UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

"Desarrollo sistema SaaS para la gestión automática de citas por WhatsApp con inteligencia artificial"

TRABAJO FIN DE GRADO

Junio – 2025

AUTOR: Víctor Valladares Sánchez
DIRECTOR: Desiderio Martí Alcaraz

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

““To succeed you just need to do so many reps that it would be unreasonable for you to fail.”

— Alex Hormozi

RESUMEN

La principal finalidad de este Trabajo de Fin de Grado (TFG) es desarrollar un MVP en formato SaaS (Software as a Service) que aborde uno de los retos más habituales en las empresas: la gestión y reserva de citas con sus clientes. Para ello, se aplican los conocimientos adquiridos a lo largo de la carrera de Ingeniería Informática en Tecnologías de la Información, con el objetivo de diseñar un software escalable que, mediante inteligencia artificial, automatice el agendamiento de citas a través de una interfaz conversacional familiar para la mayoría de los usuarios: **WhatsApp**.

El sistema, impulsado por modelos de lenguaje natural, es capaz de gestionar reservas de forma autónoma, sin intervención humana directa. Además, de ofrecer a cada negocio su propio panel de administración web desde el que puede visualizar, modificar y gestionar tanto citas como pacientes o empleados. La solución busca ser ligera, modular y fácilmente adaptable, sin necesidad de que el cliente final instale ninguna aplicación o acceda a una web: puede reservar su cita directamente desde WhatsApp.

Biblioteca
Universitas Miguel Hernández

Agradecimientos

Quiero agradecer a todas las personas que han hecho posible este proyecto.

En primer lugar, a mi tutor Desiderio Martí Alcaraz, por confiar en mi visión desde el primer momento. Le estaré siempre agradecido por su compromiso incansable, su atención constante y por estar disponible incluso en fines de semana y festivos. Su ayuda ha sido fundamental en cada fase de este trabajo.

A mis padres, Víctor Valladares y Susan Sánchez, por haberme dado todo lo que soy: mi educación, mis valores y un apoyo incondicional que ha sido un motor en todas mis etapas de la vida. Son mi modelo a seguir y un ejemplo diario de amor, esfuerzo y determinación.

A mis abuelos, Fernando Sánchez y Marianita Ruíz de Lope, por enseñarme que el trabajo bien hecho siempre deja huella y que con esfuerzo constante se puede conseguir lo que te propongas.

A mi hermano David, gracias por las charlas que hemos tenido sobre el proyecto, que me han dado ideas y enfoques nuevos en momentos clave.

A Rocío Campanini, por acompañarme en cada paso de este camino. Por tu paciencia, por creer en mí siempre, por animarme en los días largos y celebrar cada pequeño logro como si fuera tuyo. Tu apoyo ha significado más de lo que las palabras pueden expresar.

Y también a su familia, por acogerme y ser una fuente inagotable de cariño, inspiración y apoyo.

En definitiva, gracias a todos por estar siempre ahí

LISTA DE CONTENIDOS

LISTA DE FIGURAS IX

LISTA DE TABLAS XI

1. INTRODUCCIÓN 2
1.1. Contexto del problema 2
1.2. Justificación del Proyecto 3
1.3. Metodología empleada 3

2. ESTADO DEL ARTE 6
2.1. Evolución de la Automatización en la Gestión de Reservas 6
2.1.1. Años 50 y 60: Surgimiento de los Sistemas de Reserva Computarizados (CRS) 6
2.1.2. Años 70 y 80: Expansión y Diversificación de los CRS 7
2.1.3. Años 90: Digitalización y Aparición de Sistemas de Gestión de Propiedades 7
2.1.4. Años 2000: Auge de las .com y la Web 2.0 en la Gestión de Citas 7
2.1.5. Últimos 20 Años: La Revolución Digital y la Era de la Inteligencia Artificial en la Gestión de Citas 8
2.2. Aplicación de la Inteligencia Artificial en la Gestión de Citas 9
2.2.1. Chatbots 9
2.2.1.1. Primera generación: Chatbots basados en reglas (1960 a 1970) 9
2.2.1.2. Segunda generación: Chatbots basados en aprendizaje automático (1990 a 2000) 10
2.2.1.3. Tercera generación: Asistentes virtuales inteligentes (2010 a 2020) 10
2.2.1.4. Cuarta generación: Chatbots basados en LLM's (2020 – Actualidad) 10
2.3. Integración de Chatbots con Plataformas de Mensajería 11
2.3.1. Ventajas de la integración 11
2.3.2. Métodos de integración 12
2.3.2.1. APIs Oficiales 12
2.3.2.2. Servicios de terceros 13
2.4. Comparativa de Soluciones de Automatización en la Gestión de Reservas 13
2.4.1. Software Personalizado: la apuesta por la exclusividad 14
2.4.2. SaaS tradicionales: democratización con limitaciones 14
2.4.3. Sistemas de Gestión con Chatbots en WhatsApp 14
2.5. Desafíos en la Implementación de Sistemas de Gestión con Chatbots en WhatsApp 15

3. MOTIVACIÓN Y OBJETIVOS 17
3.1. Motivación 17
3.2. Objetivos del Proyecto 17
3.2.1. Objetivos generales 17
3.2.2. Objetivos específicos 18
1. Analizar las necesidades y requisitos del sector 18
2. Diseñar una arquitectura modular y escalable 18
3. Evaluar la viabilidad técnica y económica 18
4. Integrar con la API oficial de WhatsApp 18
5. Implementar el chatbot conversacional 18
6. Desarrollar lógica de negocio y conexión a la base de datos 19
7. Desarrollar panel de administración web 19
8. Validar el sistema 19
9. Documentar el proceso y extraer conclusiones 19

4. HIPOTESIS DEL TRABAJO 21
4.1. Visión general 21

4.2. Herramientas Utilizadas 22
4.3. Lenguajes y tecnologías utilizadas 23
4.4. Servicios Externos 24

5. METODOLOGÍA Y RESULTADOS 26
5.1. Fase de Análisis 26
5.1.1. Análisis de Requerimientos 26
5.1.2. Casos de Uso 27
5.1.2.1. Cliente 28
5.1.2.2. Administrador del sistema 29
5.1.3. Requisitos Funcionales 30
5.1.3.1. Subsistema WhatsApp 30
5.1.3.2. Subsistema Panel Web 32
5.2. Planificación 33
5.2.1. Diagrama de Gantt 33
5.2.2. Recursos técnicos 35
5.2.3. Estimación de costes 36
5.2.4. Recursos humanos 39
5.2.5. Plan de negocio 40
5.3. Fase de Diseño 44
5.3.1. Diseño general del sistema 44
5.3.2. Diseño del módulo de chatbot 45
5.3.2.1. Diseño del flujo FAQ 46
5.3.2.2. Diseño del flujo de cita 47
5.3.3. Diseño del módulo integración LLM 48
5.3.4. Diseño de la interfaz web 49
5.3.4.1. Diseño Página de Inicio 50
5.3.4.2. Diseño Calendario de Citas 50
5.3.4.3. Diseño Gestión de Empleados 51
5.3.4.4. Diseño Gestión de Clientes 52
5.3.5. Diseño del backend 52
5.3.5.1. Esquema entidad-relación 53
5.4. Fase de Implementación 54
5.4.1. Implementación General del Sistema 54
5.4.2. Implementación Chatbot 54
5.4.2.1. Integración con Meta 55
5.4.2.2. Flujos del chatbot 56
5.4.2.2.1. Flujo de Registro 58
5.4.2.2.2. Flujo de Cita 59
5.4.2.3. Envío de recordatorios mediante plantillas 62
5.4.2.4. Gestión Manual con Chatwoot 65
5.4.3. Implementación módulo integración LLM 66
5.4.4. Implementación de la interfaz web 67
5.4.4.1. Implementación Página de Inicio 69
5.4.4.2. Implementación Calendario de Citas 71
5.4.4.3. Implementación Gestión de Empleados 73
5.4.4.4. Implementación Gestión de Clientes 74
5.4.5. Implementación del backend 76
5.5. Fase de Pruebas 77
5.5.1. Pruebas Funcionales 77
5.5.2. Evaluación del rendimiento del modelo conversacional 78
5.5.2.1. Elección de la herramienta de evaluación: Ragas 79
5.5.2.2. Preparación del entorno de evaluación 80
5.5.2.3. Iteración y mejora del rendimiento conversacional 81
5.5.2.3.1. Prompt de detección de intención 81
5.5.2.3.2. Prompt de respuesta FAQ 82
5.5.3. Conclusión 83
5.6. Seguridad, escalabilidad y mantenimiento 83

Biblioteca UNIVERSITAS Miguel Hernández

5.6.1. Seguridad 83
5.6.1.1. Autenticación y control de acceso en el panel web 83
5.6.1.2. Seguridad en el chatbot 84
5.6.2. Escalabilidad 85
5.6.3. Mantenimiento y despliegue 86
5.6.3.1. Despliegue del Chatbot 87
5.6.3.2. Despliegue del Panel Web 88
5.7. Usabilidad 88
5.7.1. Interfaz Chatbot 88
5.7.2. Panel de Administración 89

6. CONCLUSIONES Y TRABAJOS FUTUROS 91
6.1. Reflexión personal 91
6.2. Trabajos futuros 92
6.3. Conclusión 93

REFERENCIAS BIBLIOGRÁFICAS 96

ANEXO 1. CONFIGURACIÓN DE CURSOR 102

ANEXO 2. DIAGRAMA DE GANTT 103

ANEXO 3. CASOS DE USO 105

A. CASOS DE USO: ACTOR CLIENTE 105

B. CASOS DE USO: ACTOR ADMINISTRADOR 115

ANEXO 4. PROMPTS 129

A. PROMPT DETECTION INICIAL 129

B. PROMPT DETECTION OPTIMIZADO 129

C. PROMPT FAQ INICIAL 130

D. PROMPT FAQ OPTIMIZADO 131

Biblioteca UNIVERSITAS Miguel Hernández

LISTA DE FIGURAS

Figura 1. Sistemas APIs 13
Figura 2. Visión General Tecnologías 21
Figura 3. Actores del sistema 28
Figura 4. Casos de Uso Cliente 28
Figura 5. Casos de Uso del Admin 29
Figura 6. Ingresos vs Gastos por Escenario 39
Figura 7. Detección de Intención 46
Figura 8. Diseño Flujo FAQ 46
Figura 9. Flujo de Cita 48
Figura 10. Diseño modular proveedor LLM 49
Figura 11. Diseño Página de Inicio 50
Figura 12. Diseño Calendario de Citas 51
Figura 13. Diseño Gestión de Empleados 51
Figura 14. Diseño Gestión de Clientes 52
Figura 15. Diseño BBDD Diagrama Entidad-Relación 53
Figura 16. Proyectos en Github 54
Figura 17. Creación Proyecto Chatbot 54
Figura 18. Petición HTTP Postman 56
Figura 19. Ejemplo Flujo Registro 59
Figura 20. Ejemplo Flujo Cita 61
Figura 21. Ejemplo Recordatorio 62
Figura 22. Plantilla Mensaje Recordatorio 64
Figura 23. Chat de Chatwoot 65
Figura 24. Teléfono Añadido a la Blacklist 66
Figura 25. Configuración Proveedores LLM 67
Figura 26. Creación Proyecto Web Next.js 67
Figura 27. Componentes del Proyecto Web 69
Figura 28. Página Principal 69
Figura 29. Inicio de Sesión 70
Figura 30. Código Temporal de Acceso 70
Figura 31. Código Verificación Email 71
Figura 32. Página Calendario Citas 72
Figura 33. Añadir o Editar Cita 72
Figura 34. Gestión de empleados 73
Figura 35. Añadir o Editar Empleado 74
Figura 36. Gestión de Clientes 75
Figura 37. Añadir o Editar Cliente 75
Figura 38. Tablas Supabase 77
Figura 39. Diagrama explicativo Ragas 80
Figura 40. Resultado prompt detección Ragas 81
Figura 41. Resultado prompt FAQ Ragas 82
Figura 42. Dashboard Supabase 86
Figura 43. Apertura Túnel Ngrok en Puerto 3008 87
Figura 44. Túnel Abierto Ngrok 87
Figura 45. Cursor Rules 102
Figura 46. Ranking LM Arena [40] 102
Figura 47. Diagrama de Gantt Parte 1 103
Figura 48. Diagrama de Gantt Parte 2 104
Figura 49. Chat Bienvenida Usuario Registrado 106
Figura 50. Chat Registro Nuevo Usuario 106
Figura 51. Chat Flujo Registro 107
Figura 52. Chat Pedir Cita 108
Figura 53. Chat Elegir fecha y hora 109
Figura 54. Chat Elegir profesional 110

Figura 55. Chat Confirmar datos cita 111
Figura 56. Calendario Web Cita Reservada sin confirmar 112
Figura 57. Chat Recordatorio Cita 113
Figura 58. Chat confirmación de asistencia 114
Figura 59. Calendario Web Cita confirmada 114
Figura 60. Landing Page 115
Figura 61. Inicio Sesión Doble Factor 116
Figura 62. Calendario Web Administración 117
Figura 63. Web Creación de Nueva Cita 118
Figura 64. Web Editar o Eliminar Cita 119
Figura 65. Web Página de Clientes 120
Figura 66. Web Editar Cliente 121
Figura 67. Web Página de Empleados 122
Figura 68. Web Editar Empleado 123
Figura 69. Chatwoot chat 124
Figura 70. Chatwoot Desactivar Bot 125
Figura 71. Deploy Logs Bot Desactivado 125
Figura 72. Chatwoot Activar Bot 126
Figura 73. Deploy Logs Bot Activado 126
Figura 74. Chatwoot Consultar Estado del Bot 127
Figura 75. Deploy Log Bot Status 127
Figura 76. Chatwoot Envío Mensaje Manual al Cliente 128

Biblioteca
UNIVERSITAS Miguel Hernández

LISTA DE TABLAS

Tabla 1. Herramientas Utilizadas 22

Tabla 2. Lenguajes y tecnologías utilizadas 23

Tabla 3. Servicios Externos 24

Tabla 4. Requisitos Funcionales RF-W1: Registro automático de nuevos clientes 30

Tabla 5. Requisitos Funcionales RF-W2: Detección de intención con IA 30

Tabla 6. Requisitos Funcionales RF-W3: Proceso de reserva de citas por chat 31

Tabla 7. Requisitos Funcionales RF-W4: Envío automático de recordatorios 24h antes 31

Tabla 8. Requisitos Funcionales RF-W5: Transferencia de la conversación a agente humano 31

Tabla 9. Requisitos Funcionales RF-A1: Inicio de sesión seguro con autenticación doble factor 32

Tabla 10. Requisitos Funcionales RF-A2: Gestión completa de citas 32

Tabla 11. Requisitos Funcionales RF-A2: Gestión de clientes y empleados 32

Tabla 12. Requisitos Funcionales RF-A3: Personalización y modificación de prompt 32

Tabla 13. Estimación de Costes 37

Tabla 14. Coste IA por Número de Clínicas 38

Tabla 15. Coste API Meta por Número de Clínicas 38

Tabla 16. Tarifa Personal 40

Tabla 17. Presupuesto Total RRHH 40

Tabla 18. Datos Base para Proyección de Clientes y Costes 42

Tabla 19. Planificación Financiera (2026-2031) 43

Tabla 20. Tablas Base de Datos 76

Tabla 21. Métricas Ragas 79

Tabla 22. Caso de uso CU-C1: Iniciar Chat 105

Tabla 23. Caso de uso CU-C2: Registro 106

Tabla 24. Caso de Uso CU-C3: Preguntar Duda 107

Tabla 25. Caso de Uso CU-C4: Pedir Cita 108

Tabla 26. Caso de Uso CU-C5: Elegir fecha y hora 109

Biblioteca UNIVERSITAS Miguel Hernández

Tabla 27. Caso de Uso CU-C6: Elegir profesional 110
Tabla 28. Caso de Uso CU-C7: Confirmar datos cita 111
Tabla 29. Caso de Uso CU-C8: Recibir recordatorio 24h antes 112
Tabla 30. Caso de Uso CU-C9: Confirmar o cancelar asistencia 113
Tabla 31. Caso de Uso CU-A1: Iniciar Sesión 115
Tabla 32. Caso de uso CU-A2: Consultar citas 116
Tabla 33. Casos de uso CU-A3: Crear cita manual 117
Tabla 34. Caso de uso CU-A4: Editar cita 118
Tabla 35. Caso de uso CU-A5: Eliminar Cita 119
Tabla 36. Caso de Uso CU-A6: Consultar clientes 120
Tabla 37. Caso de Uso CU-A7: Añadir o editar clientes 121
Tabla 38. Caso de Uso CU-A8: Consultar empleados 122
Tabla 39. Caso de Uso CU-A9: Añadir o editar empleados 123
Tabla 40. Caso de Uso CU-A10: Consultar Chatwoot 124
Tabla 41. Caso de Uso: CU-A11: Desactivar Bot 125
Tabla 42. Caso de Uso CU-A12: Activar Bot 126
Tabla 43. Caso de Uso CU-A13: Consultar Estado del Bot 127
Tabla 44. Caso de Uso CU-A14: Enviar Mensaje Manual al Cliente 128

Biblioteca
UNIVERSITAS Miguel Hernández

CAPÍTULO 1

INTRODUCCIÓN

Biblioteca
UNIVERSITAS Miguel Hernández

1. INTRODUCCIÓN

En los últimos años, la expansión de los dispositivos móviles ha influido notablemente en los hábitos de consumo, modificando la forma en que las personas descubren, valoran y contratan productos o servicios. En este nuevo panorama digital, las redes sociales se han consolidado como una de las principales vías de captación de clientes, mientras que aplicaciones de mensajería como WhatsApp se han convertido en herramientas habituales de comunicación para muchos negocios locales.

Partiendo de esta realidad, el presente Trabajo de Fin de Grado se centra en el diseño y desarrollo de una solución software orientada a la automatización de la gestión de citas, apoyándose en el uso de inteligencia artificial e integrando WhatsApp como canal principal de interacción. El objetivo es ofrecer una alternativa accesible, eficaz y cercana para pequeñas empresas y profesionales, aprovechando herramientas digitales que ya forman parte del día a día tanto de usuarios como de empresas.

1.1. Contexto del problema

Analizando mi entorno, he descubierto el problema que supone la gestión de citas, sobre todo en negocios del día a día como peluquerías, dentistas, clínicas de fisioterapia o centros de entrenamiento. A pesar de la gran cantidad de recursos que consume todo el proceso, desde que el cliente desea reservar hasta que llega al establecimiento, muchos de estos negocios continúan utilizando métodos manuales que requieren la participación del personal, en muchos casos cualificado, el cual pierde el foco en el cliente presente para atender al cliente futuro.

Este proceso no solo consume tiempo, sino que además es propenso a errores y puede derivar en ineficiencias como citas duplicadas, olvidos o pérdidas de oportunidad. Algunos negocios intentan esquivar este problema desarrollando una aplicación propia; sin embargo, esto no se ajusta a la realidad de muchos usuarios, que prefieren canales de comunicación más inmediatos y familiares

como WhatsApp, dejando fuera de estos sistemas a aquellos que no se adaptan a esta forma de interacción.

Por tanto, se detecta una oportunidad clara: automatizar la gestión de citas a través de una interfaz conversacional, manteniendo al mismo tiempo la cercanía que aporta una herramienta tan extendida como WhatsApp. La incorporación de inteligencia artificial permite, además, ofrecer una experiencia personalizada, escalable y capaz de operar 24/7 sin intervención humana.

1.2. Justificación del Proyecto

El desarrollo de este sistema responde a una necesidad real observada en el entorno de las PYMES españolas: disponer de una herramienta de gestión de citas sin barreras tecnológicas ni costes elevados, y que se adapte a la forma en que los clientes ya se comunican habitualmente.

Además de resolver un problema real, el proyecto plantea un reto que permite explorar la integración de diferentes tecnologías emergentes: procesamiento de lenguaje natural mediante modelos de lenguaje, automatización de flujos conversacionales, bases de datos en la nube, APIs de mensajería y frameworks modernos de desarrollo web. entre otras tecnologías.

Esta propuesta busca implementar un sistema completo, eficiente y funcional combinando herramientas modernas para ofrecer un servicio de calidad tanto para el negocio como para su cliente final. Desde su concepción, se ha diseñado de forma modular y escalable, pensado en su adaptación a otros sectores más allá de su caso de uso inicial.

1.3. Metodología empleada

Para el desarrollo de este proyecto se ha seguido un enfoque ágil e iterativo, adaptado a mi situación de desarrollador único. Se ha estructurado el proyecto en fases de desarrollo incremental, similares a pequeños sprints, que han permitido construir, validar y mejorar el sistema de manera progresiva.

En cada iteración se abordaron funcionalidades específicas, la primera iteración, por ejemplo, consistió en conectar la API de Whatsapp, integrar la inteligencia artificial y utilizar una hoja de cálculo online como base de datos. Cada iteración

permitía testear lo más rápido posible cada hipótesis, ajustando decisiones técnicas sobre la marcha en función de los resultados obtenidos y la evolución natural del proyecto.

La selección de herramientas se realizó de forma estratégica para optimizar tiempos de desarrollo y garantizar la escalabilidad del sistema: Supabase se utilizó como servicio de backend y base de datos en la nube, Railway para el despliegue de la lógica del chatbot, Netlify para el hosting web, y la API de WhatsApp a través de Meta para las comunicaciones con los usuarios. Para el procesamiento del lenguaje natural se empleó la API de OpenAI, trabajando con modelos económicos como GPT-4o mini.

Esta metodología permitió validar funcionalidades de forma temprana, detectar posibles problemas técnicos en etapas iniciales y garantizar que el producto final fuera estable, funcional y fácilmente ampliable en el futuro.

Biblioteca
UNIVERSITAS Miguel Hernández

CAPÍTULO 2

ESTADO DEL ARTE

2. ESTADO DEL ARTE

En este capítulo, se analizará la evolución de la automatización en la gestión de reservas, destacando cómo la tecnología ha transformado este proceso a lo largo del tiempo. Se explorarán los sistemas iniciales, su desarrollo y las tendencias actuales que integran inteligencia artificial y plataformas de mensajería para optimizar la experiencia del usuario.

2.1. Evolución de la Automatización en la Gestión de Reservas

La gestión de reservas ha experimentado una transformación significativa desde mediados del siglo XX hasta la actualidad, impulsada por avances tecnológicos que han mejorado la eficiencia y la experiencia de usuario.

2.1.1. Años 50 y 60: Surgimiento de los Sistemas de Reserva Computarizados (CRS)

Durante las décadas de 1950 y 1960, la industria de la aviación se enfrentó a un crecimiento exponencial en la demanda de vuelos, lo que evidenció las limitaciones de los sistemas manuales de gestión de reservas basadas en registros físicos, susceptibles a errores y falta de eficiencia. En respuesta a estas necesidades, se inició una revolución tecnológica en el sector.

La colaboración entre American Airlines e IBM dio lugar al desarrollo del sistema SABRE (Semi-Automated Business Research Enviroment) **[1]**, como se referencia en el paper de Karsten, el primer CRS, que se implantó a finales de la década de 1950 y se consolidó en 1960. SABRE permitió automatizar la gestión de reservas mediante el uso de ordenadores IBM 7090, facilitando el procesamiento de miles de llamadas diarias y reduciendo significativamente los errores asociados a la intervención humana.

Este sistema supuso un hito en la forma de gestionar las reservas en la aviación y estableció las bases del desarrollo de sistemas más avanzados.

