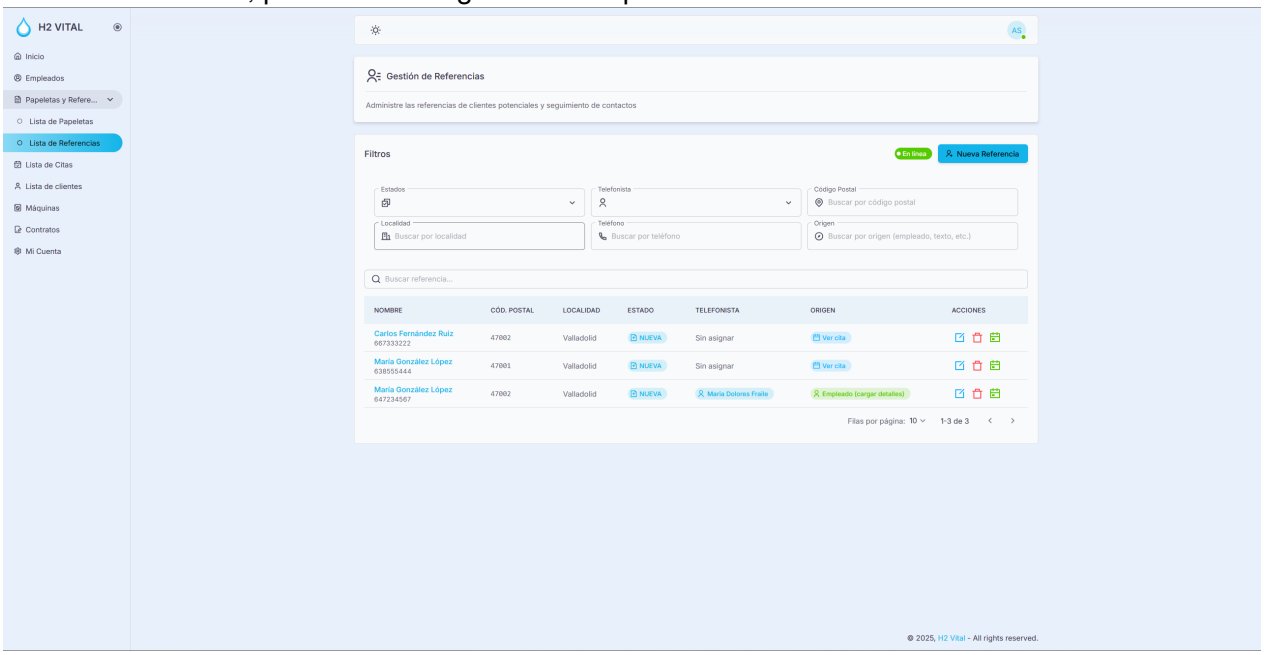


Figura 5: Gestión de referencias de clientes potenciales. Sistema de filtros similar a papeletas con seguimiento específico para contactos referenciales.

4.3. Detalles de Papeleta

La vista de detalle de papeleta centraliza toda la información del contacto comercial con asignación visual de personal responsable y seguimiento de estados.

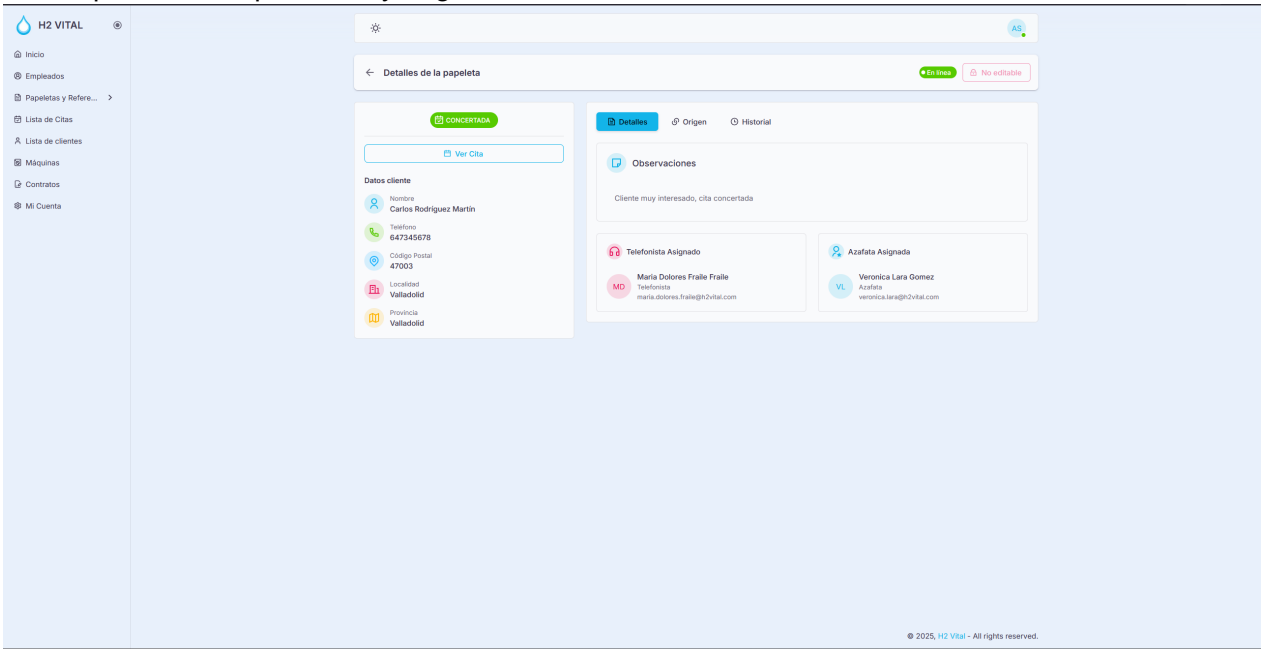


Figura 6: Vista detallada de papeleta en estado CONCERTADA. Muestra datos completos del cliente, asignación de telefonista y azafata, observaciones y botón de conversión a cita.

4.4. Detalles de Referencia

La interfaz de referencia mantiene coherencia visual con papeletas pero implementa flujos específicos para el manejo de contactos referenciales.