2.1.2. Años 70 y 80: Expansión y Diversificación de los CRS

Los sistemas de reserva centralizada (CRS) evolucionaron hacia plataformas multiservicio, sentando las bases para la integración de reservas en sectores como salud o servicios profesionales. En su trabajo, Toledo Valera **[2]** analiza cómo esta evolución tecnológica, junto con el modelo SaaS, ha permitido que pequeñas empresas accedan a soluciones de gestión antes reservadas a grandes corporaciones. Su propuesta se centra en el desarrollo de un sistema ERP como servicio, demostrando que la accesibilidad, la escalabilidad y la reducción de costes de despliegue hacen que el software bajo demanda sea especialmente atractivo para PYMES. Este enfoque anticipó el modelo de distribución actual, donde sistemas SaaS permiten automatizar tareas clave con barreras de entrada mínimas.

2.1.3. Años 90: Digitalización y Aparición de Sistemas de Gestión de Propiedades

Durante los años 90, la digitalización transformó la gestión de reservas con la llegada de los Sistemas de Gestión de Propiedades (PMS), especialmente en el sector hotelero. Estos sistemas centralizaban datos en tiempo real, integraban múltiples canales (teléfono, fax e internet) y personalizaban la experiencia del cliente mediante perfiles históricos. La conexión con plataformas como los Sistemas de Distribución Global (GDS) permitió a los negocios gestionar reservas de forma más eficiente y ofrecer servicios personalizados.

Aunque inicialmente diseñados para hoteles, los principios de los PMS comenzaron a aplicarse en otros sectores. Por ejemplo, en la salud surgieron sistemas como, que gestionaban citas médicas sincronizadas con historiales clínicos digitales. Estos avances sentaron las bases para las plataformas SaaS modernas al introducir arquitecturas centralizadas y escalables.

2.1.4. Años 2000: Auge de las .com y la Web 2.0 en la Gestión de Citas

La década de los 2000 estuvo marcada por la consolidación de internet y la llegada de la Web 2.0, que transformó la gestión de reservas al permitir una

interacción más directa y autónoma entre usuarios y empresas. Plataformas digitales comenzaron a ofrecer sistemas accesibles para pequeñas y medianas empresas, integrando calendarios interactivos y disponibilidad en tiempo real. Herramientas como OpenTable (1998) en el sector restaurantero y Zocdoc (2007) en el ámbito médico permitieron a los usuarios gestionar sus citas directamente desde páginas web, eliminando la necesidad de llamadas telefónicas. Esta evolución tecnológica no solo mejoró la experiencia de usuario, sino que también aumentó la seguridad y eficiencia de los sistemas, como señala Bates **[3]**, quien destaca el papel clave de las tecnologías de la información en la reducción de errores operativos y la mejora de la personalización en entornos complejos como el sanitario.

El enfoque centrado en el cliente, característico de la Web 2.0, impulsó funcionalidades como recordatorios automáticos por correo electrónico o SMS, mejorando la experiencia del usuario y reduciendo ausencias. Además, estas plataformas comenzaron a integrar APIs, facilitando la sincronización con herramientas externas como CRMs o sistemas internos de gestión.

Este período democratizó el acceso a tecnologías avanzadas, sentando las bases para los sistemas actuales basados en inteligencia artificial y mensajería instantánea.

2.1.5. Últimos 20 Años: La Revolución Digital y la Era de la Inteligencia Artificial en la Gestión de Citas

En las últimas dos décadas, la consolidación de tecnologías basadas en la nube, junto con la explosión de dispositivos móviles y aplicaciones, ha revolucionado la gestión de citas. Herramientas como Google Calendar (2006) y Calendly (2013) han redefinido la forma de organizar y automatizar el proceso de reserva, ofreciendo calendarios digitales, recordatorios automáticos y la sincronización con sistemas CRM a través de APIs.

Además, la integración de la inteligencia artificial ha transformado estos sistemas, permitiendo que el proceso completo —desde la interacción inicial con el cliente hasta la confirmación de la cita— se gestione de manera integral y eficiente. Este enfoque no solo optimiza la operatividad, sino que también mejora

significativamente la experiencia del usuario, adaptándose en tiempo real a sus necesidades.

2.2. Aplicación de la Inteligencia Artificial en la Gestión de Citas

La IA está siendo una tecnología disruptiva estos últimos años y ha cambiado por completo muchos sectores, en el mundo de las citas ha transformado como las empresas organizan y optimizan sus profesionales y recursos. En este apartado se explora como se está utilizando en 2025 la IA para la gestión de las citas, resaltando tanto los beneficios como los desafíos que conlleva su implementación.

2.2.1. Chatbots

Según la Real Academia Española (RAE), "un chatbot es un programa informático que puede mantener una conversación con los usuarios a través de mensajes de texto o por voz, utilizando inteligencia artificial para interpretar y responder a las consultas de manera automática". Los chatbots no son algo nuevo, en 1950 Alan Turing imaginó un programa informático con la capacidad de mantener una conversación con un grupo de gente y de hacer creer a estos que quien se hallaba al otro lado de la pantalla no era un humano. Esto se llama el Test de Turing, considerado por muchos el inicio de los chatbots que posteriormente surgieron en los años 60.

2.2.1.1. Primera generación: Chatbots basados en reglas (1960 a 1970)

Estos primeros chatbots como ELIZA (1960) un asistente que simulaba un psicoterapeuta, haciendo preguntas al usuario y PARRY (1972), el cuál simulaba una persona con esquizofrenia paranoide considerado más avanzado por tener 'personalidad' ambos funcionaban mediante reglas predefinidas y coincidencia de patrones para simular conversaciones humanas. Pese a ser innovadores en su época, tenían capacidades limitadas: Una baja velocidad de respuesta, no era capaz de aprender de la conversación y no comprendían el contexto cuando la conversación salía de las palabras clave. **[4]**

2.2.1.2. Segunda generación: Chatbots basados en aprendizaje automático (1990 a 2000)

La segunda generación de chatbots destaca por su capacidad de aprender de interacciones previas, con el aprendizaje automático nació A.L.I.C.E (Artificial Linguistic Internet Computer Entity) en 1995 desarrollado con un lenguaje creado a propósito llamado AIML (Artificial Inteligence Markup Language) con el cual ALICE contaba con 41.000 plantillas y patrones relacionales un gran avance comparado con las 200 palabras clave y reglas de ELIZA. Como se recoge en el trabajo de Adamopoulou y Moussiades (2020) **[4]**, estos sistemas eran limitados: no podían aprender, responder con fluidez ni entender el contexto más allá de palabras clave, pero sentaron las bases de la interacción humano-máquina moderna.

2.2.1.3. Tercera generación: Asistentes virtuales inteligentes (2010 a 2020)

Apple lideró el sector integrando la inteligencia artificial y el procesamiento del lenguaje natural con Siri (2010) que si bien no es un chatbot fue un hito en la industria y sorprendía la capacidad de comprender y procesar el lenguaje humano en múltiples idiomas, permitiendo interacciones fluidas y la realización de tareas complejas. Años después tuvimos a Google Now (2012) y Alexa (2014) siendo embebidos en dispositivos hardware principalmente para realizar tareas vinculadas con la domótica.

2.2.1.4. Cuarta generación: Chatbots basados en LLM's (2020 – Actualidad)

Los LLM (Large Lenguage Models), como GPT-3, GPT-4, Gemini, Claude… Han llevado la IA conversacional a un nuevo nivel. Estos modelos utilizan redes neuronales profundas y han sido entrenados con volúmenes de datos, lo que les permite generar respuestas coherentes, contextuales y naturales en múltiples idiomas y formatos.

A diferencia de las anteriores generaciones, las cuales dependían de reglas fijas o aprendizaje limitado, los LLM gracias al procesamiento del lenguaje natural

(NLP) y a técnicas como el aprendizaje por refuerzo con retroalimentación humana (RLHF), estos chatbots mejoran continuamente su capacidad de generar respuestas útiles y adaptadas al usuario.

2.3. Integración de Chatbots con Plataformas de Mensajería

La proliferación de plataformas de mensajería instantánea como WhatsApp, Telegram, Facebook o Instagram ha creado la necesidad de integrar estos chatbots para suplir las necesidades de ciertas empresas que utilizan estos medios como forma de adquisición de clientes, cierre de ventas, concreción de citas o atención postventa. Ante esta necesidad Meta y Telegram ofrecen sus servicios de API para facilitar esta implementación permitiendo automatizar y personalizar la comunicación entre empresas y usuarios.

2.3.1. Ventajas de la integración

La integración de chatbots en WhatsApp ha demostrado ser una solución altamente efectiva para mejorar la atención al cliente, especialmente en pequeñas empresas con recursos limitados. Según un estudio reciente **[5]**, un chatbot bien implementado **puede reducir el tiempo de respuesta de 18.55 minutos a solo 0.8 segundos**, permitiendo una interacción casi instantánea con los clientes.

Además, el sistema alcanzó una precisión del 87.75% en la clasificación de texto, lo que mejora significativamente la claridad de las respuestas. Esto contrasta con los problemas que enfrentan las pequeñas empresas al atender manualmente a los clientes, donde más del 80% reporta respuestas tardías, el 50% es ignorado y el 45.24% recibe información poco clara o irrelevante.

El impacto en la eficiencia es aún más evidente cuando se analiza la escalabilidad de estos chatbots. En pruebas comparativas realizadas en el estudio anteriormente mencionado **[5]**, un chatbot de WhatsApp demostró ser **1391 veces más rápido** que la respuesta manual de un propietario de negocio, permitiendo gestionar simultáneamente múltiples consultas sin esfuerzo.

2.3.2. Métodos de integración

La implementación de chatbots en plataformas de mensajería, se puede hacer mediante herramientas de terceros o las APIs oficiales que permiten la interacción entre usuarios y la inteligencia artificial. Cada plataforma o servicio ofrece sus soluciones específicas para integrar chatbots, permitiendo automatizar la comunicación y mejorar la experiencia de cliente.

2.3.2.1. APIs Oficiales

Las APIs oficiales como WhatsApp Business API, Telegram Bot API y Facebook Messenger API, son la opción más segura y estable a largo plazo ya que están desarrolladas por las propias plataformas, sin embargo, pueden requerir de procesos de aprobación y configuraciones complejas.

La **WhatsApp Business API [6]** permite automatizar respuestas, enviar notificaciones y procesar solicitudes de clientes de forma eficiente.

En el caso de **Telegram [7]**, permite crear un bot a cualquier usuario sin necesidad de autorización previa. Este sistema es altamente flexible y admite múltiples funciones, como la automatización de respuestas, el envío de contenido multimedia y la ejecución de comandos personalizados. Gracias a esta accesibilidad, Telegram se ha convertido en una de las plataformas más populares para el desarrollo de chatbots sin restricciones comerciales.

Por último, **Facebook Messenger [8]** cuenta con todas las funcionalidades de WhatsApp al ser ambos de la empresa Meta, pero cuenta con la integración de pagos que permite gestionar transacciones sin salir de Messenger.

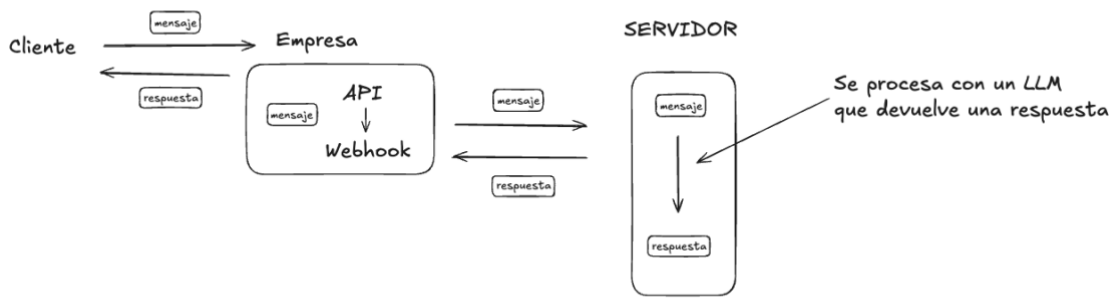

Figura 1. Sistemas APIs

2.3.2.2. Servicios de terceros

Si bien las APIs oficiales son la opción más robusta, su configuración y aprobación pueden ser un obstáculo para algunas empresas. Para facilitar el proceso, existen servicios de terceros como Twilio **[9]** que actúan como intermediarios para acceder a WhatsApp Business API sin necesidad de gestionar directamente la infraestructura. Estos servicios permiten a los desarrolladores integrar chatbots en WhatsApp de manera sencilla, aunque con un modelo de pago basado en el volumen de mensajes enviados.

También existen alternativas no oficiales como Baileys **[10]**, una biblioteca de código abierto que permite interactuar con WhatsApp Web mediante ingeniería inversa. Su principal ventaja es que no requiere aprobación de Meta, permitiendo a cualquier usuario crear un chatbot sin restricciones. Sin embargo, al no ser una solución oficial, existe el riesgo de que WhatsApp bloquee las cuentas que utilicen este método, además de que cualquier actualización en WhatsApp Web podría hacer que Baileys deje de funcionar.

2.4. Comparativa de Soluciones de Automatización en la Gestión de Reservas

Como ya hemos visto en anteriores secciones, la digitalización ha cambiado la forma en la que los negocios gestionan sus citas y reservas. Desde peluquerías y fisioterapeutas hasta clínicas dentales o estudios de tatuajes buscan alternativas al clásico libro de citas en formato físico para mejorar la experiencia de cliente y optimizar su tiempo.

Hoy en día, existen tres diferenciados enfoques para gestionar las reservas en estos sectores:

- **Software personalizado:** Negocios que delegan en desarrolladores independientes o consultoras la creación de una aplicación personalizada para la gestión de las reservas.
- **SaaS tradicionales:** Plataformas estándar en la nube con apps o sitios web donde los clientes pueden reservar.

- **Soluciones innovadoras que implementan WhatsApp:** Sistemas que automatizan respuestas y permiten reservas mediante conversaciones en WhatsApp además de estar integradas con herramientas de gestión como los SaaS clásicos.

A continuación, analizamos las ventajas, desventajas y ejemplos de cada solución.

2.4.1. Software Personalizado: la apuesta por la exclusividad

Los negocios que priorizan la identidad de marca suelen preferir esta opción. La principal ventaja radica en el control total: diseño personalizado, colores corporativos, sistema propio de fidelización y funciones concretas que puedan necesitar para su caso de uso. Sin embargo, este modelo exige una inversión inicial alta (entre 10.000 y 100.000 € según funcionalidades) y dependencia continua de soporte técnico. Un estudio realizado por McKinsey en colaboración con la Universidad de Oxford **[11]** en el 2012 analizó más de 5.400 proyectos de IT y reveló que **el 66 % de estos proyectos software tienen sobrecostes**, y un tercio exceden el tiempo estimado.

2.4.2. SaaS tradicionales: democratización con limitaciones

Plataformas como Booksy **[12]** en el sector de la estética/belleza o Timp **[13]** en el del fitness/salud han dado acceso a las pymes a tecnología avanzada para mejorar sus servicios. No obstante, la estandarización limita la diferenciación dando herramientas funcionales pero genéricas. Además del acumulado coste recurrente. A esto se suma el coste recurrente asociado: con planes que oscilan entre 20 y 100 € al mes según el proveedor y el tamaño del negocio, en un periodo de 5 años el cliente puede llegar a invertir entre 1.200 y 6.000 € en una herramienta que no controla ni posee, simplemente por el derecho a usarla.

2.4.3. Sistemas de Gestión con Chatbots en WhatsApp

Este modelo está basado en los SaaS tradicionales mencionados anteriormente, pero evita a los clientes tener que instalar una app o acceder a una web para solicitar una reserva. Mediante la interfaz conversacional de WhatsApp permite agendar citas en lenguaje natural, aprovechando la penetración de WhatsApp en

el mercado hispano para facilitar la usabilidad y mejorar la experiencia de usuario además de automatizar la atención al cliente.

Este nicho es demasiado reciente y actualmente existen pocos productos, pero podríamos destacar a Yalo **[14]** y HubSpot **[15]**, aunque están más orientados a ventas, no al modelo de reserva de citas y myPhysia una startup del parque científico de la Universidad Miguel Hernández que aplica esta solución en el sector de las clínicas de fisioterapia **[16]**.

2.5. Desafíos en la Implementación de Sistemas de Gestión con Chatbots en WhatsApp

La innovación siempre conlleva unos ciertos desafíos a completar antes de estar completamente implementado y asumido como propio en el caso de estos sistemas puede existir una resistencia al cambio de algunos clientes que no acepten ser atendidos por máquinas en vez de personas, también es importante garantizar la protección de los datos personales para evitar posibles ataques que puedan manipular la IA para filtrar datos de la base de datos.

Otro reto es la integración de estas IA's con los sistemas ya existentes y la sincronización para garantizar el correcto funcionamiento.

Por último, en línea con la propuesta de Ley de Inteligencia Artificial de la Unión Europea **[17]**, los proveedores de sistemas de IA que interactúen con personas físicas deben garantizar la transparencia, informando claramente a los usuarios de que están comunicándose con una inteligencia artificial. Este requisito, aún en desarrollo legislativo, busca fomentar la confianza del usuario y evitar decisiones basadas en percepciones erróneas. Las empresas que utilicen estas tecnologías deben prepararse para implementar mecanismos de notificación, ya que el incumplimiento podría derivar en sanciones económicas y responsabilidades legales.

CAPÍTULO 3

MOTIVACIÓN Y OBJETIVOS

Biblioteca
UNIVERSITAS Miguel Hernández

3. MOTIVACIÓN Y OBJETIVOS

3.1. Motivación

La idea de este proyecto nace de una observación directa de las necesidades que afrontan muchos negocios locales y pequeñas empresas en mi entorno de Elche. En mi propia experiencia como cliente y conversaciones con varios propietarios de este tipo de negocios me di cuenta de que, con demasiada frecuencia, se deja de atender al cliente que tienen en frente para anotar, avisar, confirmar y cancelar citas. Se trata de una tarea rutinaria que, gracias a la inteligencia artificial, puede automatizarse y ofrecer incluso mejores resultados.

En un contexto donde aplicaciones como WhatsApp forman parte esencial de la realidad de la mayoría de los españoles en su vida cotidiana. Apoyarse en el canal de mensajería más extendido tiene todo el sentido cuando entran en juego las economías de red: una plataforma gana más valor cuantos más usuarios la utilizan. Según estimaciones de eMarketer, hechas en julio de 2022, más de 32 millones de personas, aproximadamente un 66% de la población, emplean WhatsApp de forma activa en España **[18]**, una cifra que habla por sí sola de su implementación y de su facilidad de uso.

Frente a esta situación, surge la motivación de crear una solución accesible, funcional y adaptada a la realidad actual de las empresas, que permita automatizar la gestión de citas. Además de cubrir una necesidad real, el proyecto representa un reto personal y académico: integrar distintas tecnologías en un sistema práctico, modular y escalable.

3.2. Objetivos del Proyecto

3.2.1. Objetivos generales

Desarrollar una solución software que permita a las pequeñas empresas automatizar por completo la gestión de citas a través de un chatbot integrado en WhatsApp, ofreciendo una experiencia fluida, fiable y fácilmente escalable tanto para el negocio como para sus clientes.

3.2.2. Objetivos específicos

1. **Analizar las necesidades y requisitos del sector**

Realizar entrevistas y tener conversaciones relevantes con propietarios, principalmente clínicas de fisioterapia, peluquerías y centros de entrenamiento, para identificar sus procesos actuales de reserva, los puntos de dolor más frecuentes y las expectativas del cliente final. A partir de esa información descartar o añadir requisitos funcionales al producto.

2. **Diseñar una arquitectura modular y escalable**

Definir los componentes que conformarán el sistema (LLM, plataforma, base de datos, tecnologías web y servicios externos) y las relaciones entre ellos. El diseño debe permitir añadir nuevas funciones o integrarse con otros canales de mensajería en el futuro, sin reescribir la lógica principal.

3. **Evaluar la viabilidad técnica y económica**

Estimar los recursos necesarios, computacionales y humanos, y calcular el coste mensual en distintos escenarios de uso (número de usuarios, cantidad de mensajes, almacenamiento, llamadas a los LLM…). Con estos datos podremos determinar si la solución puede ser sostenible para la realidad actual de una PYME y que la tecnología es adecuada en términos de rendimiento y mantenimiento.

4. **Integrar con la API oficial de WhatsApp**

Registrar un número de empresa en Meta for Developers, completar la configuración de la WhatsApp Business Cloud API y establecer un canal de comunicación bidireccional que garantice el envío y la recepción fiables de mensajes entre los usuarios y el servidor.

5. **Implementar el chatbot conversacional**

Desarrollar los flujos de diálogo necesarios para gestionar reservas, modificar citas, enviar recordatorios automáticos y responder preguntas frecuentes. Utilizar modelos de lenguaje para dotar al bot de comprensión del contexto y capacidad de respuesta natural, manteniendo siempre un tono profesional y cercano.

6. **Desarrollar lógica de negocio y conexión a la base de datos**

Crear los modelos de datos y las consultas necesarias para almacenar clientes, citas, empleados y configuraciones de empresas. Garantizar la consistencia, integridad y escalabilidad del sistema.

7. **Desarrollar panel de administración web**

Diseñar e implementar una interfaz accesible desde cualquier navegador que permita al negocio, revisar añadir, editar o cancelar citas. Además de gestionar clientes y empleados.

8. **Validar el sistema**

Ejecutar pruebas unitarias e integradas para cada módulo: comprobar que los casos de uso definidos se cumplen, analizar la eficacia de los prompts y el LLM seleccionado para distintos escenarios y la respuesta ante errores.

9. **Documentar el proceso y extraer conclusiones**

Registrar las decisiones de diseño, los problemas encontrados y las soluciones aplicadas, elaborar un análisis crítico de los resultados, identificando las limitaciones del prototipo y proponiendo posibles mejoras o líneas de trabajo futuro.

CAPÍTULO 4

HIPOTESIS DEL TRABAJO

Biblioteca
UNIVERSITAS Miguel Hernández

4. HIPOTESIS DEL TRABAJO

La finalidad de este capítulo es describir el "esqueleto" técnico sobre el que se sostiene la solución. Se muestra las distintas herramientas, lenguajes y tecnologías utilizadas en el desarrollo del sistema. El objetivo es contextualizar el entorno técnico desde el cual se ha construido la solución, explicando por qué se han elegido ciertas herramientas frente a otras y qué papel cumple cada una en el sistema.

4.1. Visión general

El sistema se concibe como una plataforma SaaS ligera y modular, pensado para facilitar la gestión de citas automatizándolas sin asumir grandes costes ni mantener servidores propios.

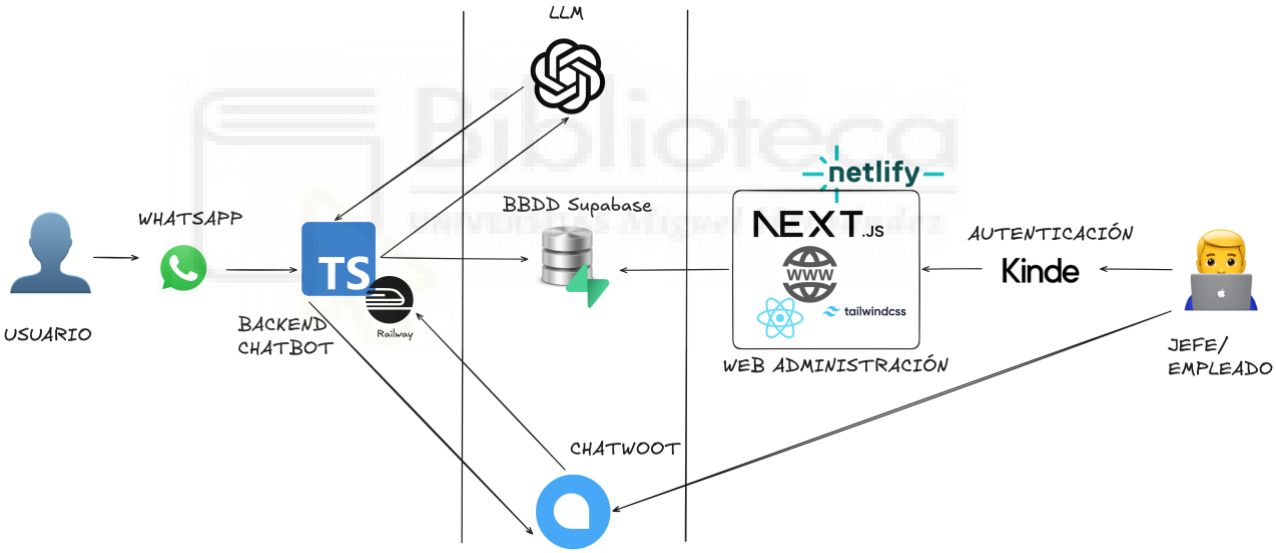

Figura 2. Visión General Tecnologías

En esta imagen se muestra una representación global del ecosistema tecnológico del sistema. A la izquierda, el cliente inicia la conversación a través de WhatsApp, que se conecta con el backend del chatbot desplegado en **Railway [19]**. Este backend se comunica con la base de datos gestionada en **Supabase [20]**, consulta la lógica del sistema escrita en **TypeScript [21]**, y utiliza los modelos de lenguaje (LLM) como **OpenAI [22]** o **DeepSeek [23]** para interpretar mensajes y generar respuestas si es necesario.

Cuando la IA no logra entender al usuario o se requiere intervención manual, el mensaje se transfiere a **Chatwoot [24]**, una plataforma de atención al cliente. Paralelamente, el negocio accede a una web de administración desarrollada con **Next.js [25]** y desplegada en **Netlify [26]**. Para garantizar el acceso seguro, se ha integrado un sistema de autenticación mediante **Kinde Auth [27]**.

Esta visión de conjunto permite entender de un solo vistazo cómo se conectan todos los componentes y qué tecnologías intervienen en cada parte del sistema.

4.2. Herramientas Utilizadas

Durante el desarrollo del sistema se han empleado diversas herramientas de apoyo que facilitaron la programación, depuración y pruebas del sistema. La siguiente tabla resume las más relevantes, indicando su propósito y el papel que han desempeñado en el proyecto.

Nombre	Uso	Logo
Cursor [28]	Editor con IA usado para el desarrollo, especialmente útil para acelerar tareas repetitivas, refactorizar funciones y validar estructuras de código en tiempo real.	
Ngrok [29]	Herramienta utilizada para exponer servidor local del chatbot a través de una URL pública temporal. Clave durante las pruebas con WhatsApp Cloud API antes de los despliegues en producción.	ngrok
Postman [30]	Utilizada para validar el funcionamiento de los endpoints de la API de Meta. Gracias a esta herramienta, pude depurar errores y simular peticiones antes de conectar con el backend real.	POSTMAN

Tabla 1. Herramientas Utilizadas

4.3.Lenguajes y tecnologías utilizadas

El sistema desarrollado combina una serie de tecnologías modernas que permiten ofrecer una solución SaaS ligera, modular y escalable. Las decisiones técnicas se han tomado priorizando criterios de rapidez de desarrollo, compatibilidad entre componentes y facilidad de mantenimiento. A continuación, se detallan los lenguajes y frameworks más relevantes.

Nombre	Uso	Logo
TypeScript	Lenguaje utilizado en todo el proyecto por su tipado estático, que mejora la fiabilidad del código. Facilita detectar errores en desarrollo y mantener código robusto.	
Next.js	Framework empleado para crear el panel web, optimizado para rendimiento y escalabilidad. Su sistema de rutas y compatibilidad con TypeScript agilizan el desarrollo.	
Tailwind CSS [31]	Librería utility-first para aplicar estilos de forma rápida y coherente. Evita archivos CSS complejos y acelera el diseño visual.	
Shadcn/ui [32]	Colección de componentes accesibles y personalizables basada en Tailwind y Radix UI. Aporta consistencia visual sin sacrificar velocidad de desarrollo.	
PostgreSQL [33]	Motor de base de datos relacional utilizado mediante Supabase. Robusto, escalable y con soporte completo para relaciones entre tablas.	
tRPC [34]	Librería que conecta frontend y backend con llamadas tipadas sin definir rutas REST. Asegura integridad entre datos enviados y recibidos.	

Tabla 2. Lenguajes y tecnologías utilizadas

4.4. Servicios Externos

En esta sección se recogen los servicios externos clave que han sido utilizados para complementar el desarrollo de la aplicación. Cada uno de ellos cumple una función específica dentro del sistema, desde el despliegue hasta la autenticación o el soporte conversacional. La tabla resume para qué se ha empleado cada uno, facilitando así la comprensión del ecosistema técnico en el que se apoya la solución.

Nombre	Uso	Logo
Supabase	Se ha utilizado como base de datos central para almacenar información de citas, clientes y empleados. También gestiona el control de acceso multitenant mediante sus políticas de RLS.	supabase
Kinde Auth	Usado para implementar el sistema de login seguro del panel web, permitiendo que solo usuarios autorizados puedan acceder a la interfaz de administración.	Kinde
Railway	Plataforma donde se ha desplegado el backend del chatbot. Ha facilitado la exposición del webhook y la gestión de variables de entorno sin necesidad de mantener infraestructura propia.	
Netlify	Sirve como entorno de despliegue para el panel web. Cada vez que se hace un cambio en el código, este se compila automáticamente y se publica en producción.	netlify
OpenAI / DeepSeek	Proveedores de IA utilizados para interpretar los mensajes de los usuarios y generar respuestas en lenguaje natural. El sistema permite alternar entre ellos según disponibilidad o coste.	deepseek
Chatwoot	Sistema integrado como respaldo para el chatbot, que permite a un humano tomar el control de la conversación en caso de que el bot no sepa responder o haya un error.	chatwoot

Tabla 3. Servicios Externos

CAPÍTULO 5

METODOLOGÍA Y RESULTADOS

Biblioteca
UNIVERSITAS Miguel Hernández

5. METODOLOGÍA Y RESULTADOS

Tras definir las tecnologías utilizadas, este capítulo detalla cómo se ha organizado el proyecto en cuanto a tiempo y recursos (humanos, técnicos y económicos) necesarios para ser completado. La planificación se ha basado en una combinación realista de **hitos académicos** (como entregas del TFG al tutor) e **hitos ingenieriles** (como el diseño de la base de datos o la integración de APIs).

El trabajo se ha abordado en iteraciones cortas inspiradas en la filosofía ágil, pero adaptadas a un único desarrollador. Cada fase concluye con un hito claro que permite comprobar avances y en caso necesario, ajustar lo que queda por delante.

5.1. Fase de Análisis

5.1.1. Análisis de Requerimientos

Durante las semanas iniciales del desarrollo, mantuve conversaciones informales con varios propietarios de clínicas y centros de fisioterapia, entrenamiento y estética. En estos encuentros me explicaron sus procesos actuales de gestión de citas, las herramientas que utilizaban desde agendas en papel hasta plataformas como Timp o Booksy y, sobre todo, los principales problemas que encontraban en su día a día.

Uno de los puntos más repetidos fue el tiempo que dedicaban a responder mensajes de WhatsApp, incluso fuera del horario laboral. Esta situación les generaba frustración, sensación de descontrol y, en ocasiones, pérdida de clientes por no poder atender a tiempo. Otros problemas comunes eran la falta de automatización en los recordatorios, la duplicación de citas por malentendidos y la dificultad para tener una visión clara de su agenda sin depender de múltiples herramientas.

Estas conversaciones reales me permitieron identificar una serie de requerimientos clave que debía cubrir el sistema:

- Automatizar la atención por WhatsApp sin perder el trato personalizado.
- Permitir la gestión completa de las citas desde un panel web sencillo.

- Ahorrar tiempo y reducir errores en la gestión del día a día.
- Organizar los datos por negocio (clínica) de forma segura y privada.
- Adaptarse a usuarios no técnicos con una experiencia clara y accesible.

Este análisis fue el punto de partida para construir los casos de uso y requisitos funcionales, asegurando que cada funcionalidad tuviera un propósito claro y resolviera un problema real.

5.1.2. Casos de Uso

Para entender en profundidad cómo funciona el sistema en la práctica, en este apartado se detallan los casos de uso desde el punto de vista de cada uno de los actores que interactúan con él.

El sistema cuenta con dos actores principales:

- **Cliente:** es el usuario final que desea reservar una cita en la clínica. No necesita instalar ninguna aplicación ni registrarse manualmente, ya que toda la interacción se realiza mediante mensajes de WhatsApp, a través de un asistente conversacional. Desde ese canal puede registrarse, pedir cita, resolver dudas frecuentes o confirmar su asistencia.
- **Administrador del sistema:** es el propietario o personal autorizado de la clínica. Accede a un panel web desde donde puede gestionar citas, clientes y empleados. Este panel proporciona una visión global de la actividad del centro y permite realizar ajustes de forma rápida y centralizada.

Cada caso de uso se ha documentado mediante una tabla descriptiva acompañada de una captura real del sistema (disponible en el **Anexo 3**). Este enfoque permite no solo entender qué funcionalidades ofrece el sistema, sino también visualizar cómo se implementan y ejecutan en un entorno real. Se ha optado por una presentación práctica y aplicada, orientada a mostrar de forma concreta el funcionamiento del producto final.

Figura 3. Actores del sistema

5.1.2.1. Cliente

El cliente o paciente final, es decir, la persona que desea reservar una cita con la clínica. Este usuario no necesita descargar ninguna app ni registrarse previamente: toda la interacción ocurre desde su propio WhatsApp, a través de un chatbot conversacional que automatiza el proceso de reserva.

En la siguiente imagen se representa de forma visual el recorrido habitual que puede realizar un cliente dentro del sistema. Todo parte desde el primer mensaje que envía por WhatsApp, y desde ahí, según su estado (nuevo o ya registrado), se activa el flujo de registro o se continúa directamente con la reserva de cita. Además, si el usuario no quiere reservar, también puede consultar dudas o acceder a información frecuente sin necesidad de intervención humana. Por otro lado, cuando ya tiene una cita programada, el sistema le enviará un recordatorio automático 24h antes, desde el que puede confirmar o cancelar su asistencia con un simple mensaje.

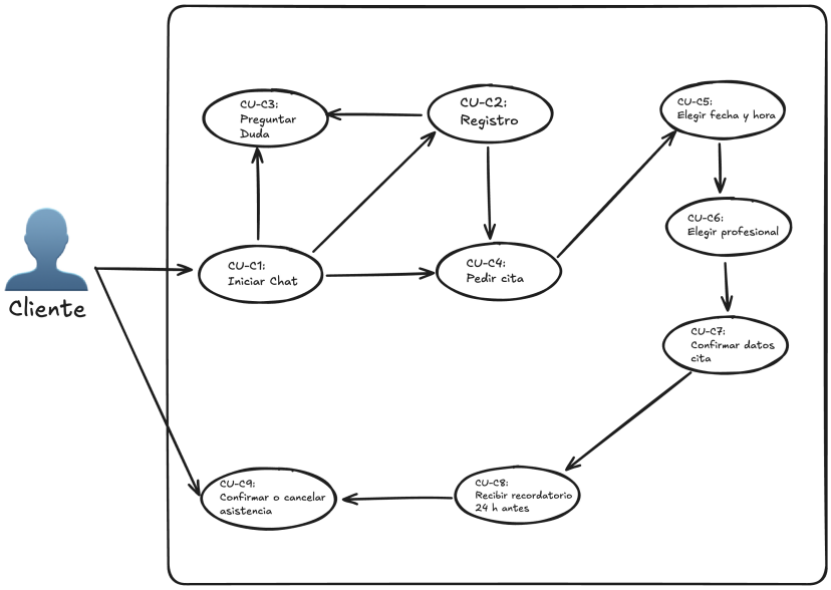

Figura 4. Casos de Uso Cliente

Como hemos comentado anteriormente los casos de uso de la perspectiva del Cliente se detallan en el Anexo 3.A, es decir, de la persona que interactúa con el chatbot por WhatsApp para gestionar sus citas. Estos casos de uso recogen las distintas acciones que puede realizar un paciente a través de la conversación automatizada: desde iniciar un nuevo chat, registrarse por primera vez, hasta confirmar una cita o resolver una duda puntual. Para facilitar la comprensión, cada caso se acompaña de una tabla explicativa y una imagen real del sistema, lo que permite visualizar con claridad cómo se comporta la solución en cada escenario.

5.1.2.2.Administrador del sistema

Para entender cómo interactúa el administrador con el sistema, lo más práctico es visualizarlo en forma de flujo. En la siguiente imagen se representan todos los casos de uso del panel web, organizados desde el primer paso: iniciar sesión. A partir de ahí, el usuario puede consultar o modificar citas, gestionar empleados y clientes o incluso intervenir en el funcionamiento del bot en situaciones puntuales. Esta visión general permite hacerse una idea rápida de todo lo que puede hacer el negocio desde su espacio de gestión.

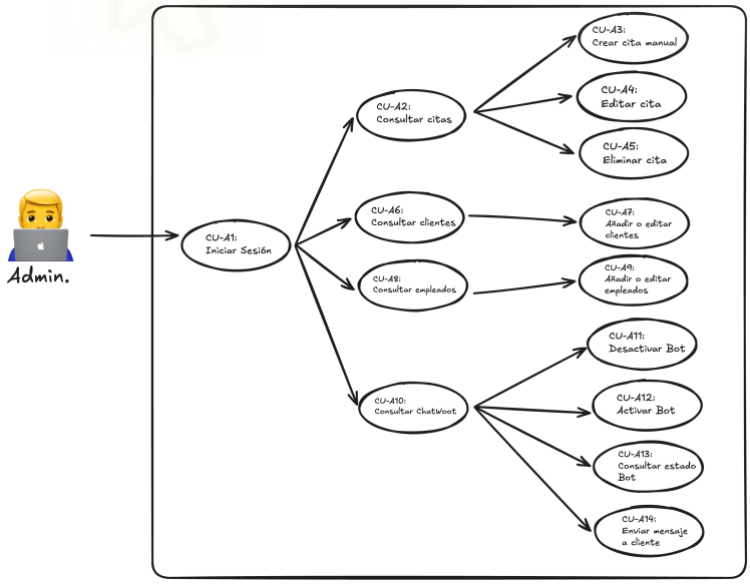

Figura 5. Casos de Uso del Admin

Todos los casos de uso del Administrador vienen detallados en el Anexo 3.B, consultar para mayor detalle.

5.1.3. Requisitos Funcionales

Los requisitos funcionales definen las capacidades esenciales que debe cumplir el sistema para dar respuesta a las necesidades del negocio y de sus usuarios. En este proyecto se han identificado dos subsistemas claramente diferenciados: el chatbot conversacional por WhatsApp y el panel de administración web. Cada uno presenta funcionalidades específicas que permiten automatizar el proceso de reserva de citas, gestionar los datos del negocio y ofrecer una experiencia fluida tanto al cliente como al personal de la clínica. A continuación, se describen los requisitos funcionales más relevantes de cada subsistema.

5.1.3.1. Subsistema WhatsApp

RF-W1 - Registro automático de nuevos clientes
El sistema debe identificar si el número de teléfono que inicia la conversación es nuevo. En ese caso, solicitará de forma conversacional el nombre completo y el correo electrónico del cliente, guardando esta información en la base de datos Supabase asociada al identificador telefónico.

Tabla 4. Requisitos Funcionales RF-W1: Registro automático de nuevos clientes

RF-W2 – Detección de intención con IA
El sistema debe analizar cada mensaje recibido mediante un modelo de lenguaje (LLM) y clasificarlo como intención de reservar una cita o pregunta frecuente. Esta lógica está separada del resto del sistema y puede adaptarse mediante prompt.

Tabla 5. Requisitos Funcionales RF-W2: Detección de intención con IA

RF-W3 – Proceso de reserva de citas por chat

El bot debe guiar paso a paso al cliente durante la reserva: preguntando la fecha, validando el formato, mostrando los profesionales disponibles ese día y registrando la cita en la base de datos con estado "pending".

Tabla 6. Requisitos Funcionales RF-W3: Proceso de reserva de citas por chat

RF-W4 – Envío automático de recordatorios 24h antes

El sistema debe revisar cada día las citas agendadas para el día siguiente y enviar automáticamente un recordatorio por WhatsApp. El mensaje debe permitir al cliente confirmar o cancelar la cita desde el propio chat.

Tabla 7. Requisitos Funcionales RF-W4: Envío automático de recordatorios 24h antes

RF-W5 – Transferencia de la conversación a agente humano

El sistema debe permitir que el cliente sea atendido manualmente y debe mantenerse el historial completo de la conversación.

Tabla 8. Requisitos Funcionales RF-W5: Transferencia de la conversación a agente humano

5.1.3.2. Subsistema Panel Web

RF-A1 – Inicio de sesión seguro con autenticación doble factor
El acceso al panel exigirá autenticación de doble factor y garantizar que cada usuario solo vea la información de su propia clínica.

Tabla 9. Requisitos Funcionales RF-A1: Inicio de sesión seguro con autenticación doble factor

RF-A2 – Gestión completa de citas
El administrador debe poder ver todas las citas de su clínica en un calendario o lista, modificarlas, cancelarlas o crearlas manualmente. Los cambios deben sincronizarse en tiempo real con la base de datos y el bot.

Tabla 10. Requisitos Funcionales RF-A2: Gestión completa de citas

RF-A3 – Gestión de clientes y empleados
Desde la interfaz web debe poder visualizarse, editarse y eliminar la información de clientes registrados, así como dar de alta o baja a empleados que aparecerán en el flujo de selección del bot.

Tabla 11. Requisitos Funcionales RF-A2: Gestión de clientes y empleados

RF-A4 – Personalización y modificación de prompt
El panel debe permitir modificar el nombre del negocio para atención personalizada y subir un archivo .txt que se almacene en la base de datos para adaptar el comportamiento del chatbot.

Tabla 12. Requisitos Funcionales RF-A3: Personalización y modificación de prompt

5.2. Planificación

La planificación ha sido una pieza clave en el desarrollo de este proyecto. Desde las fases iniciales se adoptó un enfoque estructurado que permitiera definir hitos claros, estimar el tiempo y los recursos necesarios, y evaluar la viabilidad tanto técnica como económica de la solución. Para ello, se ha utilizado un diagrama de Gantt como herramienta principal de organización, junto a estimaciones de costes y recursos que ayudan a contextualizar el alcance del trabajo realizado. A lo largo de este capítulo se expone cómo se ha estructurado el proyecto, qué herramientas se han empleado para su gestión y qué previsión económica se contempla en caso de llevarlo al mercado real.

5.2.1. Diagrama de Gantt

Para organizar el proyecto he optado por un diagrama de Gantt por dos motivos. El primero es académico, en la asignatura de *Gestión de Proyectos de Ingeniería del Software* descubrí esta forma de planificar entre otras y me pareció la más clara para observar en un vistazo que tareas dependen de cuáles y cuánto tiempo ocupa cada fase. El segundo motivo es operativo, ya estoy familiarizado con **GanttProject**, una herramienta libre y open-source que aprendí a manejar en la misma asignatura. Esa experiencia previa me ha permitido avanzar más rápido sin invertir tiempo extra en aprender un nuevo software, manteniendo así el foco en el proyecto.

Este formato encaja perfectamente con la dinámica de un TFG, permite colocar hitos como "Arquitectura probada" o "Pruebas completadas", asignarles una fecha realista y ver si se están cumpliendo los objetivos o por el contrario hay que cambiar la planificación por exceso de optimismo o pesimismo. En mi caso, el desarrollo ha seguido un ritmo bastante constante, y aunque siempre pueden surgir imprevistos, el hecho de trabajar solo y conocer bien el alcance del proyecto me ha permitido adaptarme con facilidad a los cambios y cumplir los hitos sin grandes desviaciones.

Un Gantt sencillo evita la sobrecarga de métodos ágiles más formales y al mismo tiempo ser disciplinado con los plazos.

1. **Fase de planificación**
    - Definición del tema, objetivos y estudio de viabilidad.
    - Revisión preliminar de literatura y alcance inicial.
    - *Hitos H0 y H1* marcan la validación del tema y la factibilidad técnica.
2. **Análisis de requisitos**
    - Identificación de necesidades de base de datos, chatbot, LLM y CRM.
    - Culmina con el *Hito H2*, que confirma que el proyecto tiene un conjunto de requisitos coherentes.
3. **Diseño del sistema**
    - Arquitectura del chatbot, flujos conversacionales e interfaz web.
    - Diseño de las APIs y el modelo de datos.
    - Finaliza con el *Hito H3*: la arquitectura queda aprobada y congelada para empezar a codificar.
4. **Desarrollo**
    - Implementación del chatbot y su integración con el LLM.
    - Desarrollo simultáneo del backend, la base de datos y la interfaz web.
    - El *Hito H4* señala que el producto mínimo viable está listo para someterse a pruebas.
5. **Fase de pruebas**
    - Verificación de prompts, integración y funcionalidades clave.
    - *Hito H5* certifica que el sistema funciona según lo esperado.
6. **Documentación**
    - Redacción de la documentación técnica, guía de usuario y pruebas realizadas.
    - Concluye con el *Hito H6*, que deja la parte técnica cerrada.

7. **Revisión final del TFG**
    - Redacción, correcciones del tutor y preparación de la presentación.
    - El *Hito H7* coincide con la entrega definitiva.

Para mayor detalle acudir al Anexo 2 para ver el esquema completo del diagrama de Gantt realizado en la herramienta de GanttProject.

5.2.2. Recursos técnicos

A la hora de elegir herramientas y servicios he intentado usar lo que conozco y aprovechar planes gratuitos siempre que sea posible. El resultado es un ecosistema moderadamente moderno, pero sobre todo práctico, que me permite dedicar la mayor parte del tiempo programando la lógica de negocio y no a configurar servidores.

En cuanto a entorno de desarrollo he utilizado Cursor como se ha comentado anteriormente, un fork de Visual Studio Code que permite codificar mucho más rápido con las sugerencias de código con inteligencia artificial y su modo agente que conoce perfectamente todo el contexto y es muy útil para comenzar con nuevas funciones para la aplicación, mostrando cambios que puedes testear antes de aceptar y que sea aplicado en tu código.

Para ver las configuraciones y LLM utilizados en Cursor acudir al **Anexo 1**.

Todo el proyecto gira en torno a **GitHub**: mantengo dos repositorios separados, uno para la web y otro para el bot. Cuando hago push en el repositorio de la web, Netlify detecta el cambio, compila el proyecto y publica la nueva versión en cuestión de segundos. Si el push va al repositorio del chatbot, Railway reconstruye la imagen y vuelve a levantar el backend automáticamente.

Por su parte, la base de datos reside en Supabase, que me ofrece un PostgreSQL gestionado con copias de seguridad automáticas y, además, genera una API REST lista para usar nada más crear las tablas.

Aunque no se ha montado un sistema completo de CI/CD personalizado, este flujo automático de despliegue tras cada commit cumple de forma sencilla esa función: escribo código, hago push y el sistema se actualiza solo. Para este tipo

de proyecto, este enfoque ligero y sin configuración extra ha resultado más que suficiente.

En cuanto a la capa de IA, el sistema se apoya principalmente en el **modelo GPT-4o-mini de OpenAI [35].** La elección de OpenAI como proveedor se debe a varios factores: es una de las plataformas más consolidadas en el ámbito de los modelos de lenguaje, ofrece una API estable y bien documentada, y cuenta con un ecosistema muy activo que facilita la integración en proyectos reales.

Dentro de las opciones disponibles, se seleccionó GPT-4o-mini por ofrecer un equilibrio óptimo entre calidad de respuesta, velocidad y coste.