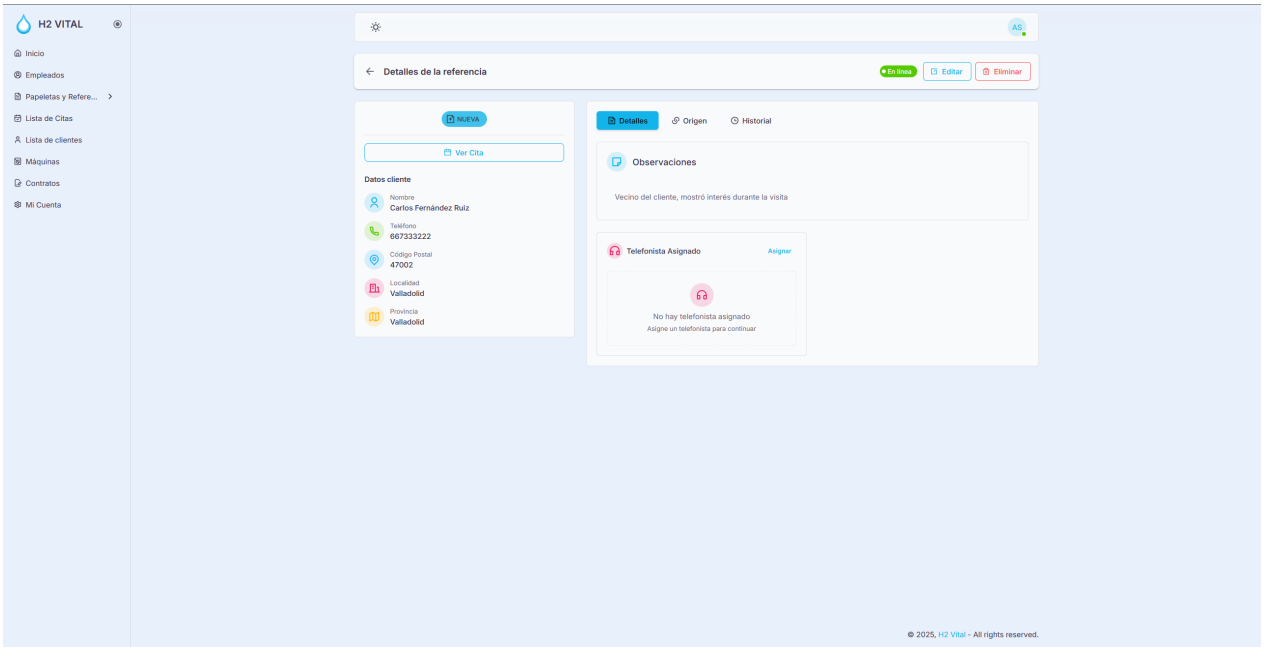


Figura 7: Detalle de referencia en estado NUEVA. Interfaz similar a papeletas con funcionalidades específicas para gestión de referencias y asignación pendiente de telefonista.

Características comunes de papeletas y referencias:

- Datos completos del cliente con validación de campos españoles
- Sistema de estados visuales con chips de color semántico
- Asignación clara de telefonista y azafata responsables
- Observaciones editables para seguimiento comercial
- Botones de acción contextual según el estado actual
- Integración con módulo de citas para conversión automática

5. Gestión de Citas Comerciales

5.1. Lista de Citas

El módulo de citas implementa un sistema completo de gestión comercial con asignación de equipos de ventas, programación temporal y seguimiento de estados hasta la conversión.

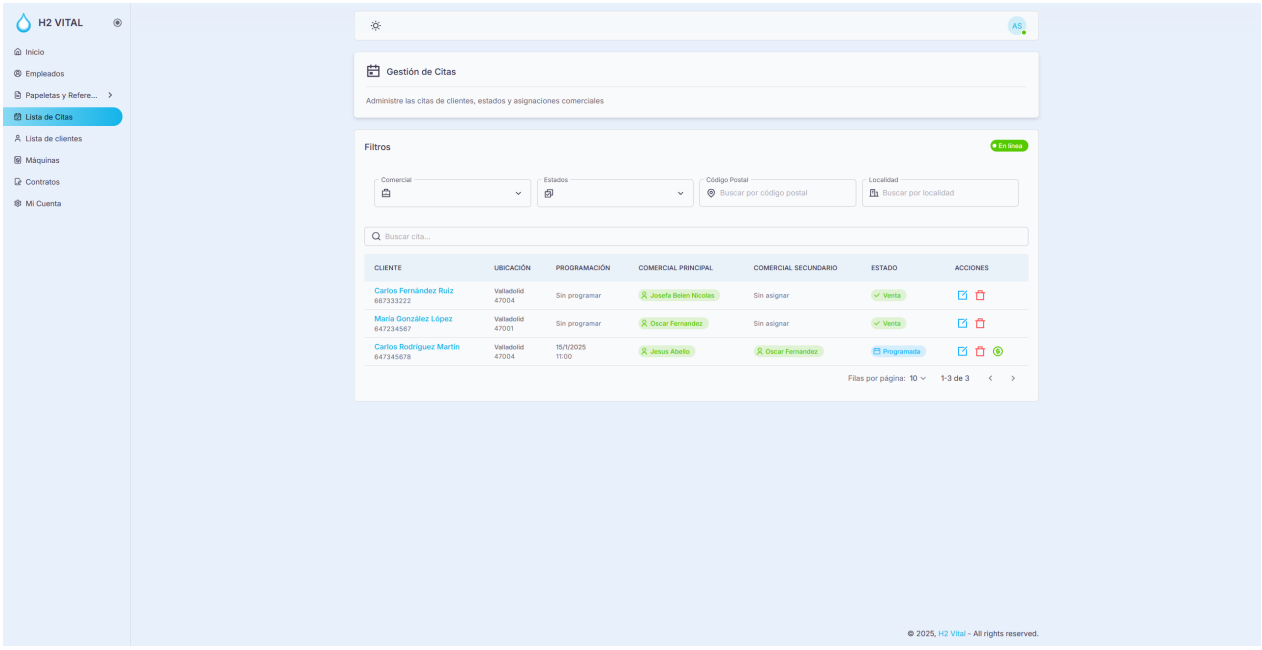


Figura 8: Gestión de citas comerciales con filtros por comercial, estados y localización. Muestra programación, asignación de equipo comercial y estados de seguimiento.

Funcionalidades de gestión de citas:

- Filtros avanzados por comercial, estados, código postal y localidad
- Programación temporal visible con fechas y horarios específicos
- Asignación de comercial principal y secundario para trabajo en equipo
- Estados semánticos: Sin programar, Programada, Venta, etc.
- Información de contacto del cliente integrada
- Botones de acción para editar y gestionar citas

5.2. Detalles de Cita

La vista de detalle de cita proporciona información completa del contacto comercial con gestión de programación y seguimiento de referencias originales.