Desde la plataforma de OpenAI introduje 5 $, que es la recarga mínima, y a día de hoy he gastado menos de 0,20 $ tras un uso intensivo. Esto demuestra que, incluso en la fase de pruebas, el coste es muy bajo y el modelo se puede considerar perfectamente viable para una solución SaaS orientada a pymes.

En resumen: stack barato, rendimiento más que suficiente y sin tiempo perdido instalando servidores.

5.2.3. Estimación de costes

Una de las grandes ventajas del stack elegido es que, mientras el proyecto sigue en modo "piloto", se puede funcionar en la capa gratuita de los proveedores. Aun así, conviene mostrar las cifras para demostrar la futura escalabilidad desde el punto de vista económico.

En primer lugar, hemos decidido seleccionar el nicho de fisioterapia para ser más precisos, se estiman en España más de 10.000 clínicas por eso he decidido hacer 4 niveles de escalabilidad.

- **MVP Académico** seria la versión actual sin clientes reales y como hemos dicho tendría un gasto mínimo de los tokens del LLM.
- **Nivel 1** para los primeros clientes de la herramienta, serían hasta 50 clínicas con un gasto aproximado de 962€.
- **Nivel 2** con unas 500 clínicas ya serían un nivel alto, manteniendo los costes muy controlados sobre los 9461€ al mes.

- **Nivel 3** sería uno de los escenarios más optimistas con 2.500 clínicas se tendrían unos gastos de 47130 € mensuales.

Servicio / Proveedor	MVP Académico	Nivel 1	Nivel 2	Nivel 3
Netlify (web)	0	0	0	19
Railway (backend bot)	0	5	30	50
Kinde Auth	0	0	0	25
Supabase (PostgreSQL)	0	0	25	100
WhatsApp API	0	900	9000	45000
GPT-4o-mini	0,2	36,6	366	1835
Chatwoot	0	19	39	100
Dominio + email	-	1	1	1
Total	0,2 €	961,6 €	9461 €	47130 €

Tabla 13. Estimación de Costes

Esto sería un cálculo aproximado de los gastos mensuales según el número de clínicas, ahora vamos a explicar la tabla para mostrar de donde salen todos los datos.

La media de citas por fisio al día suele rondar las 15 por empleado y de media se tienen entre 3 y 4 empleados. Por lo que podríamos estimar aproximadamente 1.000 citas por clínica al mes (22 días laborables).

Estimando una media de 10 mensajes y 5 peticiones de IA por cita:

Nivel 1 → 1.000 citas x 50 clínicas X 5 peticiones = 25.000 peticiones

Nivel 2 → 1.000 citas x 500 clínicas X 5 peticiones = 2.500.000 peticiones

Nivel 3 → 1.000 citas x 2500 clínicas X 5 peticiones = 12.500.000 peticiones

Asumiendo una media de tokens utilizados en cada llamada con el prompt de detección con 150 y con el prompt de respuesta de 250.

Tokens por cita

2 x 150 + 3 x 250 = 1050 tokens.

Una vez sabido el coste en tokens por cita, hacemos el cálculo para mayo de 2025 con GPT-4o-mini son **0,75 $ por millón de tokens** y con el cambio actual de 1$ = 0,93 €.

Nivel	Clínicas	Citas/mes	Tokens/mes	Coste mensual
MVP	1	1000	105200	0,74 €
Nivel 1	50	50000	52600000	36,60 €
Nivel 2	500	500000	526000000	366 €
Nivel 3	2500	2500000	2630000000	1.835 €

Tabla 14. Coste IA por Número de Clínicas

En cuanto a los gastos de la API de meta, actualmente son gratis de manera ilimitada siempre y cuando el cliente contacte con nosotros. Sin embargo, si queremos nosotros responder pasadas 24h tras la última interacción, si queremos una utilidad de negocio son 0,018 € por conversación iniciada y 0,057 € en comunicaciones de marketing.

El único momento en el que se utiliza este sistema es para los recordatorios ya que normalmente se necesita iniciar la conversación con el cliente pasadas esas 24h.

Nivel	Clínicas	Citas/mes	Coste mensual
MVP	0	100	0 €
Nivel 1	50	50000	900 €
Nivel 2	500	500000	9.000 €
Nivel 3	2500	2500000	45.000 €

Tabla 15. Coste API Meta por Número de Clínicas

El resto de los gastos son fácilmente entendibles, se ha visto los diferentes precios por servicios y un aproximado de que versión del plan de pago encaja con el volumen de clínicas planteado en cada nivel.

Por último, he querido mostrar en un gráfico simple, la rentabilidad del proyecto y los grandes márgenes en el software como servicio (>60%). Con un plan simple de 50 € al mes por clínica un precio significativamente inferior a la mayoría de las soluciones similares tendríamos esta gráfica.

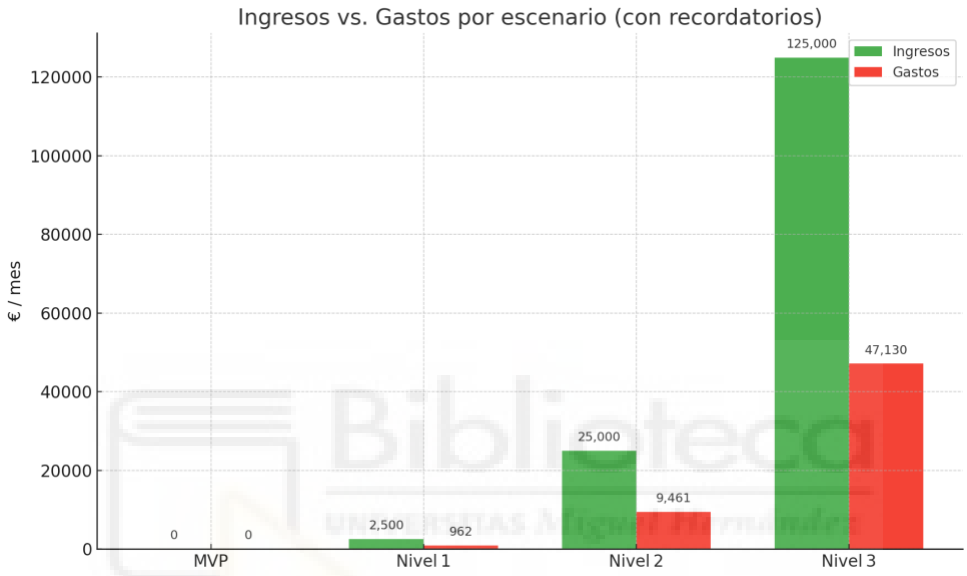

Figura 6. Ingresos vs Gastos por Escenario

Como podemos ver el coste es moderado con respecto al potencial ingreso, siendo a ese precio de las soluciones más baratas del mercado. Es evidente, que este producto quizá actualmente no cumple con todos los requerimientos del sector, pero quería mostrar el camino a la escalabilidad, y como en el futuro veremos nuevas pequeñas empresas de software compitiendo con las ya establecidas gracias a sus reducidos costos de infraestructura y su capacidad de innovación.

5.2.4. Recursos humanos

Este trabajo de fin de grado lo he codificado en solitario, dedicando 8h diarias desde principios de febrero hasta finales de mayo más o menos unas 800h entre investigación, diseño, programación full-stack, pruebas manuales, redacción completa de la memoria y preparación de la memoria.

Tarifa del personal implicado	
Apellido	Valladares Sánchez
Nombre	Víctor
Puesto	Ingeniero de Software Full-Stack
Tarifa/Hora	30 €

Tabla 16. Tarifa Personal

He estimado 30€ de tarifa por hora en total serían unos 24.000€ para el proyecto completo en costes de recursos humanos como Ingeniero de Software Full-Stack con tareas como análisis de requisitos, diseño de arquitectura, desarrollo Full-Stack, pruebas y documentación.

Presupuesto Total Recursos Humanos	
Total Meses	4
Total Semanas	16
Total Horas	800
Total	24.000 €

Tabla 17. Presupuesto Total RRHH

5.2.5. Plan de negocio

Aunque este proyecto nace como un TFG, he querido ir un paso más allá y elaborar una previsión financiera a cinco años para entender si realmente tiene sentido ser comercializado.

Según la estimación de clínicas de privadas de fisioterapia en España existen unas 10.000 y se propone un crecimiento sostenido empezando con 50 clínicas en 2026, es decir, un 1% del mercado hasta una ambiciosa cifra del 45% del mercado con 4.500 clínicas.

Para simplificar los números hemos fijado el precio de **50€/mes por clínica** que serían 600€ al año. En los costes tendríamos los que ya hemos visto a nivel tecnológico de infraestructura, WhatsApp y LLM. Pero para hacer funcionar esta idea, no basta con la tecnología se necesita un equipo. Por eso he añadido dos perfiles clave al plan:

- Una persona encargada de **marketing y ventas**, con un salario de **26.000€/año**.
- Otra dedicada a atención al **cliente, onboarding y tareas administrativas**, con un salario de **25.000€/año**.

A esto se suma mi propio sueldo como ingeniero responsable del sistema, encargado de su mantenimiento técnico, soporte, monitorización y mejora continua. Este coste se entiende como parte del gasto operativo, ya que el desarrollo inicial del producto ya ha sido completado.

También se incluyen los **gastos generales** asociados al funcionamiento básico de la empresa: oficina, suministros y conexión a internet.

Todos estos cálculos mencionados se pueden observar a continuación en la tabla 18.

Biblioteca
UNIVERSITAS Miguel Hernández

DATOS PARA CUENTA PERDIDAS Y GANANCIAS	2026	2027	2028	2029	2030	2031
Clinicas Estimadas España	10.000					
Clinicas clientes	50	200	500	1200	2500	4500
% Clinicas	1%	2%	5%	12%	25%	45%

GASTOS PERSONAL	
Numero personas administracion	1
Ventas	1
Sueldo CEO	30.000
Sueldo Atencion a clientes- Administracion	25.000
Sueldo Comercial	26.000
Total costes salariales	81.000
Costes mantenimiento	2.000
Tiempo pago clientes	0
Pago a proveedores	1
Impuestos	25%

			Valor	
VALORACION DE LA EMPRESA :				
Precio SaaS :		50 x 12 =	600	
Costes fijos	Dominio y backend		170	
	Chatwoot		100	
	Supabase		100	
Gastos Variables			Cantidad de citas al mes	Coste unitario
	Coste LLM		1.000	0,74
	Coste Whatsapp		1.000	0,018

Tabla 18. Datos Base para Proyección de Clientes y Costes

BUSINESS PLAN						
FASE	1	1	1	2	2	3
	2026	**2027**	**2028**	**2029**	**2030**	**2031**
VENTAS TOTALES	**30.000**	**120.000**	**300.000**	**720.000**	**1.500.000**	**2.700.000**
Ventas licencias	30.000	120.000	300.000	720.000	1.500.000	2.700.000
COSTES TOTALES						
Coste Servicios Infraestructura	370	370	370	370	370	370
Coste LLM	444	1.776	4.440	10.656	22.200	39.960
Coste Whatsapp	10.800	43.200	108.000	259.200	540.000	972.000
Inversiones	0	0	0	0	0	0
Costes de transporte	0	0	0	0	0	0
Margen Bruto	**18.756**	**75.024**	**187.560**	**450.144**	**937.800**	**1.688.040**
Margen bruto %	62,5%	62,5%	62,5%	62,5%	62,5%	62,5%
Sueldos	81.000	81.000	81.000	190.000	190.000	190.000
Alquiler oficina	12.000	12.000	12.000	12.000	12.000	12.000
Luz, agua e internet	300	300	300	300	300	300
Benef. Oper.	-74.544	-18.276	94.260	247.844	735.500	1.485.740
Beneficio antes Intereses e Impuestos	-74.544	-18.276	94.260	247.844	735.500	1.485.740
Margen Bruto	-248%	-15%	31%	34%	49%	55%
Int a corto						
Interest						
BAI y G. Amort						
BAI	-74.544	-18.276	94.260	247.844	735.500	1.485.740
Taxes	0	0	360	61.961	183.875	371.435
BDI	**-74.544**	**-18.276**	**93.900**	**185.883**	**551.625**	**1.114.305**

Tabla 19. Planificación Financiera (2026-2031)

Como podemos ver en la tabla 19, el primer año serían -74,544€ de perdidas, el segundo año tampoco se alcanzaría aún el punto de equilibrio, con un resultado negativo, aunque mucho más reducido.

Con esta estructura, el primer año cerraríamos con pérdidas de -74.544 €, y el segundo aún estaríamos por debajo del punto de equilibrio (-18.276 €). Es a partir del tercer año (2028) cuando el negocio empezaría a generar beneficios reales: 93.900 €, que se reinvertirían para escalar la operación y crecer más rápido.

Esa fase de crecimiento abre la puerta a la fase 2, donde pasaríamos de 500 a 1.200 clínicas, y los números empiezan a coger forma: un EBITDA de 247.844 € y un beneficio neto de 185.883 €. A partir de ahí, el crecimiento es prácticamente exponencial. En 2031, el último año previsto en este plan, la empresa facturaría 2,7 millones de euros y generaría un beneficio neto superior a 1,1 millones.

Como puede verse, es un modelo que apuesta por un crecimiento progresivo y controlado, pero con mucho potencial una vez se supera el punto de equilibrio. Tener pérdidas en los primeros años es algo habitual en startups tecnológicas de rápido crecimiento.

5.3. Fase de Diseño

5.3.1. Diseño general del sistema

El diseño del sistema no partió de una estructura cerrada desde el inicio, sino que fue el resultado de un proceso de reflexión y exploración constante. Al explicar la idea a personas de mi entorno, tanto del ámbito técnico como no técnico, fui dándome cuenta de cuáles eran los verdaderos retos del proyecto. Esta práctica de verbalizar el problema me permitió ordenar mis ideas, detectar incoherencias y tomar decisiones de diseño más acertadas.

Desde el primer momento tuve clara la visión modular del sistema: dividirlo en bloques independientes pero conectados entre sí. Esta separación permite escalar, mantener y adaptar el sistema más fácilmente, y ha sido clave para su desarrollo. También concebí el sistema como una solución multitenant desde el principio, ya que mi objetivo siempre fue construir un producto SaaS que pudiera

ser utilizado por múltiples clínicas sin necesidad de replicar el código o la base de datos.

Los pilares fundamentales que han guiado el diseño han sido: escalabilidad, para poder crecer sin reescribir el sistema; solidez, para asegurar que cada parte funcione de forma fiable; modularidad, para mantener el control del código y facilitar futuras ampliaciones; y, por último, aprovechar herramientas con planes gratuitos, especialmente útil en esta fase inicial del proyecto.

La experiencia previa con herramientas como Supabase, Railway y Next.js también influyó notablemente en las decisiones de diseño. Conocía sus ventajas y limitaciones, y eso me permitió integrarlas de forma eficiente sin necesidad de invertir demasiado tiempo en aprendizaje. Además, el contenido disponible en comunidades y canales de YouTube enfocados en el desarrollo de productos SaaS sirvió de inspiración para consolidar un diseño moderno y alineado con las mejores prácticas actuales.

5.3.2. Diseño del módulo de chatbot

Desde el principio, la arquitectura se planteó de forma modular. Cada función relevante del bot (registro, cita, FAQ, recordatorio…) debía implementarse como un flujo separado para facilitar su mantenimiento y evolución. Inicialmente, intenté que la IA se encargara por completo de gestionar reservas, leyendo la base de datos y generando respuestas dinámicas. Sin embargo, los errores frecuentes y la falta de control sobre el flujo llevaron a un rediseño importante. Finalmente opté por una solución híbrida: el modelo de lenguaje clasifica la intención del usuario y, a partir de ahí, el control pasa a un flujo programado por mí. Esto garantiza solidez, control y una experiencia mucho más coherente.

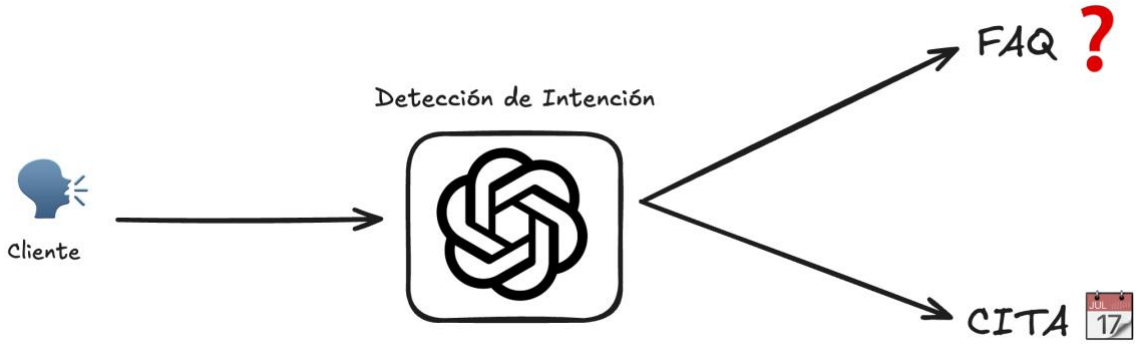

Figura 7. Detección de Intención

Una vez que el modelo de lenguaje detecta la intención del mensaje (como muestra la Figura 7), el sistema se ramifica hacia dos grandes bloques de flujo: FAQ o Cita. Ambos están diseñados por separado y gestionados íntegramente por código, aunque parten de la detección de intención realizada por IA. A continuación, se detalla el diseño de cada uno de ellos.

5.3.2.1. Diseño del flujo FAQ

Cuando el mensaje del cliente no tiene relación con una reserva (por ejemplo, saluda, da las gracias o pregunta por el horario), el sistema activa el flujo faqFlow.ts. En este caso, se utiliza un segundo prompt diseñado para que la IA actúe como un asistente capaz de responder dudas frecuentes de forma clara, útil y natural. Este prompt está personalizado con la información de cada clínica, como su nombre, teléfono, dirección, tratamientos, horarios o condiciones.

Este flujo no tiene lógica programática compleja, ya que se limita a enviar el mensaje al modelo de lenguaje y mostrar la respuesta al usuario como se muestra en la figura 8. Se ha diseñado así para maximizar la naturalidad de la conversación y permitir que la respuesta se adapte al contexto del negocio.

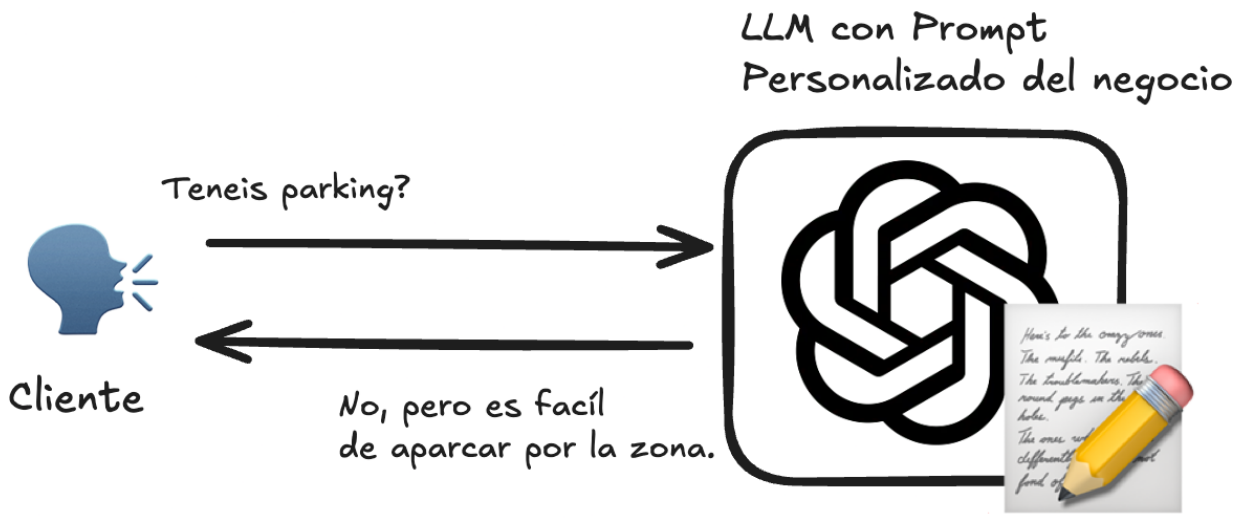

Figura 8. Diseño Flujo FAQ

5.3.2.2. Diseño del flujo de cita

El flujo de reserva de citas es el núcleo del sistema y, por tanto, el más elaborado a nivel de diseño. Tras identificar que el usuario desea agendar una cita, el bot entra en una conversación guiada donde cada paso es validado de forma independiente. El objetivo es minimizar errores, mantener el control del flujo y asegurar que la cita que se registra es válida y completa.

El flujo está dividido en las siguientes etapas:

1. Solicitud de fecha: el sistema solicita al cliente que indique cuándo desea acudir. Este mensaje puede ser informal o en lenguaje natural ("mañana por la tarde", "el próximo lunes", etc.).

2. Traducción de fecha: el mensaje se envía al modelo de lenguaje (LLM), que interpreta y extrae una fecha concreta en formato legible para el sistema, "2025-05-27 17:00:00+00".

3. Retorno fecha con el formato correcto: la IA devuelve una fecha normalizada que el sistema puede procesar directamente.

4. Consulta en la base de datos: el backend lanza una consulta SQL a Supabase para comprobar si en esa fecha y franja horaria hay disponibilidad para nuevos pacientes.

5. Confirmación de disponibilidad: si existe disponibilidad, se responde al cliente con un mensaje positivo y se avanza al siguiente paso. Si no, se solicita una nueva fecha.

6. Elección de profesional: si hay más de un profesional disponible en ese horario, se muestra una lista con los nombres para que el cliente elija su preferido.

7. Confirmación final: una vez elegidos todos los datos (fecha, hora y profesional), el bot presenta un resumen de la cita y ofrece tres opciones mediante botones: confirmar, cambiar la fecha u cancelar.

8. Registro: si el cliente confirma, se guarda la cita en Supabase y se cierra el flujo con un mensaje de confirmación y agradecimiento.

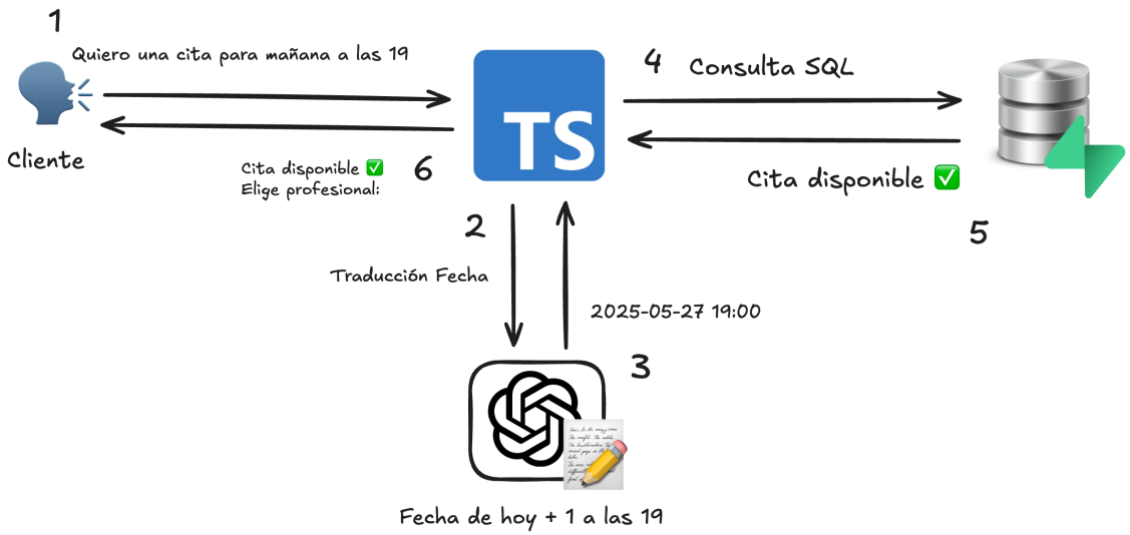

Figura 9. Flujo de Cita

Como podemos ver en el diagrama de la Figura 9, se observa la parte principal del proceso del flujo de cita, sin embargo, tras el paso 6 quedaría que el cliente elija el profesional, una vez seleccionado el cliente recibe un resumen con los datos de su reserva y la opción de confirmar la reserva, cambiar la fecha o cancelar el proceso.

En caso de que la IA no entienda alguno de los pasos, se notifica al cliente con un mensaje amable. Si esto ocurre tres o más veces, se sugiere contactar con el negocio por teléfono para continuar la gestión.

5.3.3. Diseño del módulo integración LLM

Uno de los pilares del sistema es el uso de modelos de lenguaje natural (LLM) para interpretar los mensajes entrantes. Desde el inicio, este módulo se concibió como una pieza independiente, con el objetivo de mantener la lógica de inteligencia artificial separada del resto de funcionalidades. Esto permitiría reemplazar el proveedor de IA en cualquier momento (por ejemplo, por coste, rendimiento o privacidad) sin reescribir todo el sistema.

El diseño se basó en los siguientes principios:

- **Desacoplamiento:** El módulo debía ser autónomo, sin acoplamientos directos con flujos del bot o el backend, permitiendo testearlo o actualizarlo de forma independiente.

- **Extensibilidad:** Se planteó desde el inicio que el sistema pudiera trabajar con distintos proveedores de IA (OpenAI, DeepSeek, etc.) sin necesidad de modificar su comportamiento externo.
- **Abstracción de complejidad:** Toda la lógica técnica para conectar con las APIs debía ocultarse al resto del sistema. Desde fuera, solo se requería enviar un mensaje y obtener una respuesta, sin preocuparse del formato o proveedor.

El comportamiento del módulo puede resumirse en el siguiente flujo conceptual de la figura 10:

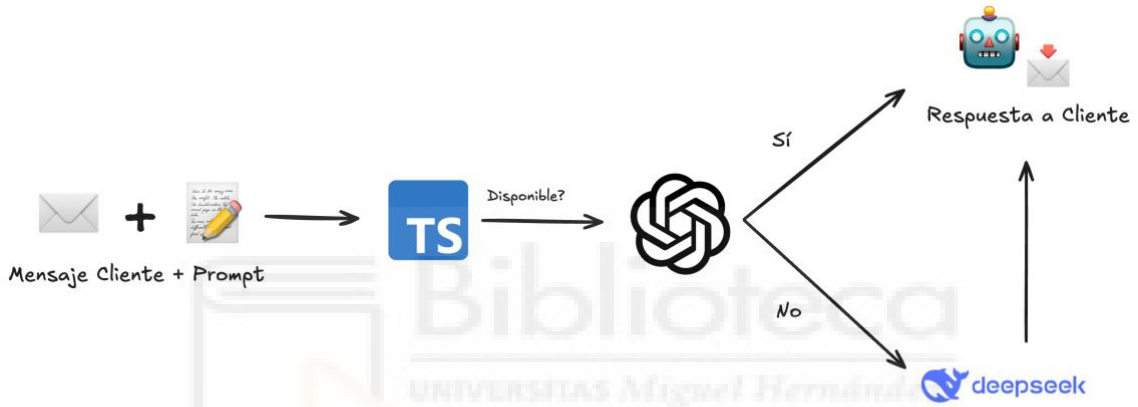

Figura 10. Diseño modular proveedor LLM

Este enfoque permite que la arquitectura evolucione con el tiempo, integrando nuevos modelos o ajustando los prompts por negocio sin romper la lógica existente.

5.3.4. Diseño de la interfaz web

El panel web de administración ha sido diseñado pensando en la claridad, rapidez de uso y modularidad. Su objetivo principal es ofrecer a las clínicas una herramienta intuitiva desde la cual puedan gestionar todo su flujo de citas, clientes y empleados sin necesidad de conocimientos técnicos.

Antes de implementarlo, realicé prototipos sencillos y rápidos con Figma, herramienta que ya conocía gracias a su uso en la asignatura de Diseños Interactivos. Su facilidad de uso me permitió validar rápidamente el diseño y

estructurar la navegación sin necesidad de escribir una sola línea de código. Esto me ayudó a definir los elementos clave, identificar posibles mejoras y asegurar una experiencia coherente para el usuario final.

A continuación, se presentan las principales pantallas de la aplicación web, acompañadas de una breve descripción de su funcionalidad.

5.3.4.1. Diseño Página de Inicio

Esta es la primera pantalla que ve el usuario tras autenticarse. Se ha diseñado como una página limpia y directa, donde se presenta el nombre de la clínica y un acceso rápido al calendario. También incluye un botón destacado que permite gestionar la cuenta, consultar información y cerrar sesión. El enfoque es ofrecer una entrada clara al sistema sin sobrecargar de información.

Figura 11. Diseño Página de Inicio

5.3.4.2. Diseño Calendario de Citas

Esta es la pantalla principal de trabajo del negocio. El calendario muestra de forma visual todas las citas agendadas, organizadas por día y hora. Se puede navegar entre fechas, ver detalles al hacer clic en cada cita y añadir nuevas de forma manual. La interfaz está pensada para ser clara incluso cuando hay muchas reservas en un mismo día, lo que se consigue con colores diferenciados y un diseño bien jerarquizado.

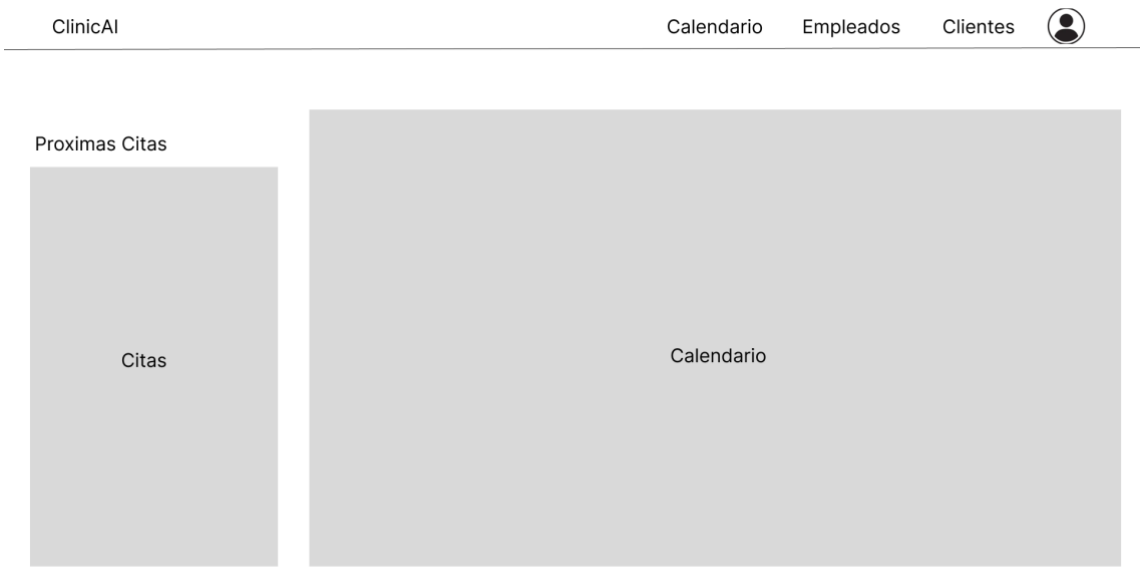

Figura 12. Diseño Calendario de Citas

5.3.4.3. Diseño Gestión de Empleados

Desde esta vista, el personal autorizado puede añadir nuevos empleados, modificar datos de los ya existentes o eliminar aquellos que ya no formen parte del equipo. Cada empleado puede tener un rol definido (por ejemplo: fisioterapeuta, recepcionista…), lo cual puede utilizarse para futuras funcionalidades como la segmentación de citas.

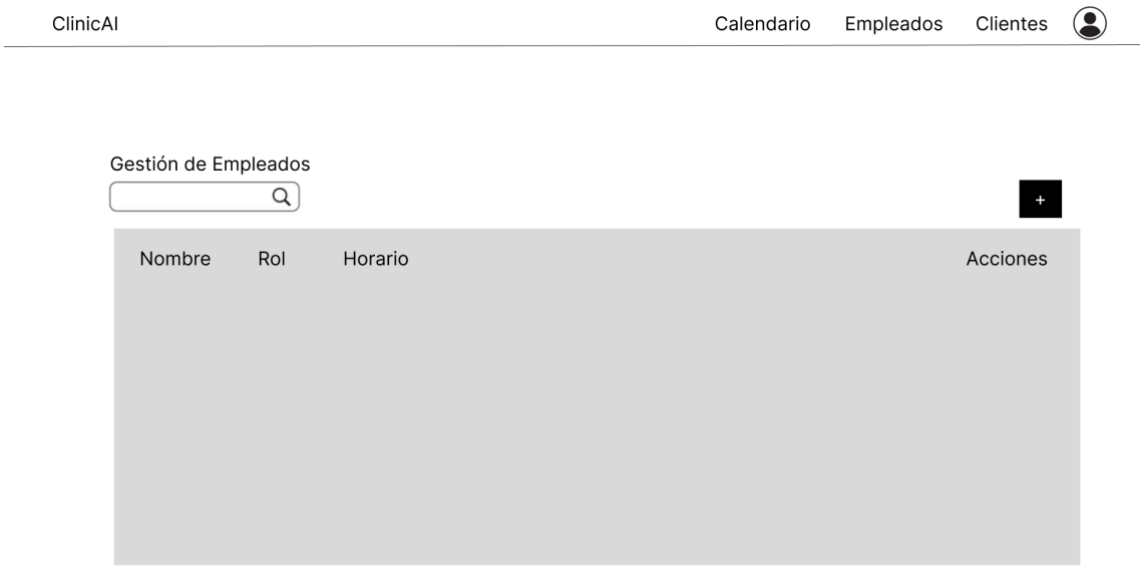

Figura 13. Diseño Gestión de Empleados

5.3.4.4. Diseño Gestión de Clientes

Esta sección permite consultar el historial de clientes, editar sus datos o registrar nuevos manualmente. Aunque la mayoría de las clientes se registran automáticamente al escribir por WhatsApp, esta pantalla ofrece flexibilidad para casos puntuales o para revisar la información de contacto. Además, cada cliente se relaciona con sus citas pasadas y futuras, facilitando así el seguimiento desde la misma interfaz.

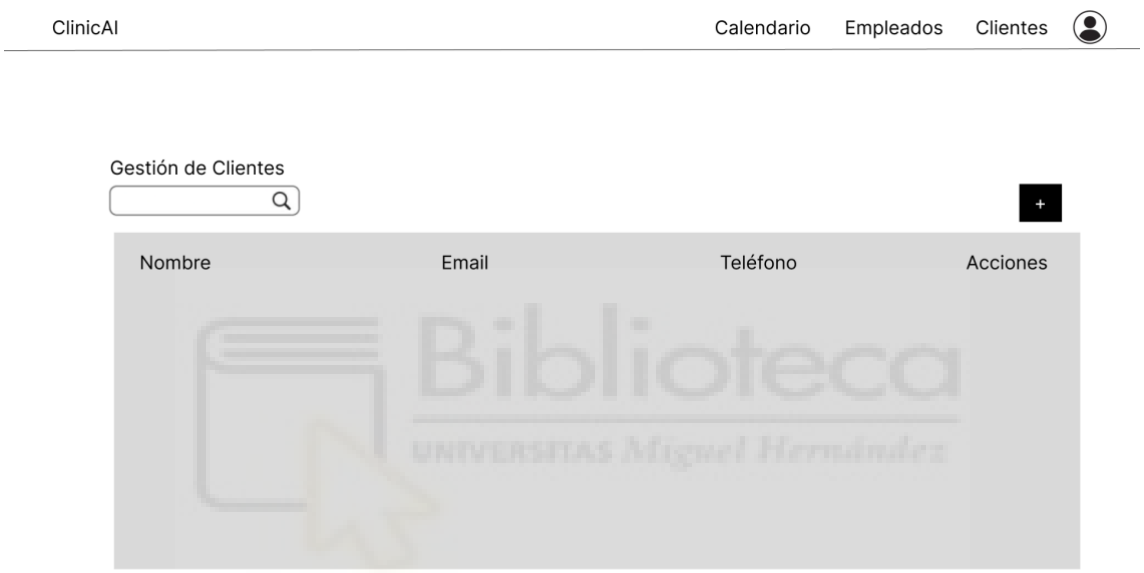

Figura 14. Diseño Gestión de Clientes

5.3.5. Diseño del backend

El diseño del backend se concibió desde el inicio con una arquitectura sencilla pero escalable, que permitiera mantener separados los distintos dominios de la aplicación (clientes, citas, empleados…) y facilitar su evolución en el tiempo.

Uno de los objetivos clave era asegurar que toda la lógica de negocio estuviera centralizada en un único lugar, desde donde pudiera ser consumida por distintos clientes: el panel web, el chatbot de WhatsApp o incluso futuras aplicaciones móviles. Para ello, se diseñó una capa intermedia que expone los recursos necesarios y actúa como puente entre la base de datos y las interfaces de usuario.

Además, desde el principio se adoptó una mentalidad multitenant, de forma que los datos de cada clínica se gestionaran de manera completamente aislada, respetando las políticas de seguridad y privacidad necesarias para un producto SaaS.

A nivel conceptual, el backend se organiza en torno a cuatro módulos principales:

- **Clientes:** Alta, modificación y consulta de pacientes registrados.
- **Empleados:** Gestión del personal disponible en cada centro.
- **Citas:** Creación, modificación, cancelación y confirmación de citas.
- **Configuración:** Datos del negocio, preferencias, claves, etc.

Todas las operaciones están pensadas para operar con un identificador de negocio (tenant_id) que actúa como filtro en cada acción. Así se garantiza que ningún negocio acceda a información de otro.

5.3.5.1. Esquema entidad-relación

Para planificar el diseño del backend, se elaboró un esquema básico de las entidades que se almacenarán posteriormente en la base de datos. Esta visión previa ayuda a identificar las relaciones clave y definir una estructura coherente para las futuras operaciones CRUD.

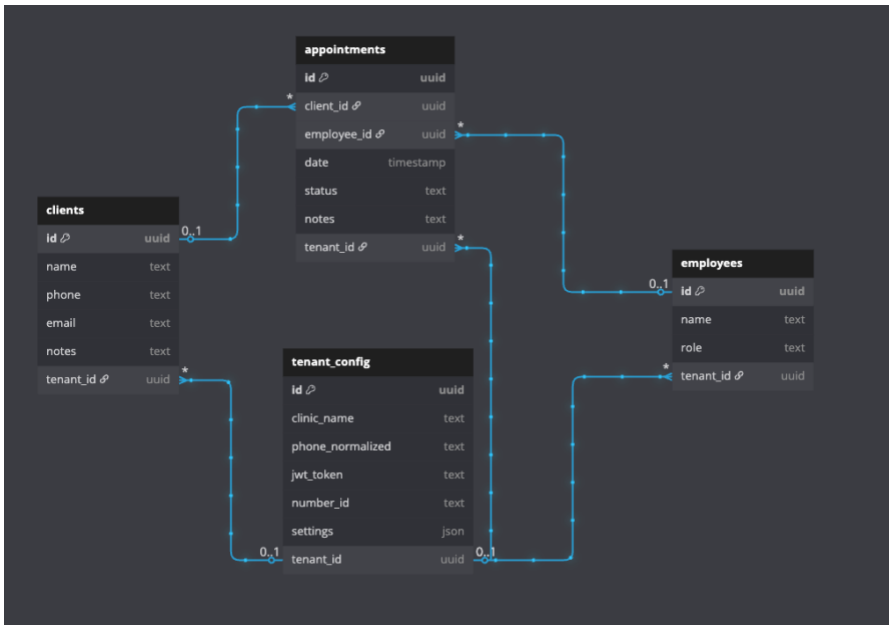

Figura 15. Diseño BBDD Diagrama Entidad-Relación

5.4. Fase de Implementación

Para explicar el proceso real seguido para construir y desplegar el sistema se utilizará este capítulo. Aquí se detallará el trabajo técnico: comandos utilizados, librerías instaladas, problemas encontrados y como se resolvieron. No se trata de una guía exhaustiva, pero si de una exposición fiel y concreta del desarrollo.

5.4.1. Implementación General del Sistema

El desarrollo comenzó con la creación de dos repositorios en Github, visitar para ver el código en: /ClinicAI-bot para el chatbot y /clinicai-web para el panel de administración. Ambos trabajan con TypeScript, un lenguaje de programación que tenía muchas ganas de probar y aprender. La adaptación y aprendizaje ha sido bastante sencilla por su similitud con JavaScript.

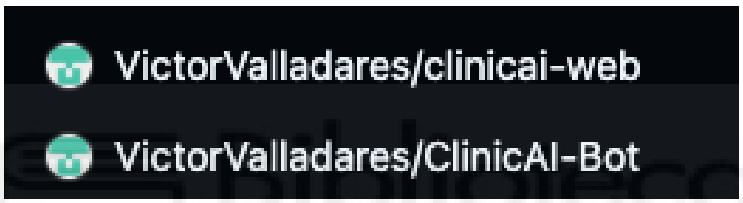

Figura 16. Proyectos en Github

5.4.2. Implementación Chatbot

En primer lugar, inicié el chatbot con la librería de **BuilderBot [36]** que facilitaba la integración con Meta y las respuestas del chatbot. Para iniciar el proyecto con esta librería debemos utilizar el siguiente comando en la terminal con node.js instalado.

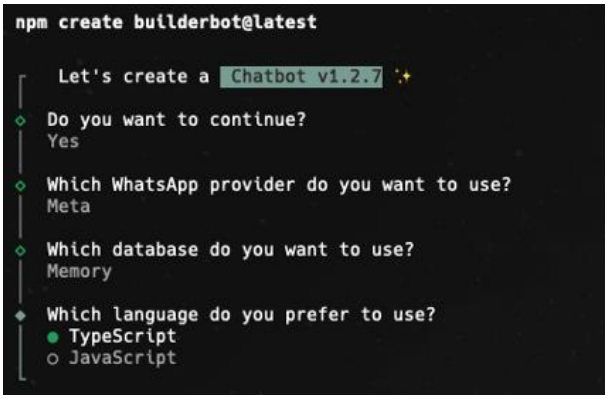

Figura 17. Creación Proyecto Chatbot

Aceptamos continuar, seleccionamos Meta como proveedor para WhatsApp, como base de datos la propia memoria del dispositivo y como lenguaje de programación TypeScript.

Este proceso inicial genera una estructura base con carpetas para los flujos, middlewares y configuración del proveedor. A partir de ahí comencé la integración a Meta, el desarrollo de los flujos, la conexión a Supabase y toda la lógica para las citas.

5.4.2.1. Integración con Meta

Uno de los pilares del sistema es su conexión directa con la Cloud API oficial de WhatsApp, que permite enviar mensajes, recibir respuestas y automatizar flujos sin necesidad de pasar por terceros. Todo se gestiona mediante peticiones HTTP autenticadas con tokens, y la seguridad está garantizada por una firma incluida en los encabezados.

Antes de que Meta acepte un webhook, es necesario superar el proceso de verificación: Meta realiza una petición GET al endpoint del servidor con varios parámetros (*hub.mode, hub.challenge* y *hub.verify_token*) y espera a que el servidor devuelva exactamente el valor del *hub.challenge.*

Al principio Meta me denegaba constantemente el webhook, así que utilice **Postman** una herramienta que permite simular peticiones al backend.

Como se puede ver en la figura 18, configuré una petición GET al endpoint /webhook con los siguientes parámetros:

- **hub.mode=**suscribe
- **hub.challenge=**123456789
- **hub.verify_token=**testWebhook

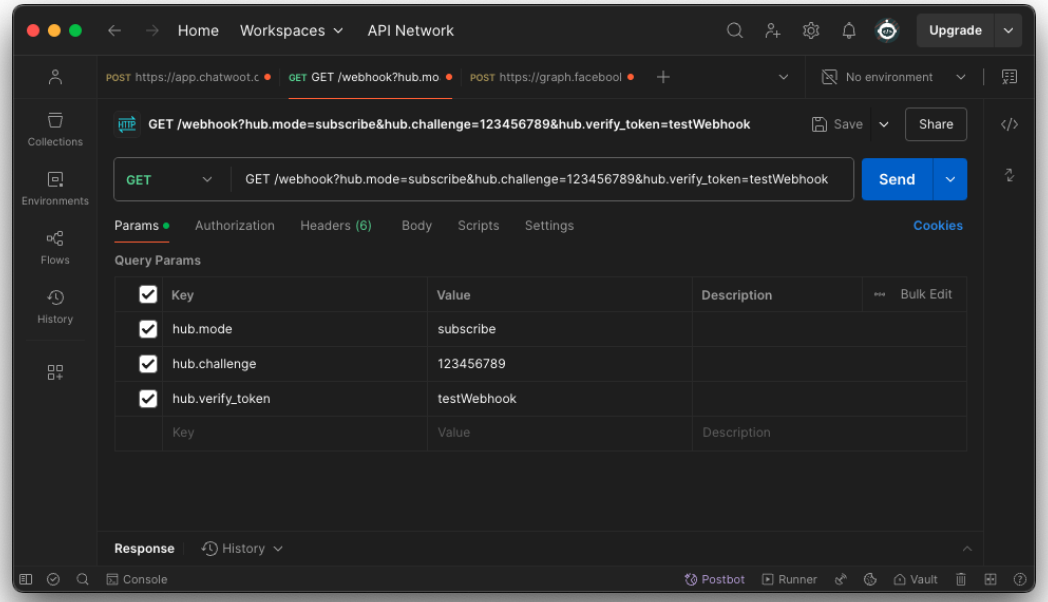

Figura 18. Petición HTTP Postman

Con esto averigüé que el problema estaba en mi backend, ya que no estaba gestionando correctamente la petición y devolvía **un código de error HTTP 500**, que indica un **fallo interno del servidor**. Tras probar varias soluciones, finalmente conseguí corregir el código y obtener una respuesta con **código HTTP 200 OK**, que confirma que la petición ha sido procesada con éxito. Al testarlo desde la consola de Meta, el webhook fue aceptado correctamente y quedó operativo

5.4.2.2.Flujos del chatbot

La librería de ***@builderbot/bot*** ha sido muy útil y clave en el desarrollo de la lógica del chatbot, ya que está pensada específicamente para crear flujos de conversación de forma modular y escalable.

BuilderBot se organiza en 3 componentes principales:

- **Flow (flujo)**: define la lógica conversacional. Cada flujo reacciona a ciertas palabras clave o eventos y responde en consecuencia.
- **Provider**: es el puente entre el bot y el canal de mensajería (en este caso, WhatsApp a través de Meta).

- **Database**: permite guardar el contexto y el estado de la conversación.

Cuando el flujo reacciona a alguna palabra, ocurre de esta manera.

```
addKeyword(['hola', 'cita', 'agendar'])
  .addAnswer('¡Hola! ¿Qué cita te gustaría reservar?')
```

Usando una sintaxis declarativa muy intuitiva, también permite capturar entradas del usuario, usar botones o detectar eventos (como recepción de audio, imagen o ubicación). Todo esto se maneja a través de funciones asíncronas que ofrecen control total sobre el flujo de la conversación. toda la lógica conversacional dentro de la carpeta /src/templates, siguiendo un enfoque modular.

Cada flujo corresponde a una fase específica del proceso de reserva o a una funcionalidad concreta, y está encapsulado en su propio archivo para facilitar el mantenimiento.