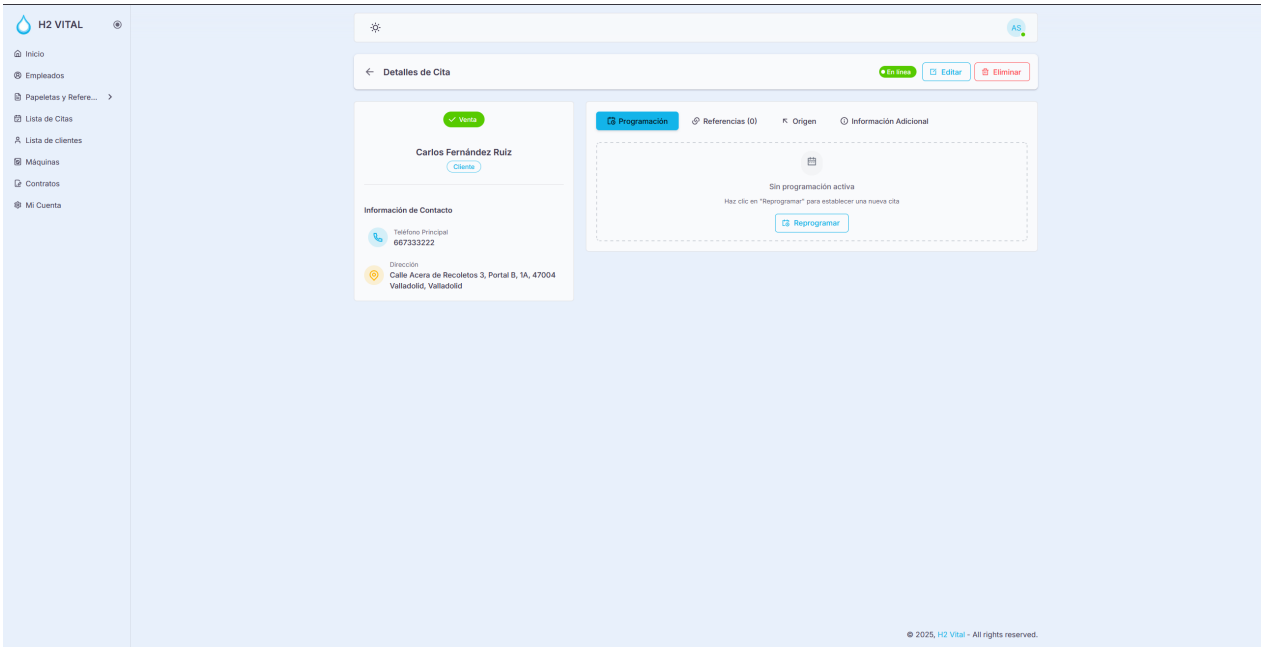


Figura 9: Detalle de cita comercial en estado VENTA. Muestra información completa del cliente, gestión de programación, referencias asociadas e información adicional del proceso comercial.

Elementos del detalle de cita:

- Estado prominente con chip V̈ENTAïndicando éxito comercial
- Información completa de contacto del cliente
- Pestañas organizadas: Programación, Referencias, Origen e Información Adicional
- Herramientas de reprogramación con botón específico
- Integración con sistema de referencias para trazabilidad
- Botones de acción contextual para editar y eliminar

6. Gestión de Clientes

6.1. Lista de Clientes

El módulo de clientes centraliza la información de todos los contactos con funcionalidades de búsqueda avanzada y métricas de relación comercial.

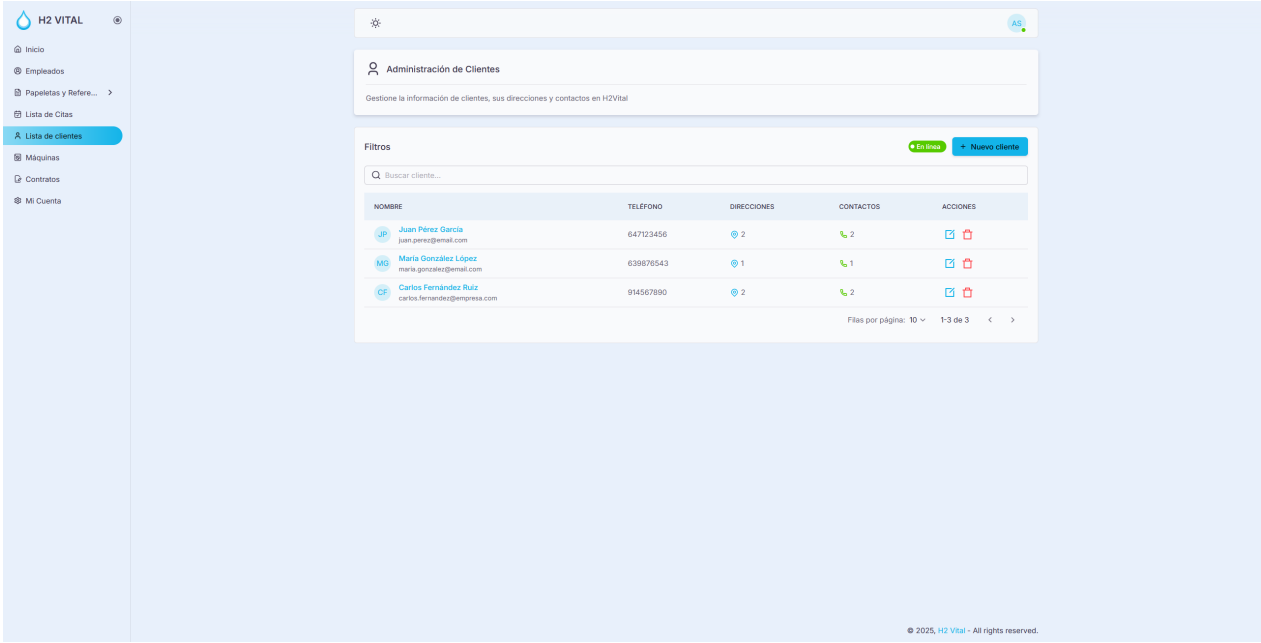


Figura 10: Administración de clientes de H2Vital. Vista consolidada con información de contacto, múltiples direcciones, contratos asociados y herramientas de búsqueda avanzada.

6.2. Detalles de Cliente

La vista de cliente integra gestión completa de información personal, direcciones múltiples y contactos asociados con funcionalidades de edición in-situ.

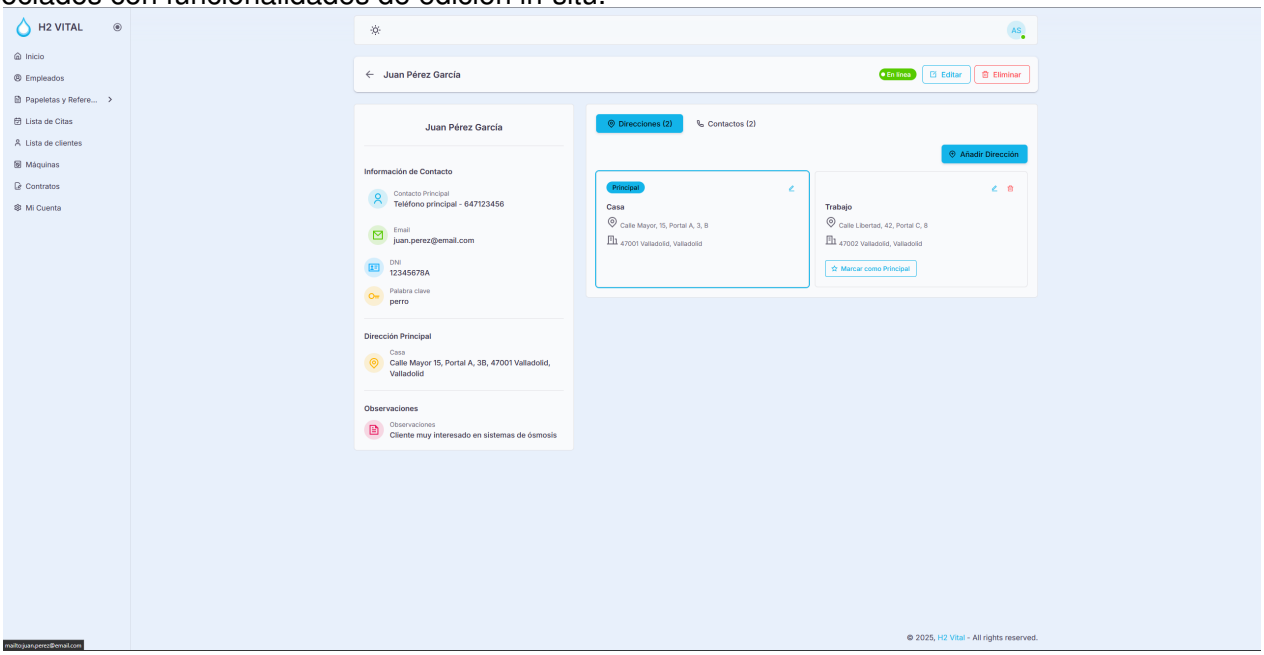


Figura 11: Perfil completo de cliente Juan Pérez García. Gestión de información de contacto, direcciones múltiples (casa/trabajo), contactos asociados y funcionalidad de añadir nuevas direcciones.

Características del perfil de cliente:

- Información de contacto completa con datos validados

- Gestión de múltiples direcciones (Principal/Casa/Trabajo)
- Pestañas organizadas para direcciones y contactos
- Funcionalidad Äñadir Direcciónc̈on formulario modal
- Marcadores de dirección principal y de trabajo
- Botones de acción para marcar como principal cada dirección
- Información detallada con códigos postales y ciudades

7. Gestión de Contratos

7.1. Lista de Contratos

El módulo de contratos implementa un sistema completo de administración de ventas con información financiera, equipos instalados y seguimiento de estados.

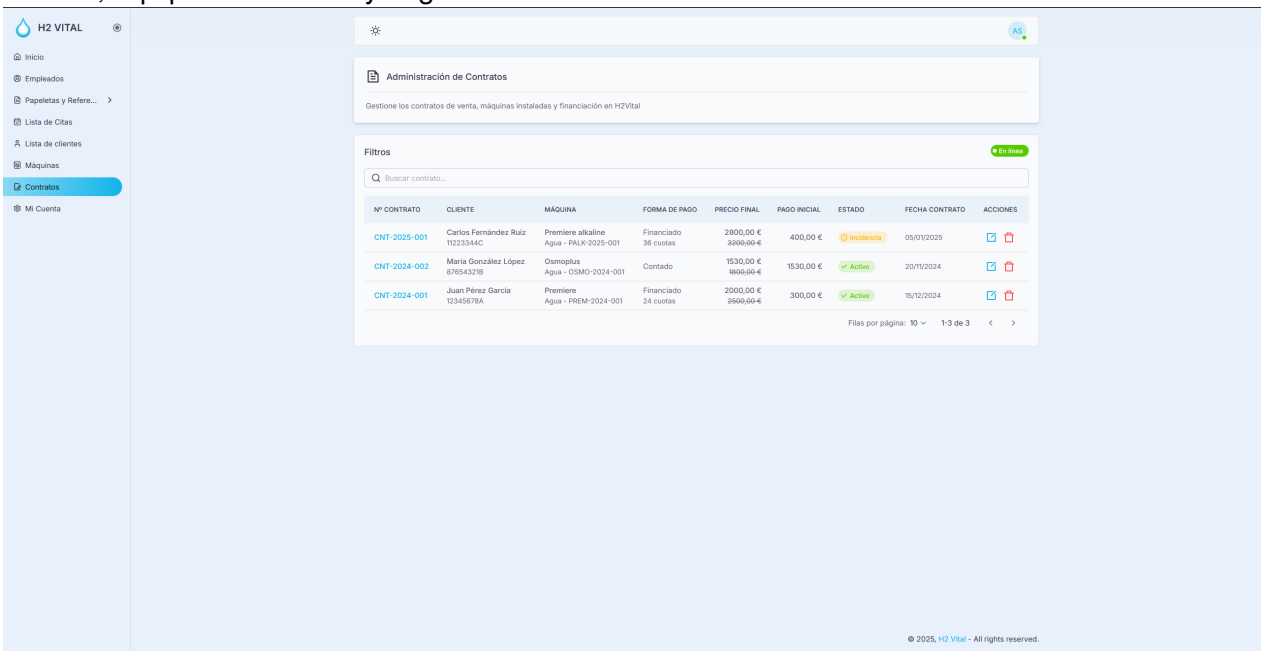


Figura 12: Administración de contratos de venta con información completa: números de contrato, clientes, máquinas instaladas, modalidades de financiación, precios y estados.

7.2. Detalles de Contrato

La vista de contrato centraliza información comercial, técnica y financiera con seguimiento de instalación y alertas de estado.

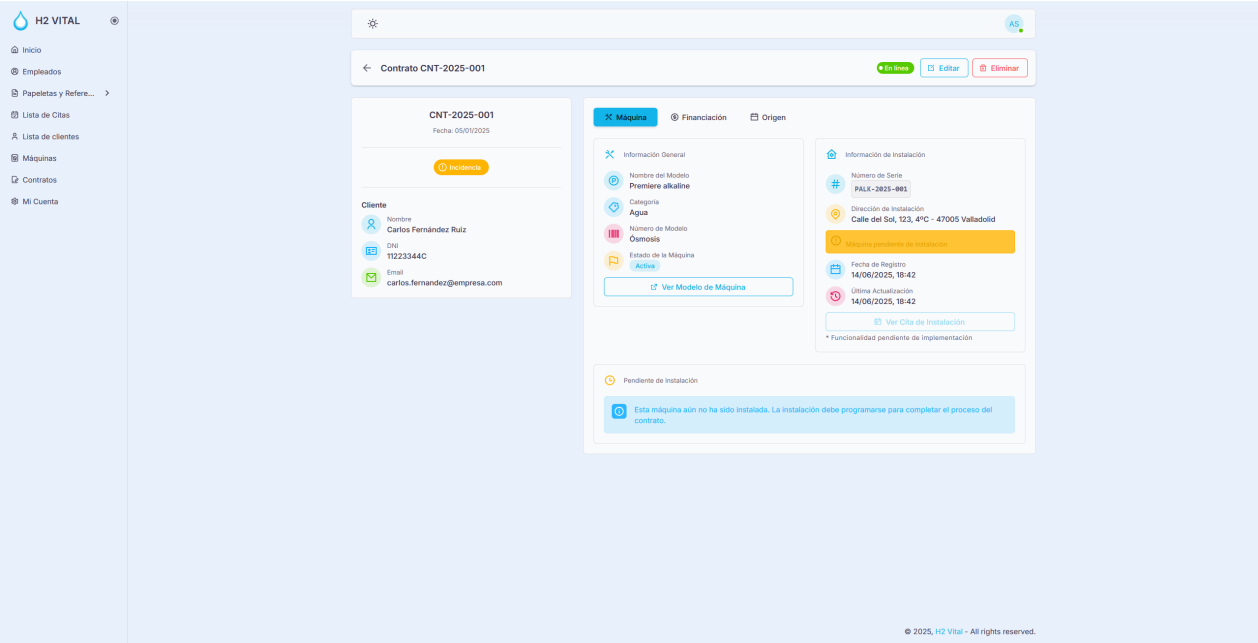


Figura 13: Detalle del contrato CNT-2025-001 en estado .En Incidencia". Muestra información del cliente, máquina asignada (Premiere alkaline), datos de instalación y alerta de pendiente de instalación.