Los flujos principales que componen el sistema son:

- **mainFlow.ts** → flujo principal que actúa como entrada general del bot; redirige según el estado del cliente o su mensaje inicial.
- **registerFlow.ts** → gestiona el registro de nuevos usuarios solicitando nombre completo y correo.
- **intentionFlow.ts** → clasifica el mensaje recibido (por ejemplo: si el usuario quiere reservar una cita o está haciendo una consulta).
- **citaFlow.ts** → inicia el proceso de reserva de cita solicitando fecha y hora.
- **employeeFlow.ts** → muestra los empleados disponibles en función de la fecha elegida.
- **confirmationFlow.ts** → confirma la reserva y guarda la cita en Supabase.

- **reminderResponseFlow.ts.** → gestiona las respuestas a los recordatorios enviados el día anterior (confirmar, cancelar, etc.).
- **faqFlow.ts** → responde a preguntas genéricas del usuario (saludos, dudas, agradecimientos…).

Cada uno de estos flujos utiliza la función addKeyword para definir los disparadores, y addAnswer o addAction para gestionar las respuestas. Además, de capture para guardar entradas directas del usuario (por ejemplo, una fecha o nombre), y de gotoFlow para saltar entre flujos cuando es necesario continuar una conversación o derivar a otro bloque.

Gracias a esta estructura modular, el código se mantiene limpio y escalable. Si en el futuro quiero añadir, por ejemplo, un flujo de cancelación de cita o de reprogramación, simplemente creo un nuevo archivo y lo conecto desde el flujo principal.

5.4.2.2.1. Flujo de Registro

Antes de poder gestionar una cita o preguntar alguna duda, el sistema debe asegurarse de que el cliente está identificado correctamente. Para ello, se ha implementado un flujo de registro automático desde WhatsApp, sin necesidad de descargar ninguna app ni acceder a formularios externos.

Este flujo se activa de forma automática cuando el sistema detecta un mensaje entrante de un número que no está registrado en la base de datos de clientes del negocio al que está escribiendo.

El proceso es el siguiente:

1. El cliente envía un primer mensaje a través de WhatsApp
2. El backend identifica que el número no está asociado a ningún cliente correspondiente a ese negocio (tenant).
3. El sistema responde con un mensaje de bienvenida personalizado, utilizando los datos de configuración del negocio.
4. A continuación, se lanza una pregunta interactiva con botones para confirmar si desea iniciar el proceso de registro.

5. Si el cliente acepta, el bot solicita primero su nombre completo, y a continuación su correo electrónico.
6. El correo introducido se valida mediante una expresión regular (regex) para comprobar que sigue el formato estándar de una dirección valida.
7. Una vez se recogen y validan los datos, se almacena automáticamente un nuevo cliente en la tabla clients de Supabase, junto con el número de teléfono del cliente que se obtiene en el webhook.

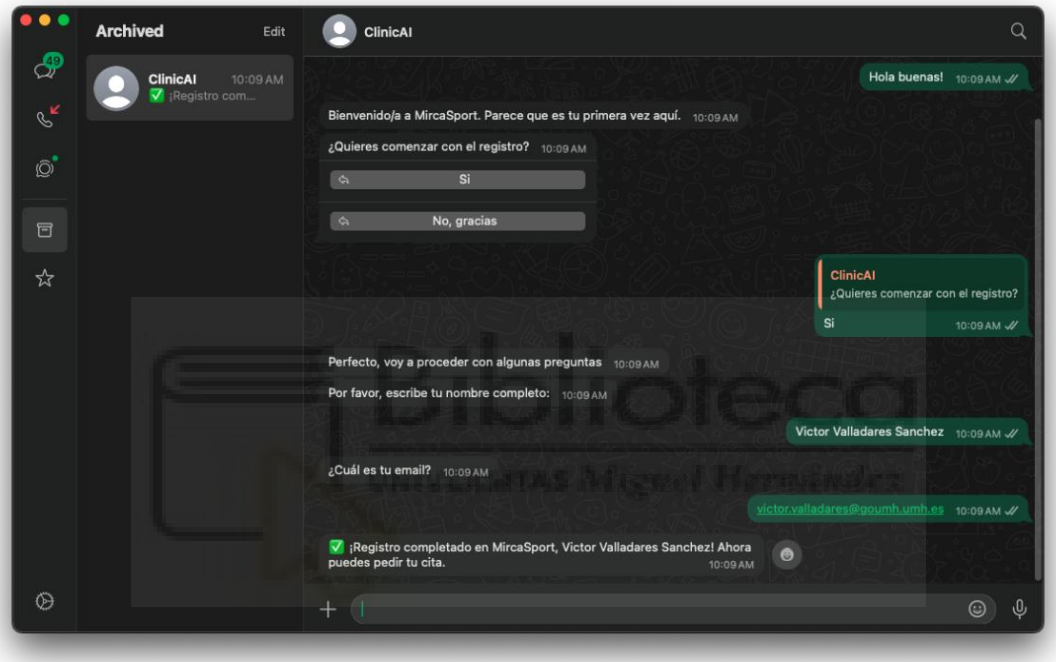

Figura 19. Ejemplo Flujo Registro

A partir de ese momento, cada vez que el cliente escriba al negocio, el sistema lo reconocerá y podrá interactuar con él con total normalidad.

5.4.2.2.2. Flujo de Cita

Una vez que el cliente está registrado en el sistema, puede comenzar a interactuar con el chatbot de forma natural. Uno de los flujos más importantes es la reserva de citas, que permite al usuario concertar una sesión con su centro habitual sin necesidad de llamar ni esperar.

Este flujo se activa cuando el sistema detecta que el mensaje del usuario tiene como intención principal pedir una cita, como ya se explicó en la capa de Inteligencia artificial.

Cuando se detecta que el usuario quiere reservar, el sistema entra en un flujo guiado paso a paso, manteniendo siempre un tono conversacional y cercano. A continuación, se detalla el proceso completo:

1. El cliente inicia la conversación
2. El sistema da la bienvenida utilizando el nombre del cliente y el nombre del negocio al que contacta.
3. Al detectar que la intención es reservar una cita, se pide la fecha y la hora deseada
4. El mensaje del cliente se interpreta usando procesamiento de lenguaje natural, extrayendo la fecha de forma flexible incluso si está escrita con expresiones como "pasado mañana a las 12".
5. El sistema consulta la base de datos para verificar la disponibilidad real del centro en ese día y hora. Si hay huecos responde confirmando y se muestra los profesionales disponibles en esa franja, para que el cliente pueda elegir.
6. Una vez el cliente elige al profesional, se solicita una confirmación final para asegurarse que todos los datos son correctos, si el cliente responde que si se registra la cita en la base de datos y se finaliza el proceso con un mensaje de confirmación.

Este proceso puede observarse a continuación, en la figura 20.

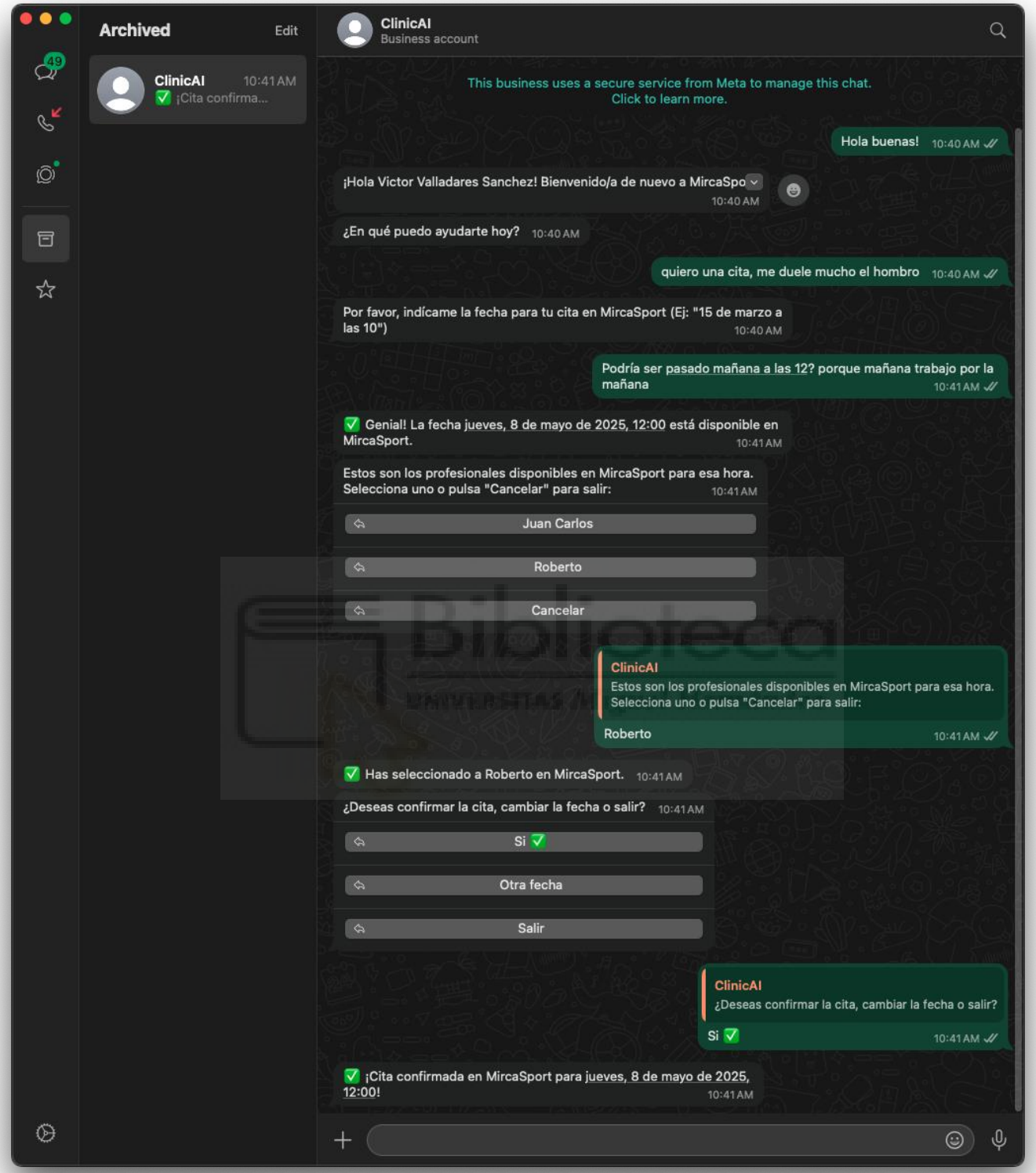

Figura 20. Ejemplo Flujo Cita

5.4.2.3. Envío de recordatorios mediante plantillas

Tras la reserva de la cita, con el objetivo de reducir las ausencias y mejorar la organización del negocio, el sistema incluye un mecanismo automático de recordatorio de citas. Esta funcionalidad no solo recuerda al cliente su próxima visita, sino que también permite confirmar o cancelar con un solo mensaje.

El proceso se ejecuta sin intervención humana todos los días a las 8 de la mañana:

1. El backend del chatbot, revisa todas las citas con estado *pending* programadas para el día siguiente.
2. Por cada cita encontrada, se genera un mensaje de recordatorio personalizado, mencionando al negocio, el nombre del cliente y la hora exacta de la cita.
3. El mensaje incluye dos botones para facilitar la interacción de confirmar o cancelar.
4. Según la respuesta del cliente, el sistema actualiza automáticamente el campo *status* de la cita.

Este flujo que se puede observar en la figura 21, mejora significativamente la eficiencia operativa del centro, ya que evita citas fallidas por olvido, elimina las confirmaciones manuales por parte del personal y permite al negocio anticipar cambios y optimizar el calendario diario.

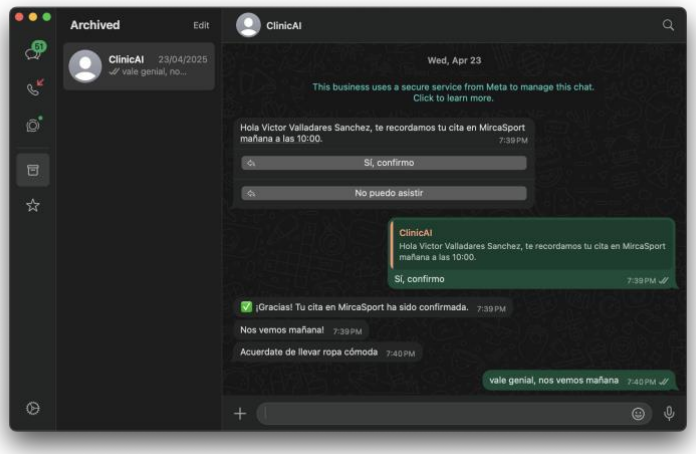

Figura 21. Ejemplo Recordatorio

Para automatizar el envío de recordatorios diarios, utilizo la librería de **node_cron**, que permite programar tareas en Node.js con una sintaxis muy sencilla. En mi caso, está configurada una tarea que se ejecuta todos los días a las 8 de la mañana.

```
cron.schedule('00 8 * * *', () => {
    console.log('⏰ Ejecutando tarea programada de recordatorios (8:00 servidor)...');
    checkAndSendReminders(provider, supabase);
  }, {
    scheduled: true,
    timezone: "Europe/Madrid"
  });
```

Esta función llama a la función checkAndSendReminders, la cual consulta en la base de datos todas las citas del día siguiente con estado pending y envía automáticamente una plantilla al cliente para confirmar o cancelar su cita.

Otro obstáculo que me enfrenté fue mientras desarrollaba los recordatorios, vi que el cliente por alguna razón no recibía los mensajes. Tras investigar por internet en diversos foros, descubrí que Meta identifica como 'spam' o mensaje molesto cuando inicias el mensaje a tu cliente pasadas 24h de la última conversación.

Y es que la API de WhatsApp funciona con dos tipos de mensajes bien diferenciados:

- **Mensajes de sesión** → Son mensajes cuando el cliente inicia la conversación y respondes dentro de la ventana de 24h. Estos mensajes son gratuitos y es la base del flujo del bot.
- **Mensajes de plantilla** → Cuando pasas esta ventana y quieres comunicarte con tus clientes no puedes mandar cualquier mensaje. Este mensaje debe estar aprobado por Meta mediante una plantilla y son cobrados por Meta.

En el caso de los recordatorios, se hace una llamada POST a la API rellenando dinámicamente los campos definidos en la plantilla.

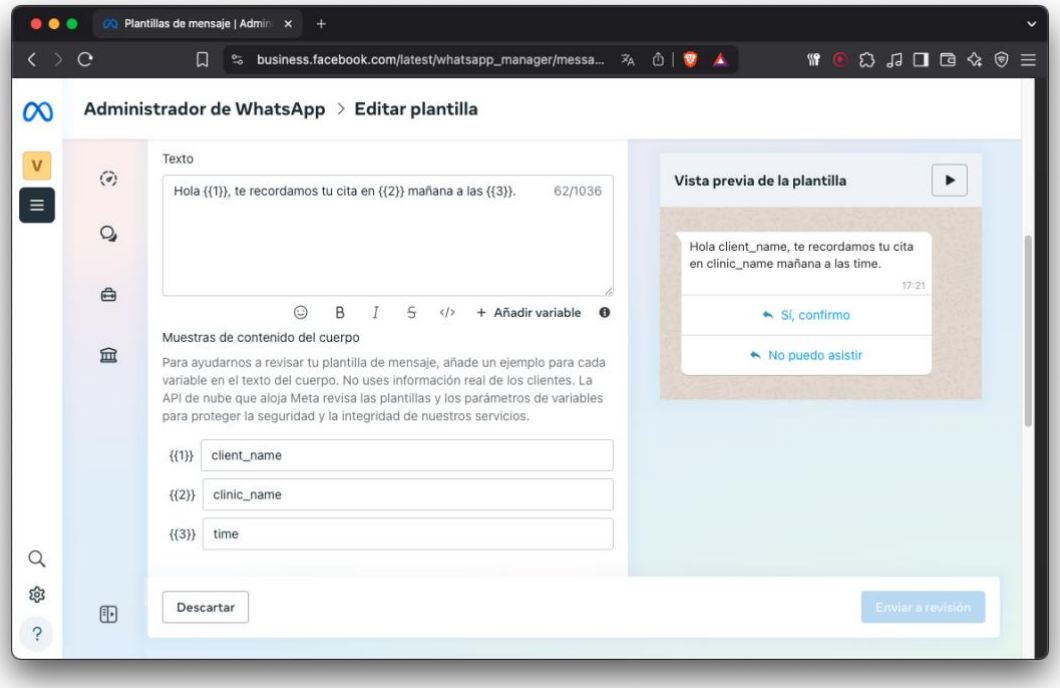

Figura 22. Plantilla Mensaje Recordatorio

Hola {{1}}, te recordamos tu cita en {{2}} mañana a las {{3}}.

Mediante estos números que se asignan a las variables (client_name, clinic_name y time), una vez creada la plantilla debemos enviarla a revisión para que sea aprobada por Meta. Según si es un uso funcional o una campaña publicitaria se cobra una tarifa u otra.

En definitiva, este apartado no solo demuestra que el sistema es capaz de comunicarse correctamente con la API oficial de WhatsApp, sino que también valida que cumple con las políticas de Meta, que responde correctamente a las verificaciones y que está preparado para escalar a múltiples clínicas de forma controlada y profesional. La conexión con WhatsApp es la base sobre la que se apoya todo el flujo automatizado del bot, y su correcto funcionamiento era imprescindible para dar sentido a todo el proyecto.

5.4.2.4. Gestión Manual con Chatwoot

Aunque el sistema está diseñado para funcionar de forma completamente autónoma en la mayoría de los casos, también se ha contemplado una vía de intervención humana como respaldo. Esta capa permite elevar la conversación del bot a un empleado cuando sea necesario, pausando automáticamente las respuestas automáticas para evitar duplicidad o confusión.

Para ello se ha integrado Chatwoot, una plataforma open source de atención al cliente que permite gestionar conversaciones desde un panel centralizado. El backend está preparado para pausar o reanudar el bot mediante comandos personalizados desarrollados para este propósito.

Estos comandos se envían de forma privada dentro del chat especifico:

- **/bot off** → Desactiva temporalmente las respuestas del bot.
- **/bot on** → Vuelve a activarlo.
- **/bot status** → Informa del estado actual del bot (activo o pausado).

En la figura 23 se puede ver como el bot responde y es pausado, para poner un mensaje personalizado a ese cliente. En la siguiente imagen, la figura 24, se puede ver lo que ocurre en el backend.

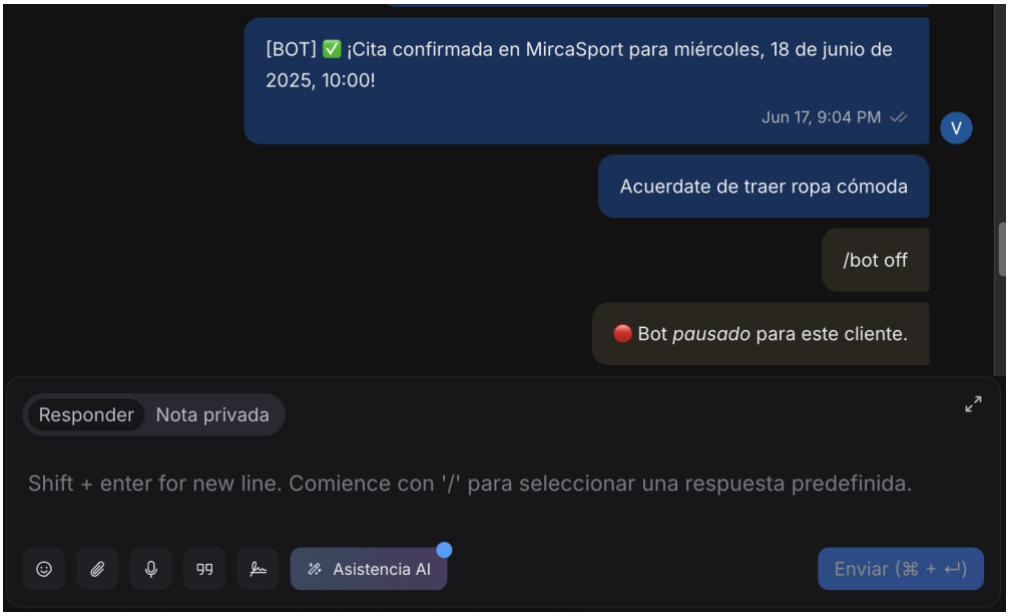

Figura 23. Chat de Chatwoot

Figura 24. Teléfono Añadido a la Blacklist

Gracias a esta integración y al sistema de comandos implementado, cuando se desactiva el bot, el número del cliente pasa automáticamente a una **lista de exclusión temporal** (blacklist). Esto garantiza que el chatbot no envíe ninguna respuesta durante ese tiempo, permitiendo que el negocio tome el control de la conversación sin interferencias. De este modo, se mantiene una experiencia fluida, profesional y completamente controlada, adaptándose a las necesidades reales del día a día.

5.4.3. Implementación módulo integración LLM

Una de las partes más relevantes del sistema es su integración con modelos de lenguaje natural (LLMs), ya que son los encargados de interpretar lo que dice el cliente, clasificar su intención y, en algunos casos, generar respuestas útiles y naturales.

Para ello, se ha creado un módulo específico en el archivo aiServices.ts dentro de la carpeta /services, que encapsula toda la lógica necesaria para conectarse con proveedores de LLM. Este diseño cumple tres objetivos clave: mantener el sistema desacoplado del proveedor, facilitar futuras migraciones o pruebas, y centralizar la lógica en un único punto del proyecto.

El módulo expone una única función: getAIResponse(). Esta función actúa como punto de entrada y decide a qué proveedor enviar la solicitud en función de una variable de entorno llamada AI_PROVIDER_URL. Gracias a esto, el sistema puede cambiar de modelo (por ejemplo, de OpenAI a DeepSeek) sin necesidad de modificar el código fuente, simplemente ajustando las variables del entorno en Railway.

Ambos proveedores utilizan el mismo endpoint base (POST /v1/chat/completions) y comparten un esquema de payload muy similar. Esto permite enviar el mismo tipo de estructura de prompt y mensajes a cualquiera

de los dos, simplificando mucho la integración como se puede ver en la figura 25.

```
// AI Provider Configuration
AI_PROVIDER: process.env.AI_PROVIDER ?? "openai", // "openai" or "deepseek"
ENABLE_FALLBACK: process.env.ENABLE_FALLBACK === "true", // Enable fallback to DeepSeek when OpenAI fails

// DeepSeek Configuration
DEEPSEEK_API_KEY: process.env.DEEPSEEK_API_KEY,
DEEPSEEK_MODEL: process.env.DEEPSEEK_MODEL ?? "deepseek-chat", // cheapest model
DEEPSEEK_BASE_URL: process.env.DEEPSEEK_BASE_URL ?? "https://api.deepseek.com",
```

Figura 25. Configuración Proveedores LLM

Este enfoque modular no solo mejora el mantenimiento y la claridad del código, sino que permite realizar pruebas con distintos modelos (por calidad, coste o rendimiento) sin modificar el resto de la lógica del sistema. A largo plazo, esto permite que el sistema evolucione de forma flexible, adaptándose a nuevos avances en el sector de la inteligencia artificial sin necesidad de una reescritura completa.

5.4.4. Implementación de la interfaz web

Para crear el proyecto de la parte web utilice Next.js con este comando seleccionado TS, ESLint, Tailwind CSS y App Router.

```
npx create-next-app@latest
Need to install the following packages:
create-next-app@15.3.2
Ok to proceed? (y) y

✔ What is your project named? … clinicai-web
✔ Would you like to use TypeScript? … No / Yes
✔ Would you like to use ESLint? … No / Yes
✔ Would you like to use Tailwind CSS? … No / Yes
✔ Would you like your code inside a `src/` directory? … No / Yes
? Would you like to use App Router? (recommended) › No / Yes
```

Figura 26. Creación Proyecto Web Next.js

Todas estas tecnologías me permitían construir una interfaz moderna, tipada y bien estructurada. Sobre esta base posteriormente integré:

- **Shadcn/ui** para los componentes de interfaz.

- **tRPC** para la comunicación con el backend con tipado de extremo a extremo.
- **Prisma** para interactuar con Supabase desde el servidor

A la hora de seleccionar las tecnologías el desarrollo rápido era parte importante en la decisión, así como dar una apariencia moderna y una buena experiencia de usuario.

La librería **shadcn/ui [22]**, proporciona un conjunto de componentes accesibles y personalizables, construidos sobre Radix UI y estilizados con Tailwind CSS. Su propósito es ofrecer una colección de elementos comunes de interfaz (botones, formularios, tablas, diálogos, etc.) que mantengan una estética moderna y profesional sin necesidad de diseñarlos desde cero.

Frente a otras librerías populares como Material UI, Chakra UI o Ant Design, shadcn/ui destaca por su enfoque minimalista y flexible: en lugar de imponer un estilo visual cerrado, permite al desarrollador adaptar cada componente fácilmente al diseño del proyecto. Además, al estar construido sobre Tailwind, se integra perfectamente con la estructura y lógica del stack tecnológico que ya estaba usando en el panel web.

La decisión de utilizar esta librería responde tanto a criterios técnicos como de productividad. Me ha permitido mantener una apariencia visual coherente y moderna en toda la aplicación, mientras me centraba en lo realmente importante: la funcionalidad. También valoro positivamente que el código fuente de cada componente se copie en el proyecto y pueda modificarse libremente si se necesita un comportamiento personalizado.

Los componentes se organizan en carpetas dentro de */components* y siguen un patrón modular.

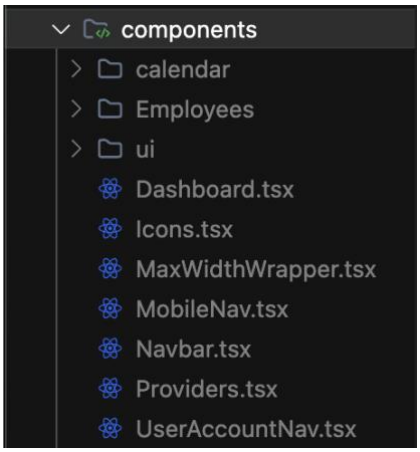

Figura 27. Componentes del Proyecto Web

En esta carpeta se guardan los componentes para las páginas, para la vista móvil, los elementos del calendario, tablas de empleados…

5.4.4.1. Implementación Página de Inicio

El acceso a la plataforma se realiza mediante una interfaz sencilla y segura. A continuación, se describen las diferentes pantallas que forman parte de la experiencia inicial del usuario en la página principal de ClinicAI:

Figura 28. Página Principal

La primera pantalla que ve el usuario no autenticado es una landing page. Se presenta un mensaje de bienvenida con una propuesta de valor clara y directa:

"Simplifica la gestión de tu clínica de fisioterapia". El botón "Empezar" actúa como llamada a la acción y redirige al proceso de login. Esta pantalla busca generar confianza mediante un diseño minimalista y profesional, inspirado en prácticas actuales de diseño web.

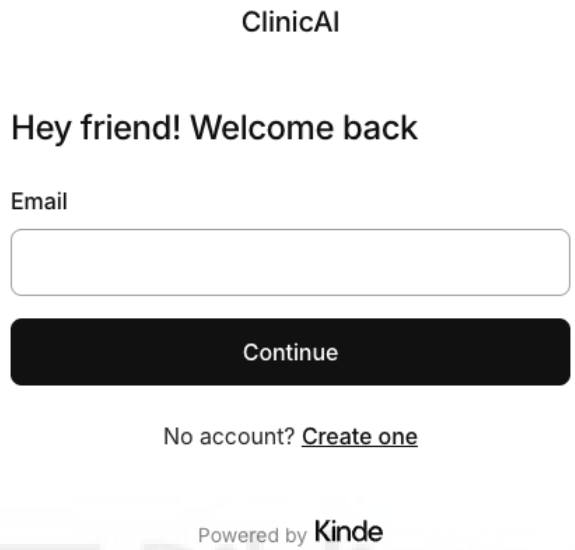

Figura 29. Inicio de Sesión

Una vez el usuario hace clic en "Empezar", es redirigido al sistema de autenticación ofrecido por Kinde Auth. La primera pantalla de este flujo solicita el correo electrónico del usuario. No requiere contraseña, ya que la validación se realiza por medio de un código temporal (código OTP), lo que simplifica el acceso y mejora la seguridad sin sacrificar usabilidad.

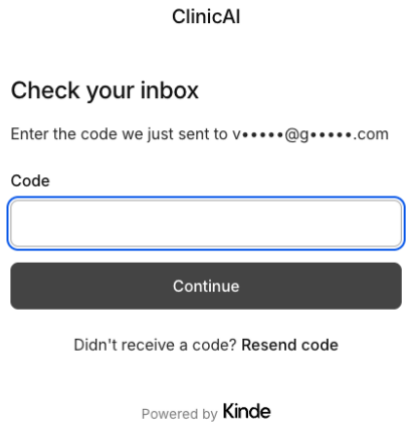

Figura 30. Código Temporal de Acceso

Tras introducir su email, el usuario recibe un código de verificación de un solo uso. En esta pantalla debe introducir dicho código para confirmar su identidad. El sistema valida este código automáticamente y, si es correcto, concede acceso al panel de administración.

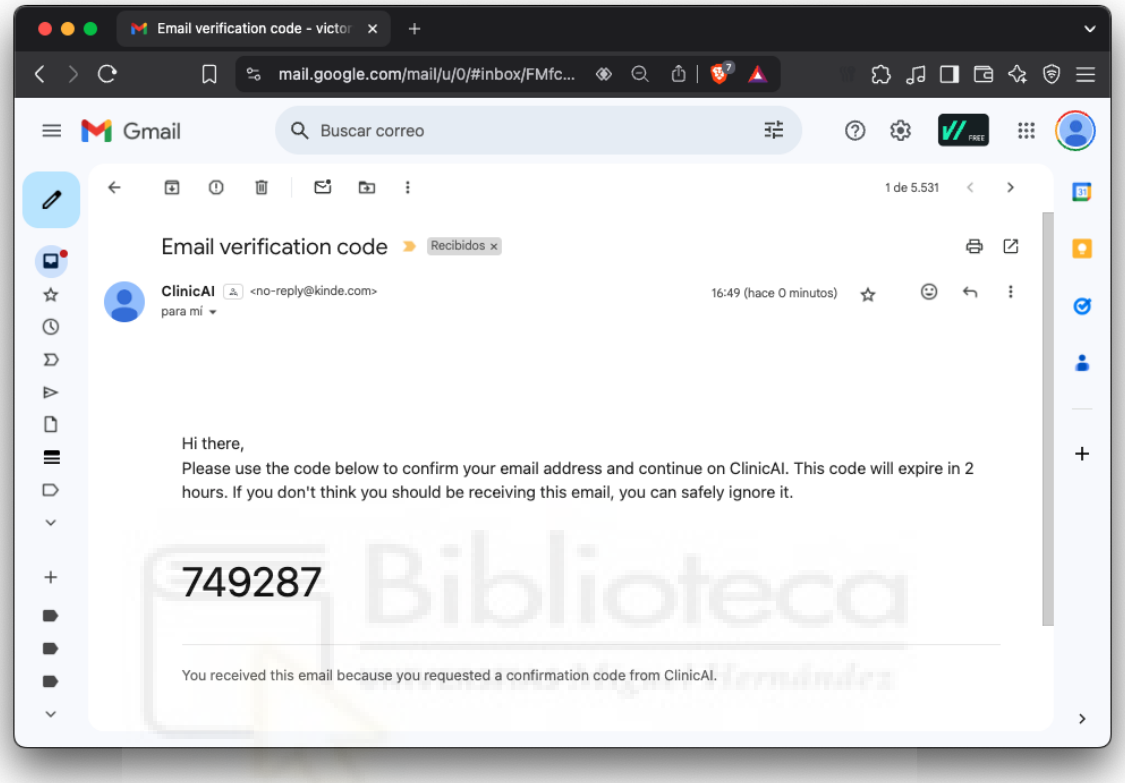

Figura 31. Código Verificación Email

El correo enviado por Kinde incluye el código necesario para completar la autenticación. Este mensaje llega en pocos segundos y tiene un formato simple y claro, lo que evita confusiones o errores por parte del usuario. Es importante destacar que el código tiene un tiempo de expiración limitado por seguridad.

5.4.4.2. Implementación Calendario de Citas

La pantalla de calendario es el eje central del panel web, desde donde el negocio puede visualizar, organizar y gestionar todas sus citas de manera clara y eficiente.

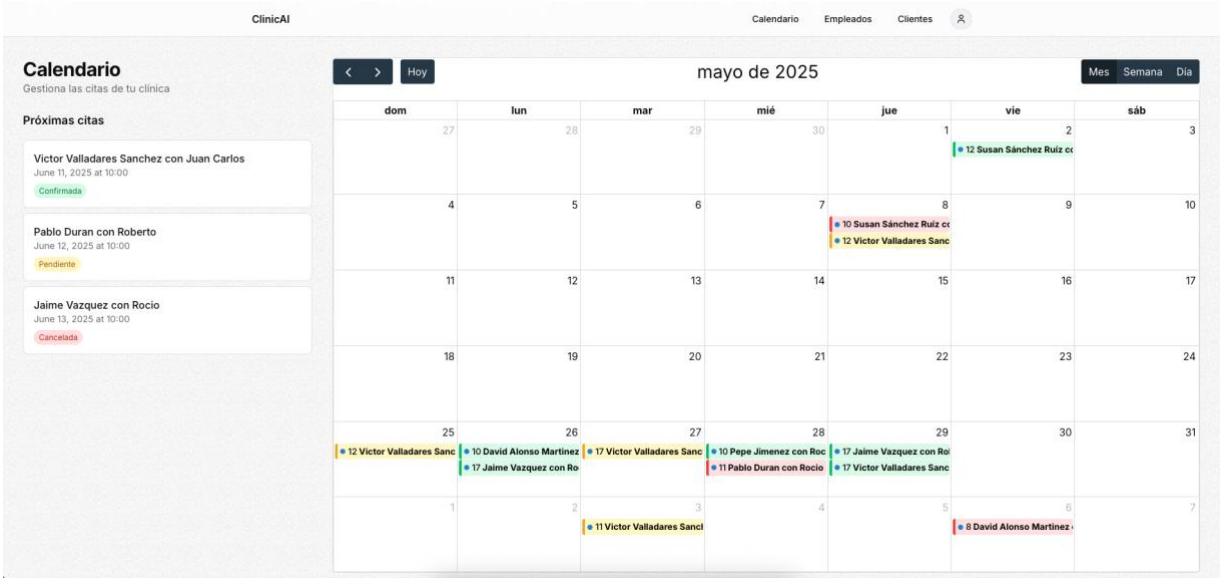

Figura 32. Página Calendario Citas

El calendario presenta una interfaz mensual por defecto, aunque es posible cambiar a vista semanal o diaria con un solo clic. Cada cita aparece representada como un bloque de color, diferenciando al profesional asignado para facilitar la lectura rápida. En la parte izquierda se muestra una lista con las próximas citas destacadas, ordenadas cronológicamente.

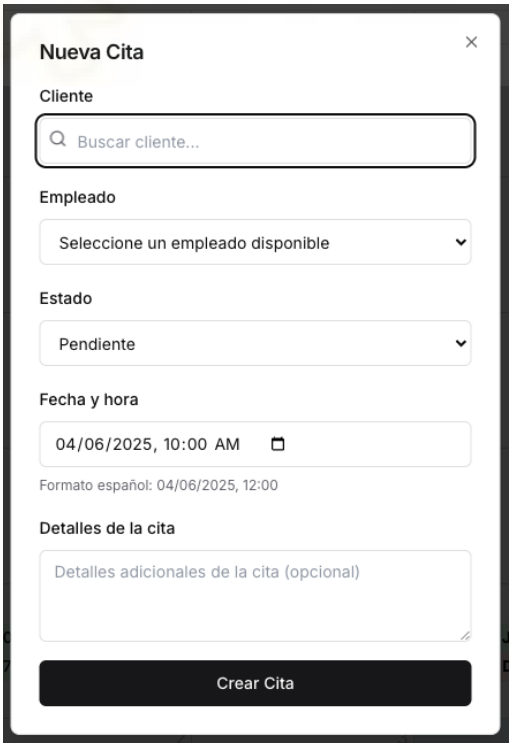

Figura 33. Añadir o Editar Cita

Al hacer clic sobre cualquier espacio vacío en el calendario, se abre el modal que se puede ver en la imagen que permite añadir una nueva cita.

Una vez completado el formulario, al hacer clic en "Crear Cita" se guarda en Supabase y aparece automáticamente en el calendario. Este flujo permite al personal registrar citas de última hora o gestionar cambios sin necesidad de pasar por WhatsApp.

5.4.4.3. Implementación Gestión de Empleados

Esta sección permite al negocio mantener actualizada la información sobre su equipo, visualizar los horarios y realizar cambios de manera sencilla e intuitiva.

En la Figura 50 podemos observar la pantalla principal de gestión de empleados. A la izquierda se muestra un buscador que permite filtrar por nombre. En el cuerpo principal de la tabla se listan todos los empleados registrados junto a su rol profesional y el horario configurado. En la columna de acciones se ofrecen accesos rápidos para editar o eliminar al empleado.

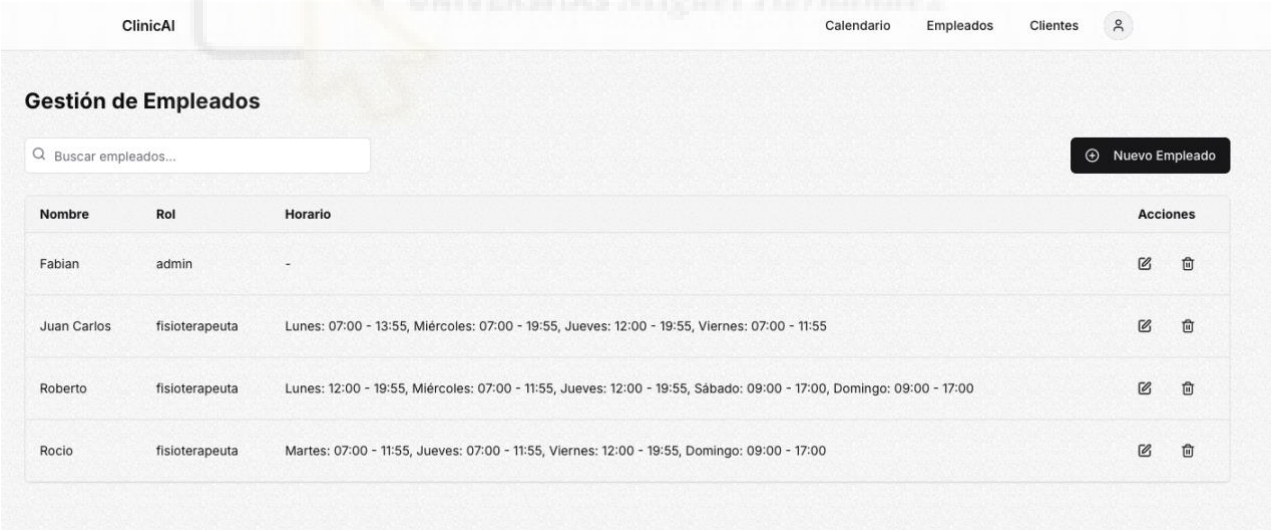

Figura 34. Gestión de empleados

Al hacer clic sobre el botón de edición o "Nuevo Empleado", se abre una ventana emergente (modal) como se muestra en la siguiente imagen.

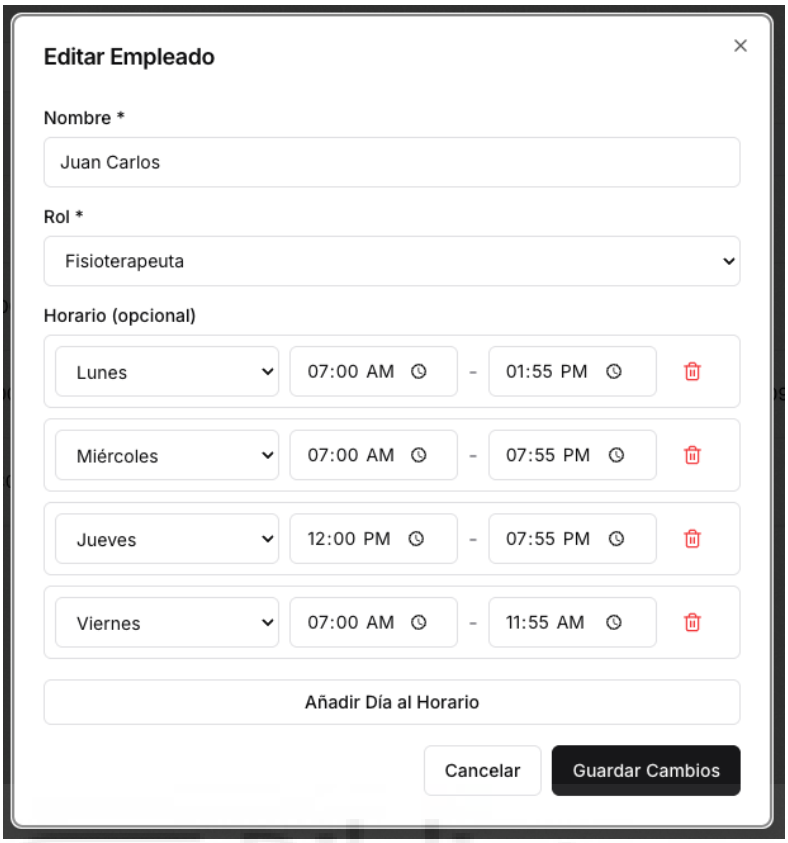

Figura 35. Añadir o Editar Empleado

5.4.4.4. Implementación Gestión de Clientes

La sección de clientes está diseñada para ofrecer al personal del negocio una visión clara y ordenada de todos los usuarios registrados. Aunque el registro de clientes se realiza automáticamente cuando escriben por WhatsApp, esta interfaz permite modificar sus datos manualmente, añadir nuevos registros o consultar la información disponible de forma cómoda y eficiente.

En la siguiente imagen, se muestra la pantalla principal del módulo. La tabla recoge todos los campos esenciales como el nombre, correo electrónico y número de teléfono. A la derecha se encuentran las acciones disponibles: editar o eliminar.

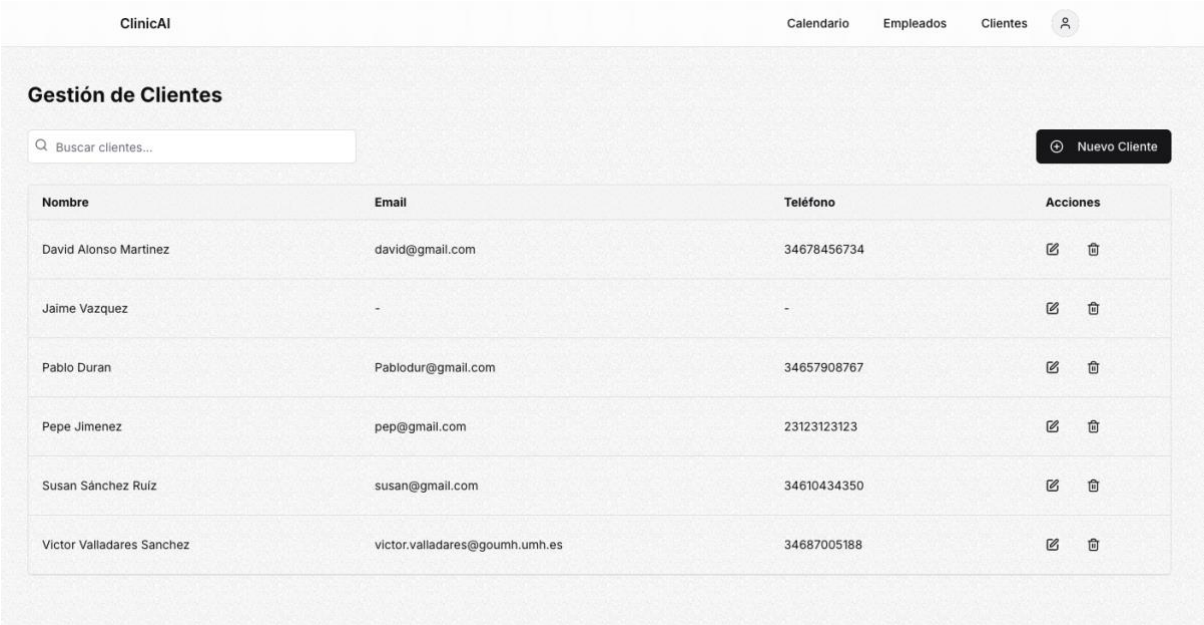

Figura 36. Gestión de Clientes

Al hacer clic en el botón "Nuevo Cliente" o la acción de editar, se despliega una ventana modal como la que se observa en la siguiente imagen. Desde ahí, el administrador puede introducir el nombre, correo electrónico, teléfono y anotaciones internas relevantes. Esto resulta útil, por ejemplo, si el cliente llama por teléfono o quiere ser registrado directamente desde recepción.

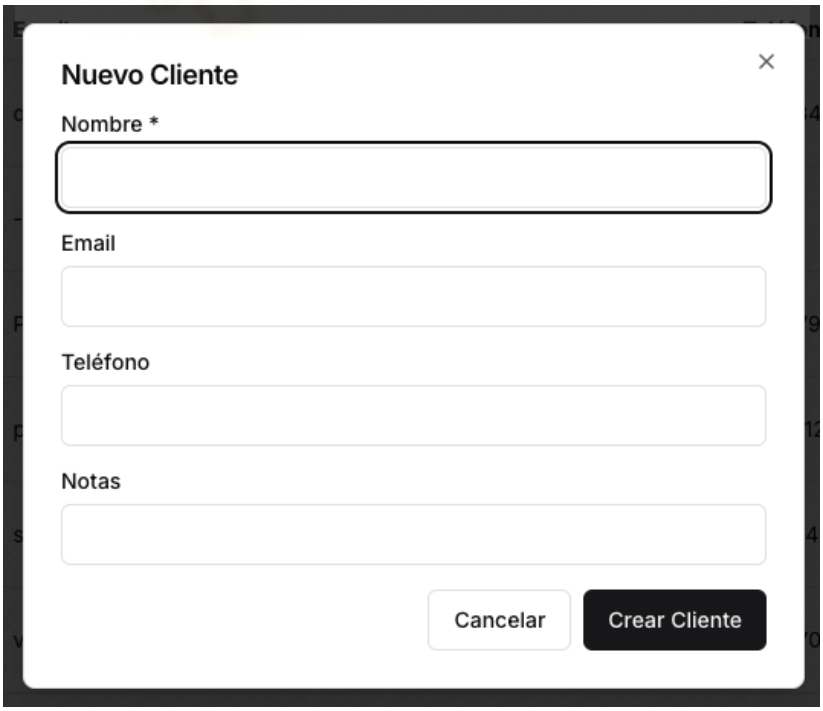

Figura 37. Añadir o Editar Cliente

5.4.5. Implementación del backend

Toda la información que consumen el chatbot y el panel de administración web reside en **Supabase**, un servicio open-source que gestiona PostgreSQL y añade ciertas ventajas para el desarrollo.

En primer lugar, al crear una tabla Supabase genera de forma inmediata una **API REST y un cliente TypeScript**. Por lo que desde la web o una app futura se puede consultar o modificar datos sin escribir controladores extra.