Elementos del contrato:

- Número de contrato único con fecha de emisión
- Estado visual Ën Incidenciaëon chip de alerta
- Información completa del cliente titular
- Pestañas organizadas: Máquina, Financiación y Origen
- Detalles de máquina: modelo, categoría, número de serie
- Información de instalación con dirección específica
- Alertas visuales para instalaciones pendientes
- Fechas de registro y última actualización

8. Gestión de Máquinas

8.1. Lista de Máquinas

El catálogo de máquinas organiza el inventario de productos por categorías con filtros específicos y información de modelos disponibles.

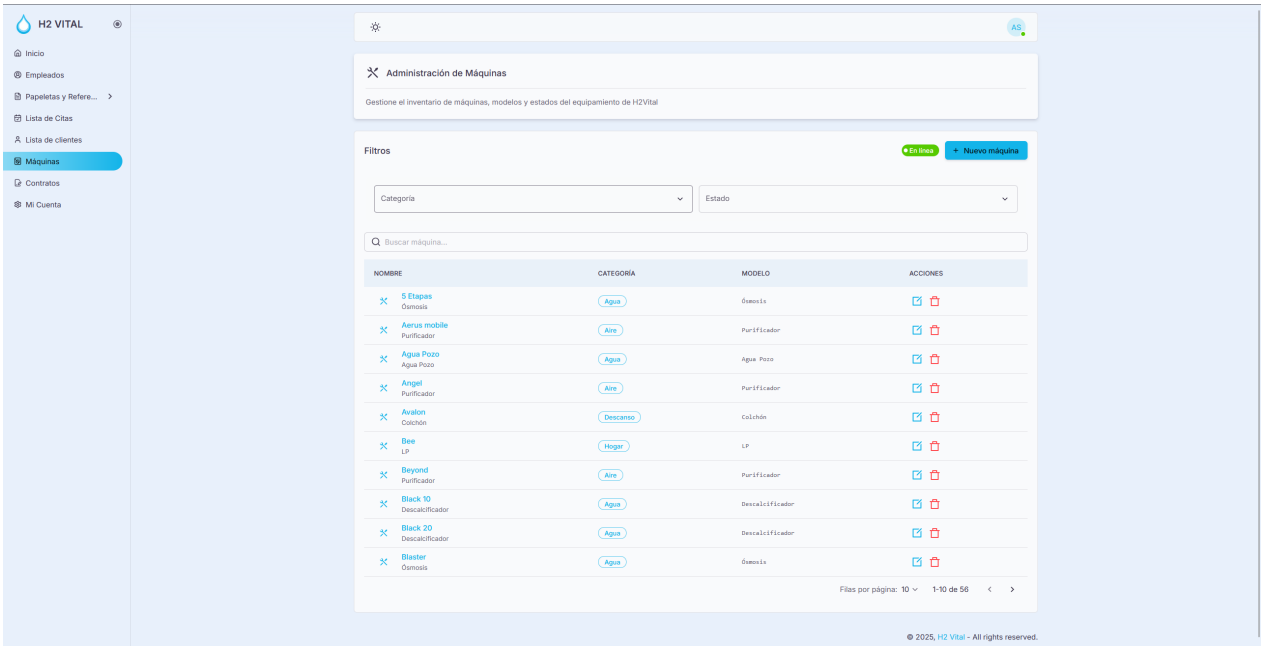


Figura 14: Administración del inventario de máquinas H2Vital. Catálogo organizado por categorías (Agua, Aire, Descanso, Hogar) con modelos específicos y filtros de estado.

8.2. Detalles de Máquina

La vista de máquina proporciona información técnica completa del equipo con especificaciones y estado de disponibilidad.

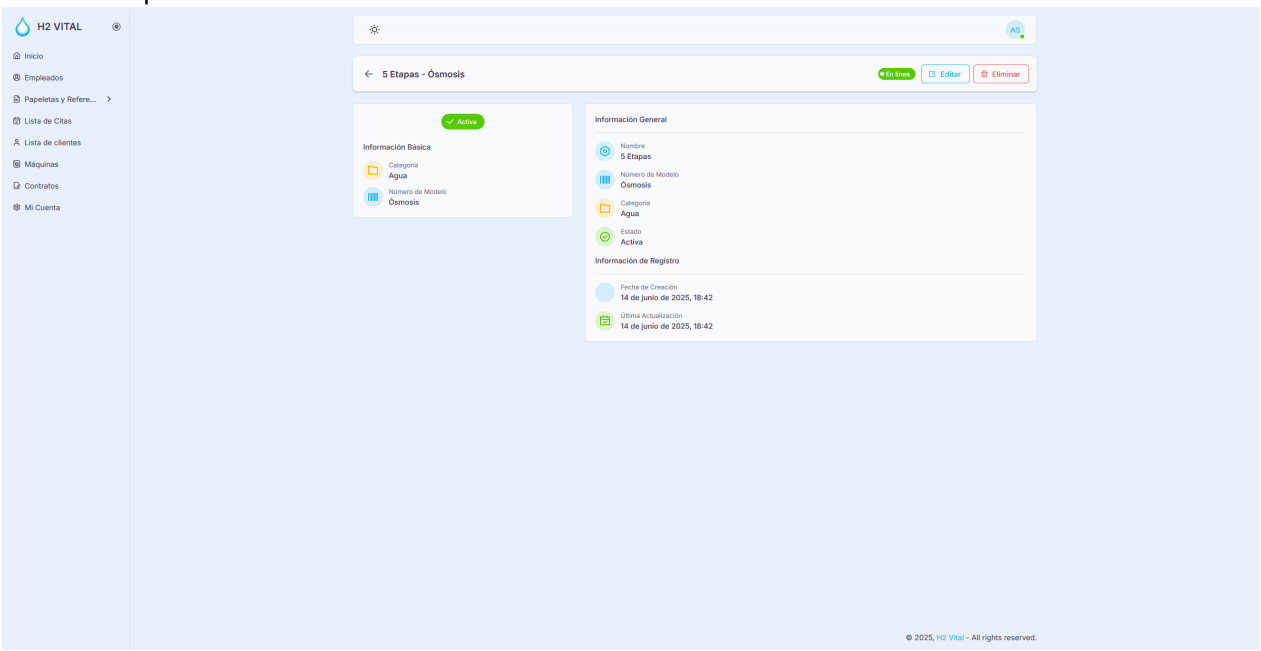


Figura 15: Ficha técnica de la máquina "5 Etapas - Osmosis". Información general del modelo, categoría Agua, número de modelo y estado activo en el catálogo.

Características del catálogo de máquinas:

- Organización por categorías: Agua, Aire, Descanso, Hogar

- Filtros por categoría y estado (Activa/Obsoleta)
- Información de modelos con números específicos
- Estados visuales para control de inventario
- Búsqueda por nombre de máquina
- Información técnica detallada por equipo
- Fechas de creación y actualización de registros

9. Configuración de Cuenta de Usuario

9.1. Mi Cuenta - Perfil Personal

La sección de cuenta personal permite a cada usuario gestionar su información, foto de perfil y configuraciones personales del sistema.

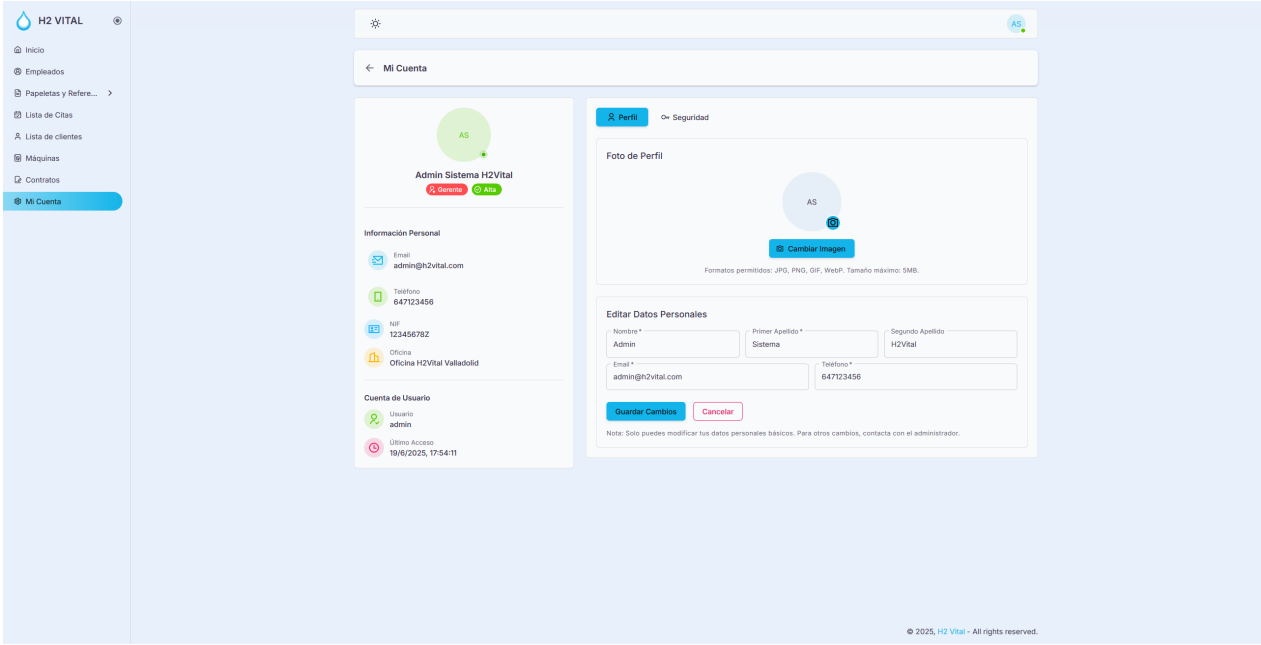


Figura 16: Configuración de cuenta personal del Admin Sistema H2Vital. Gestión de perfil, foto de avatar, datos personales editables y configuración de seguridad de la cuenta.

Funcionalidades de Mi Cuenta:

- Avatar personalizable con carga de imagen (JPG, PNG, GIF, WebP)
- Formulario de edición de datos personales en tiempo real
- Campos validados: nombre, apellidos, email y teléfono
- Información de cuenta de usuario y último acceso

- Pestañas organizadas: Perfil y Seguridad
- Botones de acción: Guardar Cambios y Cancelar
- Limitaciones de archivo claramente especificadas
- Integración con sistema de roles y permisos

10. Características Transversales del Sistema

10.1. Elementos de Interfaz Comunes

Todas las pantallas de H2Vital implementan elementos de diseño consistentes que garantizan una experiencia de usuario coherente:

Navegación y Layout:

- Sidebar izquierdo con menú de módulos y estado de conexión
- Header superior con controles de tema (claro/oscuro) y perfil de usuario
- Breadcrumbs automáticos para navegación contextual
- Logo y branding corporativo H2Vital en todas las pantallas

Filtros y Búsquedas:

- Filtros específicos por módulo con selectores intuitivos
- Búsqueda en tiempo real con debounce para optimización
- Campos de búsqueda contextual según el tipo de datos
- Indicadores visuales de filtros activos

Tablas y Listados:

- Paginación inteligente con selección de elementos por página
- Ordenación por columnas con indicadores visuales
- Botones de acción contextual (editar, eliminar, ver)
- Estados visuales con chips de color semántico
- Información de paginación detallada (ej: 1̈-3 de 3̈)

Sistema de Estados:

- Chips de color verde para estados activos/positivos

- Chips azules para estados en proceso o informativos
- Chips naranjas para advertencias o incidencias
- Chips rojos para estados críticos o errores
- Iconografía específica para cada tipo de elemento

10.2. Responsive Design y Accesibilidad

El sistema implementa diseño responsive completo que garantiza usabilidad en diferentes dispositivos:

- Adaptación automática a pantallas móviles, tablet y desktop
- Menú colapsible en dispositivos móviles
- Tablas scrollables horizontalmente en pantallas pequeñas
- Botones y elementos táctiles optimizados para touch
- Contraste de colores cumpliendo estándares WCAG 2.1 AA
- Navegación por teclado en todos los elementos interactivos

11. Conclusiones del Anexo Visual

Las capturas de pantalla presentadas demuestran la implementación exitosa de un sistema de gestión para H2Vital que digitaliza completamente los procesos administrativos y comerciales de la empresa. La interfaz desarrollada evidencia:

Coherencia Visual: Todas las pantallas mantienen un diseño consistente basado en Material MUI que facilita la navegación intuitiva y reduce la curva de aprendizaje para usuarios no técnicos.

Funcionalidad Completa: Cada módulo implementa operaciones CRUD completas con validaciones en tiempo real, filtros avanzados y herramientas de búsqueda que optimizan la productividad diaria.

Experiencia de Usuario: El sistema proporciona feedback visual inmediato, estados claros con código de colores semántico y navegación contextual que guía al usuario a través de los procesos empresariales.

Escalabilidad: La arquitectura modular evidenciada en las interfaces permite futuras expansiones del sistema manteniendo la coherencia visual y funcional establecida.

La aplicación H2Vital representa una transformación digital exitosa que no solo replica los procesos manuales existentes, sino que los optimiza mediante automatización, validación proactiva y seguimiento integral de toda la información comercial y administrativa de la empresa.

Bibliografía

[1] Cámara de Comercio de España. Informe sobre la digitalización de pymes en españa, 2023. Disponible en: https://www.camara.es. Último acceso: 1 de julio de 2025.

[2] Asociación Española de Empresas de Tratamiento de Aguas. Informe anual del sector de calidad del agua y aire 2024, 2024. Disponible en: https://www.asetub.es. Último acceso: 1 de julio de 2025.

[3] McKinsey & Company. The future of work: Automating administrative tasks, 2022. Disponible en: https://www.mckinsey.com. Último acceso: 1 de julio de 2025.

[4] Greenpeace. Impacto ambiental de la digitalización en pymes, 2023. Disponible en: https://www.greenpeace.org/espana. Último acceso: 1 de julio de 2025.

[5] Copyright © 2014-2025 Evan You. Vue v4 api, 2024. Disponible en: https://vuejs.org/. Último acceso: 1 de julio de 2025.

[6] Contentful. Next.js vs. react: The difference and which framework to choose, 2025. Disponible en: https://www.contentful.com/blog/next-js-vs-react/. Último acceso: 1 de julio de 2025.

[7] SoftWeb Solutions. Next.js vs react: Why to choose next.js in 2024?, 2024. Disponible en: https://www.softwebsolutions.com/resources/nextjs-vs-react.html. Último acceso: 1 de julio de 2025.

[8] Quest Software. Postgresql performance monitoring and optimization guide, 2024. Disponible en: https://www.quest.com/solutions/postgresql/. Último acceso: 1 de julio de 2025.

[9] Docker Inc. Why use compose?, 2024. Disponible en: https://docs.docker.com/compose/intro/features-uses/. Último acceso: 1 de julio de 2025.

[10] NinjaOne. Server backup best practices with examples, 2024. Disponible en: https://www.ninjaone.com/blog/server-backup-5-best-practices/. Último acceso: 1 de julio de 2025.

[11] PostgreSQL.org. Postgresql documentation - official guide, 2024. Disponible en: https://www.postgresql.org/docs/. Último acceso: 1 de julio de 2025.

[12] Material-UI Team. Material-ui: React components for faster development, 2024. Disponible en: https://mui.com/. Último acceso: 1 de julio de 2025.

[13] Prisma. Prisma features, 2025. Disponible en: https://www.prisma.io/. Último acceso: 1 de julio de 2025.

[14] Canonical Ltd. Ubuntu server - for scale out workloads, 2024. Disponible en: https://ubuntu.com/server. Último acceso: 1 de julio de 2025.

[15] freeCodeCamp. How to optimize next.js web apps for better performance, 2025. Disponible en: https://www.freecodecamp.org/news/optimize-nextjs-web-apps-for-better-performance/. Último acceso: 1 de julio de 2025.

[16] PagePro. Next.js performance optimisation (2025) – get started fast, 2025. Disponible en: https://pagepro.co/blog/nextjs-performance-optimization-in-9-steps/. Último acceso: 1 de julio de 2025.

[17] Cybertec. Postgresql performance tuning and optimization, 2024. Disponible en: https://www.cybertec-postgresql.com/en/postgresql-performance-tuning/. Último acceso: 1 de julio de 2025.

[18] Prisma. Prisma with postgresql: Complete integration guide, 2024. Disponible en: https://www.prisma.io/docs/concepts/database-connectors/postgresql. Último acceso: 1 de julio de 2025.

[19] Tessell. Postgresql - concepts, benefits and use cases, 2024. Disponible en: https://www.tessell.com/blogs/postgresql-concepts-benefits-and-use-cases. Último acceso: 1 de julio de 2025.

[20] Percona. What is postgresql database? introduction, benefits, use cases, 2024. Disponible en: https://www.percona.com/blog/what-is-postgresql-used-for/. Último acceso: 1 de julio de 2025.

[21] Prisma. Prisma schema, 2025. Disponible en: https://www.prisma.io/docs/orm/prisma-schema/overview. Último acceso: 1 de julio de 2025.

[22] Prisma. Prisma migrations, 2025. Disponible en: https://www.prisma.io/docs/orm/prisma-migrate. Último acceso: 1 de julio de 2025.

[23] Prisma. Prisma queries, 2025. Disponible en: https://www.prisma.io/docs/orm/prisma-client/queries. Último acceso: 1 de julio de 2025.

[24] Prisma. Prisma tools, 2025. Disponible en: https://www.prisma.io/dataguide/database-tools. Último acceso: 1 de julio de 2025.

[25] GeeksforGeeks. React material ui, 2024. Disponible en: https://www.geeksforgeeks.org/react-material-ui/. Último acceso: 1 de julio de 2025.

[26] MUI Team. Overview - mui system, 2024. Disponible en: https://mui.com/system/getting-started/. Último acceso: 1 de julio de 2025.

[27] Simplilearn. What is docker compose: Example, benefits, 2024. Disponible en: https://www.simplilearn.com/tutorials/docker-tutorial/docker-compose. Último acceso: 1 de julio de 2025.

[28] phoenixNAP. What is docker compose: Definition, usage, benefits, 2024. Disponible en: https://phoenixnap.com/kb/docker-compose. Último acceso: 1 de julio de 2025.

[29] Dev Community. A deep dive into docker compose, 2024. Disponible en: https://dev.to/alexmercedcoder/a-deep-dive-into-docker-compose-27h5. Último acceso: 1 de julio de 2025.

[30] Ken Schwaber and Jeff Sutherland. *The Scrum Guide: The Definitive Guide to Scrum*. Scrum.org, 2020 edition, 2020. Disponible en: https://scrumguides.org/. Último acceso: 1 de julio de 2025.

[31] Atlassian. Trello: Organize anything, together, 2024. Disponible en: https://trello.com/. Último acceso: 1 de julio de 2025.

[32] GanttProject Team. Ganttproject: Free project scheduling and management app, 2024. Disponible en: https://www.ganttproject.biz/. Último acceso: 1 de julio de 2025.

[33] Ian Sommerville. *Software Engineering*. Pearson, 10th edition, 2016.

[34] Andrew Hunt and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Professional, 1999.

[35] Martin Fowler. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley Professional, 2nd edition, 2019.

[36] MUI Team. Material-ui: React components for faster and easier web development, 2024. Disponible en: https://mui.com/. Último acceso: 1 de julio de 2025.

[37] Karl Wiegers and Joy Beatty. *Software Requirements*. Microsoft Press, 3 edition, 2013.

[38] Roger S. Pressman and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9 edition, 2019.

[39] IEEE Computer Society. Ieee recommended practice for software requirements specifications. Standard IEEE Std 830-1998, IEEE, 1998.

[40] Ronald G. Ross. *Principles of the Business Rule Approach*. Addison-Wesley, 2003.

[41] Orlena Gotel and Anthony Finkelstein. An analysis of the requirements traceability problem. *Requirements Engineering*, 1(2):94–101, 1994.

[42] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 2 edition, 2005.

[43] Craig Larman. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*. Prentice Hall, 3 edition, 2004.

[44] Alistair Cockburn. *Writing Effective Use Cases*. Addison-Wesley, 2000.

[45] Ivar Jacobson, Magnus Christerson, Patrik Jonsson, and Gunnar Overgaard. *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992.

[46] Eric Evans. *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley, 2003.

[47] David Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

[48] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, 3 edition, 2003.

[49] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 3 edition, 2012.

[50] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[51] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.

[52] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Pearson, 7 edition, 2015.

[53] Donald A. Norman. *The Design of Everyday Things: Revised and Expanded Edition*. Basic Books, 2013.

[54] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2 edition, 2008.

[55] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.

[56] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems*. Pearson, 7th edition, 2016.

[57] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[58] Unión Europea. Reglamento general de protección de datos (rgpd), 2018. Disponible en: https://eur-lex.europa.eu/eli/reg/2016/679/oj. Último acceso: 1 de julio de 2025.

[59] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 2nd edition, 2012.

Glosario

ACID ACID es un acrónimo que se utiliza en el contexto de las bases de datos relacionales para describir las cuatro propiedades clave que garantizan la integridad y consistencia de las transacciones: Atomicidad, Consistencia, Aislamiento y Durabilidad. 27, 29, 36

bcrypt bcrypt es una función de hash para contraseñas diseñada para ser lenta y resistente a ataques de fuerza bruta, incorporando un salt automático y un factor de coste configurable. 103, 105, 107

BSON BSON (Binary JSON) es un formato de serialización binario que se utiliza para representar datos de forma compacta y eficiente, especialmente en la base de datos MongoDB. Es una alternativa a JSON, pero en formato binario, lo que permite un procesamiento más rápido y una menor sobrecarga de almacenamiento. 27, 29

CDN CDN, por sus siglas en inglés de Content Delivery Network) es una red de servidores distribuidos geográficamente que ayudan a acelerar la entrega de contenido web, acercando los archivos de la web a los usuarios para que se carguen más rápido. 26

CSR CSR (Client-Side Rendering) es una técnica donde las páginas web se renderizan completamente en el navegador del usuario usando JavaScript, en lugar de en el servidor. 23, 35, 160

CSRF La Incursión Por Petición Fraudulenta de Un Sitio Cruce, es un tipo específico de invasión cibernética que se apropia indebidamente de las credenciales de un usuario para ejecutar acciones no permitidas dentro de una plataforma en línea, sin el conocimiento o la aprobación del usuario afectado. 24

CSS CSS Modules es una técnica que convierte CSS en módulos locales, evitando conflictos de nombres de clases. 22

CSS-in-JS CSS-in-JS es una técnica donde los estilos se escriben directamente en JavaScript, permitiendo estilos dinámicos y scoped. 22

DOM DOM (Document Object Model) es una representación estructurada del documento HTML como un árbol de objetos que JavaScript puede manipular dinámicamente. 21

ERP ERP (Enterprise Resource Planning, o Planificación de Recursos Empresariales) es un sistema de software que ayuda a las empresas a gestionar todos los procesos de negocio, desde las finanzas y los recursos humanos hasta la producción y la logística. 25

Ingeniería de Software Disciplina que aplica principios de ingeniería al desarrollo de software para crear sistemas de alta calidad de manera sistemática, disciplinada y cuantificable. 45

ISR ISR (Incremental Static Regeneration) es una técnica de Next.js que combina los beneficios de SSG (Static Site Generation) con la capacidad de actualizar páginas estáticas sin necesidad de reconstruir toda la aplicación. 22, 35

JSON JSON (JavaScript Object Notation) es un formato de intercambio de datos basado en texto, legible para humanos y analizable por máquinas. Se usa para almacenar y transmitir datos estructurados, especialmente en aplicaciones web, donde se envía información entre cliente y servidor. 27, 29

JSX JSX (JavaScript XML) es una extensión de sintaxis para JavaScript que permite escribir marcado similar a HTML dentro de un archivo JavaScript. 22

JWT JWT (JSON Web Token) es un estándar abierto que define una manera compacta y autónoma de transmitir información de forma segura entre partes como un objeto JSON, utilizado comúnmente para autenticación y autorización. 103

Material MUI Biblioteca de componentes de interfaz de usuario para React basada en Material Design de Google, que proporciona componentes consistentes y accesibles. 105

MVC MVC o Modelo-Vista-Controlador es un patrón de arquitectura de software que, utilizando 3 componentes (Vistas, Models y Controladores) separa la lógica de la aplicación de la lógica de la vista en una aplicación. 23

MVCC MVCC significa Multi-Version Concurrency Control (Control de Concurrencia Multi-Versión), permite acceder a la base de datos por multiples usuarios sin bloqueos. 29

MVVM MVVM significa Modelo-Vista-Modelo de Vista (Model-View-ViewModel). Es un patrón de arquitectura de software que ayuda a organizar el código de una aplicación para mejorar la organización, la mantenibilidad y la testabilidad. 23

ORM ORM (Object-Relational Mapping) es una técnica que permite mapear estructuras de una base de datos relacional a objetos en un lenguaje de programación orientado a objetos, simplificando el acceso y manipulación de datos. En esencia, un ORM actúa como una capa de abstracción entre la base de datos y el código de la aplicación, permitiendo que los desarrolladores interactúen con los datos usando objetos en lugar de SQL. 24, 25, 28, 33, 34, 36, 89

PWA PWA (Progressive Web App) es una aplicación web que utiliza tecnologías modernas para ofrecer una experiencia similar a las aplicaciones nativas en dispositivos móviles y escritorio. 21, 22

RBAC RBAC (Role-Based Access Control) es un método de regulación del acceso a recursos basado en los roles asignados a usuarios individuales dentro de una organización, garantizando el principio de menor privilegio. 103, 105, 107

RGPD RGPD (Reglamento General de Protección de Datos) es la normativa europea que regula el tratamiento de datos personales y garantiza la privacidad de los ciudadanos de la Unión Europea, estableciendo principios para el procesamiento legítimo de información personal. 104, 105, 107

Sass (SCSS) Sass es un preprocesador CSS que añade funcionalidades como variables, mixins, anidación y funciones. 22

SEO SEO (Search Engine Optimization) es un conjunto de técnicas y estrategias para mejorar la visibilidad de un sitio web en los resultados orgánicos (no pagados) de los motores de búsqueda, como Google. 23, 35

SSE SSE (Server-Sent Events) es una tecnología web que permite al servidor enviar datos automáticamente al cliente de forma unidireccional en tiempo real, manteniendo una conexión persistente. 88, 98, 99, 104

SSG SSG (Static Site Generation) es una técnica donde las páginas web se generan como archivos HTML estáticos durante el tiempo de construcción (build time), antes de ser desplegadas. 22, 23, 35

SSL SSL (Secure Socket Layer) es un protocolo de seguridad que establece una conexión cifrada entre un servidor web y un navegador, garantizando que todos los datos transmitidos permanezcan privados e íntegros. 89, 96, 97, 112

SSR SSR (Server-Side Rendering) es una técnica de renderizado donde las páginas web se generan en el servidor antes de enviarse al navegador del usuario. 21–23, 32, 35, 160

WebP WebP es un formato de imagen moderno, desarrollado por Google, que ofrece una compresión sin pérdidas y con pérdidas superior a formatos tradicionales como JPEG y PNG. 22

XSS Cross-Site Scripting, o XSS. En su esencia, esta amenaza virtual permite introducir comandos malintencionados en páginas web, causando estragos en el círculo de usuarios de la misma. A grandes rasgos, los infiltrados se benefician de información privada confiscada, como las cookies de identificación almacenadas por el navegador. 24

Índices

Índice de figuras

Índice de tablas