Además, ofrece **Row-Level Security (RLS)** que permiten que cada negocio vea exclusivamente sus citas y clientes, sin duplicar base de datos ni añadir lógica de filtrado en la aplicación. Sin embargo, el principal argumento para elegir Supabase sobre alternativas como Firebase es su capa gratuita que permite desarrollar el MVP perfectamente y se puede escalar según el uso sin migraciones complejas.

Tabla	Propósito	Campos principales	Relaciones clave
clients	Datos de cada cliente	id (UUID, PK), name, phone, email, notes, tenant_id.	1 a N con appointments
employees	Profesionales de cada negocio	id (UUID, PK), name, role, tenant_id.	1 a N con appointments
appointments	Citas agendadas	id, client_id, employee_id, date, status, tenant_id.	FK a clients y employees
tenant_config	Preferencias y credenciales del negocio	id, clinic_name, phone_normalized, jwt_token, number_id, settings, tenant_id.	1 a 1 con cada negocio

Tabla 20. Tablas Base de Datos

En el núcleo se encuentra *appointments*, que enlaza un cliente y un empleado mediante claves foráneas y registra la fecha, el estado y posibles notas internas. Cada entidad incluye un campo especial llamado tenant_id, que identifica a qué negocio pertenece cada dato.

Este campo permite que el sistema funcione como **multitenencia**, es decir, que varios negocios (por ejemplo, diferentes clínicas) compartan la misma base de

datos sin acceder nunca a la información de otros. Cada negocio es un tenant, y gracias a las políticas de seguridad de Supabase (**Row-Level Security**, o RLS), todas las consultas se filtran automáticamente: cuando un negocio accede al sistema, solo puede ver sus propios datos. Esto se consigue sin necesidad de añadir condiciones manuales en cada consulta, lo que mejora la seguridad y simplifica el desarrollo.

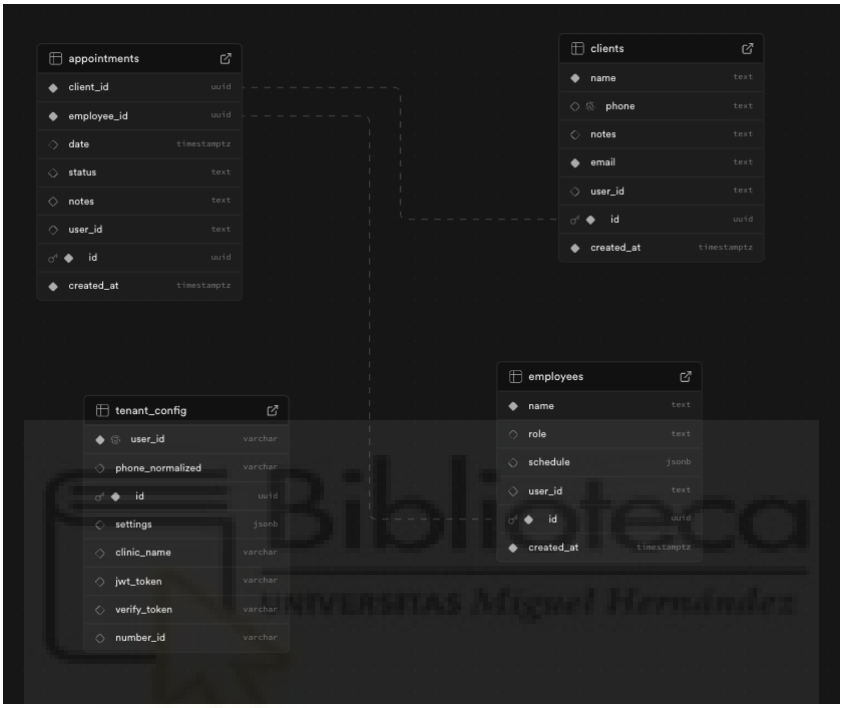

Figura 38. Tablas Supabase

5.5. Fase de Pruebas

Aunque el sistema ha sido probado funcionalmente en todas sus partes durante el desarrollo, la validación más relevante recae en el componente de inteligencia artificial. Al tratarse de un modelo generativo, no es posible verificar su comportamiento mediante pruebas clásicas automatizadas. Por eso, este capítulo se centra en analizar la calidad de las respuestas del asistente conversacional y cómo se ha medido su efectividad de forma cuantitativa.

5.5.1. Pruebas Funcionales

Durante el desarrollo del sistema, se llevaron a cabo pruebas funcionales manuales para validar que cada una de las funcionalidades básicas del sistema se comportaba según lo esperado. Este tipo de pruebas se realizó tanto en

entorno de desarrollo como en pruebas reales simuladas, especialmente en la parte del chatbot.

Para la capa conversacional, se utilizó la herramienta ngrok, que permitió exponer localmente el puerto del servidor para recibir mensajes reales desde WhatsApp Cloud API, sin necesidad de desplegar continuamente en producción. Esta estrategia facilitó iterar con rapidez, detectar errores en tiempo real y refinar los flujos conversacionales en condiciones similares a las de uso real.

Se validaron los siguientes **flujos del chatbot**:

- Registro automático de nuevos clientes tras recibir el primer mensaje.
- Reserva de citas con validación de fechas y disponibilidad.
- Confirmación y cancelación de citas mediante recordatorio.
- Detección de intenciones erróneas y redirección a contacto humano tras varios fallos.
- Flujo de preguntas frecuentes (FAQ) con respuestas generadas mediante IA.

En el **panel web**, también se realizaron pruebas funcionales desde la interfaz gráfica para comprobar:

- Acceso seguro al panel mediante autenticación con Kinde.
- Visualización y navegación en el calendario de citas.
- Creación, edición y eliminación de citas, empleados y clientes.
- Sincronización con la base de datos Supabase en tiempo real.

Estas validaciones se llevaron a cabo de forma iterativa durante el desarrollo de cada módulo. Cada funcionalidad se probaba tras su implementación, resolviendo posibles errores antes de continuar con la siguiente. Aunque no se han utilizado frameworks de testing automatizado, este enfoque manual ha resultado eficaz para asegurar la estabilidad del MVP.

5.5.2. Evaluación del rendimiento del modelo conversacional

Uno de los pilares del sistema es la capacidad de interpretar correctamente lo que dice el usuario en lenguaje natural y responder de forma precisa y coherente.

Para ello, se utilizan dos prompts distintos: uno para detectar la intención del mensaje y decidir si el usuario quiere reservar una cita o simplemente está haciendo una consulta (FAQ), y otro para generar la respuesta adecuada en caso de que se trate de una pregunta frecuente.

En el primer caso, la evaluación es binaria: el modelo acierta o falla al clasificar la intención. En cambio, en las respuestas tipo FAQ, lo importante no es solo si contesta, sino cómo de bien lo hace: si es fiel a la información, si entiende correctamente la pregunta y si transmite la respuesta de forma clara y útil.

El objetivo de esta validación es cuantificar el rendimiento del modelo integrado (GPT-4o-mini) con los prompts diseñados específicamente para este sistema. A partir de los resultados obtenidos, se analizan los errores y se toman decisiones de mejora fundamentadas en datos reales.

5.5.2.1. Elección de la herramienta de evaluación: Ragas

Para medir de forma objetiva la calidad de las respuestas generadas por el modelo, se ha utilizado **Ragas [37]**, un framework open-source pensado específicamente para evaluar sistemas conversacionales que integran modelos de lenguaje natural.

En este proyecto se utiliza un modelo generativo (GPT-4o-mini) que responde directamente a las preguntas del usuario basándose en el prompt que se le proporciona. Ragas permite analizar este tipo de respuestas mediante métricas automáticas que analizan aspectos clave como:

Nombre métrica	Significado
Answer_relevancy	Mide si la respuesta está relacionada con la pregunta planteada.
Answer_correctness	Evalúa si la respuesta es coherente con la información de referencia definida como correcta.

Tabla 21. Métricas Ragas

Estas métricas son especialmente útiles en sistemas donde no existe una única forma correcta de responder, pero sí es importante que las respuestas sean claras, útiles y fieles al contenido original.

Además de sus características, la elección de Ragas también responde a criterios personales. Mi experiencia con Python sumado al conocimiento de mi tutor quien me animó a utilizar esta herramienta ha facilitado la preparación del entorno y la codificación del script para la evaluación.

5.5.2.2. Preparación del entorno de evaluación

Para llevar a cabo toda la evaluación, primero es necesario preparar todo el entorno desde cero: definir el script, definir el dataset, estructurar los prompts y conectarlo todo a la API de OpenAI.

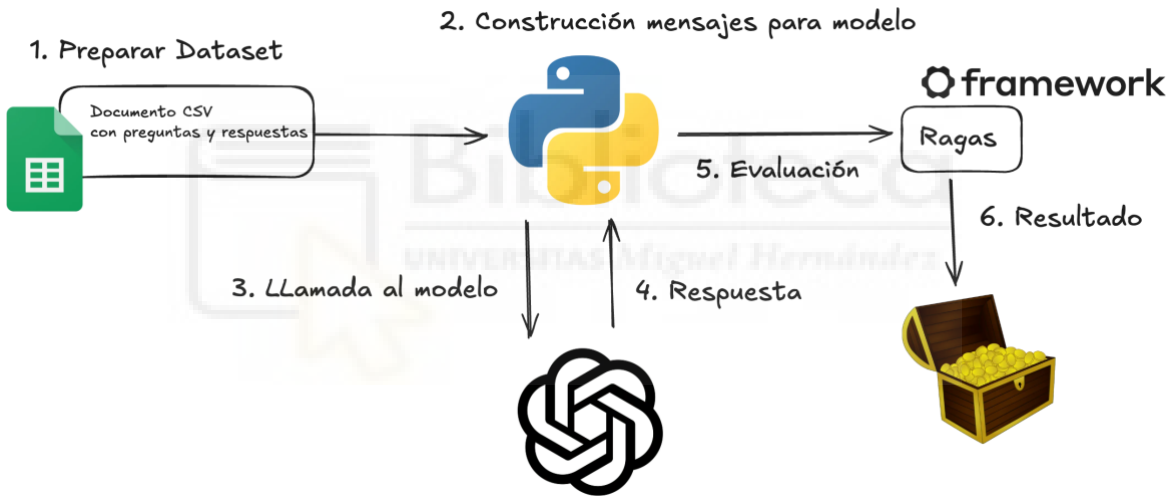

Figura 39. Diagrama explicativo Ragas

El script se desarrolló en Python, aprovechando las funciones de Ragas junto con la librería **LangChain [38]**, que facilita la conexión con modelos como GPT-4o-mini. La clave de API y el modelo se definieron mediante variables de entorno, y se configuró la llamada al modelo para que se comportara igual que en el sistema real, respetando el mismo prompt, temperatura y formato de entrada.

El prompt utilizado se almacenó en un archivo de texto externo en lugar de embebido directamente en el código. Esto se hizo para poder modificarlo fácilmente durante las pruebas sin tocar el código.

En cuanto al dataset, se generó dos archivos CSV, el primero con 100 preguntas y su clasificación para evaluar la detección de intención y el segundo con 30 preguntas frecuentes reales, adaptadas al contexto de una clínica como MircaSport. Cada pregunta iba asociada a su respuesta esperada, definida manualmente. El formato incluía dos columnas: question y ground_truth, siguiendo el estándar que requiere Ragas.

La evaluación se realizó en local. Para cada pregunta, el script construía una entrada que combinaba el prompt (como mensaje del sistema) y la pregunta (como mensaje del usuario), enviándola al modelo y capturando su respuesta. Ragas comparaba después esa respuesta con la ground truth utilizando las métricas mencionadas en el punto anterior.

Todo este proceso permitió reproducir el comportamiento del sistema en condiciones reales y obtener una evaluación cuantitativa del rendimiento del modelo en tareas conversacionales, con total control sobre el input y el output.

5.5.2.3. Iteración y mejora del rendimiento conversacional

5.5.2.3.1. Prompt de detección de intención

Al realizar una primera evaluación de la clasificación del bot ante las diferentes peticiones, en el primer prompt que aparece en el Anexo 4.A, se obtuvo una precisión estricta del 91,51%, una buena puntuación, pero no suficiente para la evaluación de intenciones.

Por ello, añadí reglas muy específicas al prompt, que se pueden ver en el Anexo 4.B para ayudar al modelo en su toma de decisiones. Con estas pequeñas mejoras ahora tenemos un porcentaje casi perfecto en las pruebas con el dataset de 100 peticiones.

```
Guardando resultados detallados en results_detection.csv...

Resumen de la evaluación (también guardado en evaluation_results.csv):
                                 user_input response reference  answer_relevancy  answer_correctness  strict_accuracy
0                          Hola, buenos días      FAQ       FAQ          0.699381                1.00                1
1     Me gustaría pedir cita para esta tarde     CITA      CITA          0.748959                0.25                1
2  ¿Qué tratamientos hacéis para cervicales?      FAQ       FAQ          0.671101                1.00                1
3                           ¿Tenéis parking?      FAQ       FAQ          0.689320                1.00                1
4     Quiero reservar una sesión con Roberto     CITA      CITA          0.705198                1.00                1

Precisión Estricta General (response == reference): 99.06%
Puntuación Media de Answer Correctness (Ragas): 0.8722
```

Figura 40. Resultado prompt detección Ragas

Un 99,06% de precisión estricta, es decir, rozando la perfección. Este incremento de efectividad de aproximadamente un 8% supondrá una notable diferencia a gran escala.

5.5.2.3.2. Prompt de respuesta FAQ

Para evaluar este prompt como hemos mencionado anteriormente, no es tan fácil de saber si la respuesta es correcta o no, por lo que la valoración de la respuesta es clave y cobrá mayor importancia la puntuación media de 'Answer_correctness' (Coherencia de la respuesta).

La primera evalución se realizó con un prompt, relativamente sencillo, disponible en el Anexo 4.C y obtuvo una puntuación de 0.67 en relevancia de las respuestas y 0,44 en exactitud, lo cual es bastante mejorable.

Este primer prompt, tenía varios problemas, se pasaba únicamente el nombre de la clínica desde la base de datos por lo que no es ni personalizable ni escalable, además al no tener unas reglas bien definidas se inventaba algunas respuestas que no conocía.

En la nueva versión del prompt, observable en el Anexo 4.D, es único para cada clínica y fácilmente personalizable, para mejorar la experiencia de usuario. Además, es guardado en la base de datos desde la web, en vez de en el repositorio como anteriormente. Este nuevo prompt ha obtenido unos resultados más coherentes y alineados con la realidad de 0,75 de relevancia y 0,77 de exactitud como se puede ver en la figura 41.

```
Evaluación de Ragas completada.
Guardando resultados detallados en results_faq.csv...

Resumen de la evaluación (también guardado en results_faq.csv):
Mostrando las primeras filas de las columnas: ['answer_relevancy', 'answer_correctness']
   answer_relevancy  answer_correctness
0          0.918555            1.000000
1          0.924409            1.000000
2          0.000000            0.545479
3          0.757945            0.748574
4          0.932691            0.992105
Puntuación Media de Answer Relevancy (Ragas): 0.7546
Puntuación Media de Answer Correctness (Ragas): 0.7734
```

Figura 41. Resultado prompt FAQ Ragas

Este bucle de aprendizaje me ha permitido iterar distintas soluciones hasta conseguir ajustes en el prompt que pueden tener un gran impacto en la calidad de las respuestas del sistema, y disponer de herramientas como Ragas facilita este tipo de mejoras de forma estructurada y medible.

5.5.3. Conclusión

La evaluación del sistema ha permitido medir de forma objetiva el rendimiento conversacional del modelo de lenguaje integrado, en concreto su capacidad para detectar la intención del usuario y responder adecuadamente a preguntas frecuentes. A través de la herramienta Ragas, se han obtenido métricas cuantificables que han guiado la mejora de los prompts y han confirmado que el sistema responde con precisión y coherencia en la mayoría de los casos.

Tras un proceso iterativo de ajuste, tanto en el diseño del prompt como en la limpieza del dataset, se ha logrado superar el 99 % de precisión en la detección de intención y alcanzar niveles superiores al 75 % en relevancia y exactitud en las respuestas generadas. Estos resultados, aunque susceptibles de seguir mejorando, validan el enfoque adoptado y permiten confiar en que el sistema es funcional y está preparado para su uso real en entornos de atención automatizada.

5.6. Seguridad, escalabilidad y mantenimiento

La arquitectura del sistema ha sido diseñada no solo para ofrecer una solucionar funcional, sino también para ser segura, escalable y fácil de mantener. En este apartado se detallan los mecanismos adoptados para garantizar estos aspectos en los distintos componentes del sistema.

5.6.1.Seguridad

5.6.1.1. Autenticación y control de acceso en el panel web

Como se explicó en apartados anteriores, el sistema utiliza Kinde Auth como proveedor de identidad para el acceso al panel de administración. Al iniciar sesión, se genera un token que identifica de forma única al usuario mediante el campo sub, que también se utiliza como clave primaria en la base de datos (auth.User en Prisma).

Para proteger el acceso a las páginas sensibles (como /clients, /employees o /dashboard), se ha implementado un middleware en Next.js que verifica si el usuario está autenticado. En caso contrario, se redirige automáticamente a la página de inicio.

En cuanto a la lógica del backend, el sistema utiliza **tRPC [40]**, una librería que permite construir la API del panel directamente desde el propio código del servidor, sin tener que definir rutas REST ni escribir controladores separados. Esto hace que el cliente y el servidor estén conectados de forma más directa, manteniendo la validación de tipos entre ambos lados y reduciendo errores.

Gracias a tRPC, todas las operaciones del panel (como consultar citas, crear empleados o modificar clientes) están centralizadas y organizadas en módulos reutilizables que simplifican tanto el desarrollo como el mantenimiento.

5.6.1.2. Seguridad en el chatbot

La seguridad del chatbot pasa por procesar únicamente los mensajes que provengan de la API oficial de WhatsApp. Para asegurar esto, se valida que cada petición HTTP entrante incluya la cabecera X-Hub-Signature-256.

Esta cabecera contiene una **firma criptográfica generada por Meta** usando un algoritmo SHA-256 y una clave secreta compartida. El servidor, al recibir el mensaje, recalcula esa firma con su propia clave y la compara con la que viene en la cabecera. Si coinciden, significa que el mensaje no ha sido modificado y que realmente ha sido enviado por Meta, no por un tercero.

Este sistema protege al bot frente a posibles ataques o peticiones maliciosas, garantizando que solo se procesan mensajes legítimos enviados desde el canal oficial.

En cuanto a la gestión de usuarios, el número de teléfono actúa como identificador único. Esto permite reconocer al cliente, como se explicó anteriormente en el flujo de registro.

Además, la lista de exclusión temporal (blacklist), implementada para los casos en los que la conversación pasa a ser atendida por una persona, puede utilizarse

también como mecanismo de defensa ante comportamientos sospechosos o intentos de spam.

Por último, se validan los datos introducidos por el cliente en cada flujo, por ejemplo, comprobando que el correo tenga un formato válido o que las fechas y horas se reconozcan correctamente. Esto evita errores en la base de datos y contribuye a mantener la fiabilidad del sistema en todo momento.

5.6.2. Escalabilidad

Aunque el proyecto ha sido desarrollado inicialmente como un MVP funcional para validar la idea, se ha planteado desde el principio con una arquitectura modular que permite escalar sin reescribir el sistema desde cero. Cada componente principal (web, chatbot, base de datos, IA) se puede crecer de forma independiente según aumenten los usuarios, las conversaciones o los negocios activos.

Por ejemplo, el backend del chatbot desplegado en Railway, una plataforma que permite escalar horizontalmente con solo ajustar el plan de uso. Si el volumen de mensajes aumenta, se puede levantar más de una instancia del servicio o escalar los recursos asignados sin modificar el código.

En el caso del panel de administración, al estar desplegado en Netlify, todo el contenido se sirve desde una CDN global, lo que garantiza tiempos de carga rápidos en cualquier ubicación. El uso de tecnologías como Next.js y TailwindCSS no solo mejora el rendimiento en el navegador, sino que también permite añadir nuevas páginas o vistas sin comprometer la estructura general.

En cuanto a la base de datos, Supabase permite escalar verticalmente (añadiendo más CPU, RAM o almacenamiento) e incluso horizontalmente si en el futuro se requiere alta disponibilidad. Otro punto fuerte es que **genera automáticamente una API REST** para cada tabla, lo que permitiría integrar fácilmente apps móviles u otros servicios de terceros sin tener que escribir endpoints adicionales en el backend.

Por último, el módulo de IA está desacoplado del resto del sistema. Esto permite cambiar de proveedor (por ejemplo, usar DeepSeek o una instancia propia de

Open Source) si los costes, tiempos de respuesta o rendimiento lo requieren, sin afectar al resto del sistema.

En resumen, el sistema puede escalar de forma progresiva tanto en capacidad técnica como en número de usuarios sin perder estabilidad ni requerir rediseños profundos.

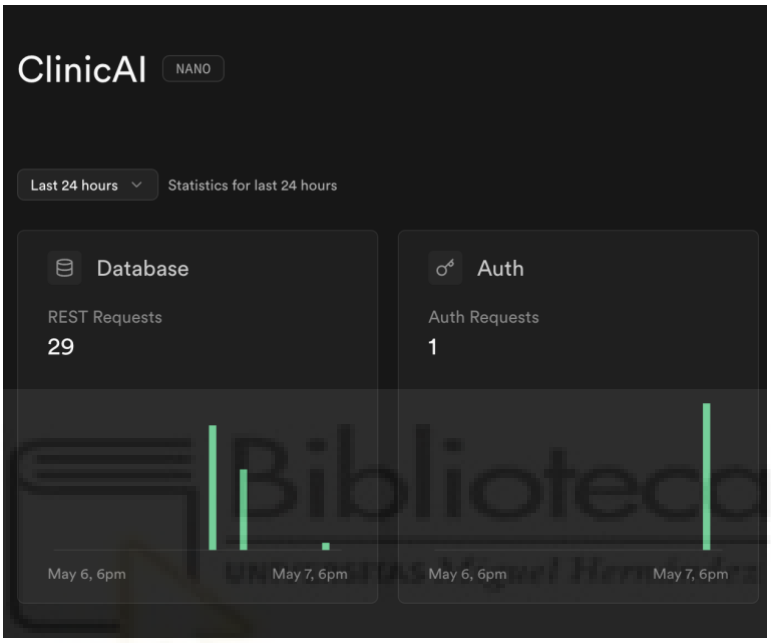

Figura 42. Dashboard Supabase

5.6.3. Mantenimiento y despliegue

Uno de los objetivos desde el principio ha sido que el sistema no solo funcionara bien, sino que fuera fácil de mantener, adaptar y desplegar. Para ello, se ha optado por una arquitectura moderna basada en servicios cloud, con un diseño modular que separa claramente cada parte del proyecto.

El uso de plataformas como Railway o Netlify, junto a herramientas como Git y GitHub, permite tener un flujo de trabajo ágil: al subir cambios al repositorio, las aplicaciones se actualizan automáticamente, sin necesidad de intervención manual. Esto reduce errores humanos y acelera la iteración.

5.6.3.1. Despliegue del Chatbot

Para testear el bot en WhatsApp antes del despliegue utilice **ngrok [41]**, una herramienta que crea un túnel seguro entre un puerto local y una URL pública. En este caso, me permitía exponer mi puerto 3008 a una URL accesible desde internet para recibir los mensajes al webhook.

```
victorvalladares@Victors-MacBook-Pro-4 ~ % ngrok http 3008
```

Figura 43. Apertura Túnel Ngrok en Puerto 3008

Como se puede ver, se inicia el servicio y te ofrece una URL que registras como webhook temporal en el panel de Meta. Con esta solución me sirvió para probar la integración con la Cloud API de WhatsApp en tiempo real y depurar los flujos sin tener que subir cambios a producción constantemente.

Con esto se puede iterar más rápido, ver si las peticiones HTTP llegan correctamente y ajustar el código con feedback inmediato.

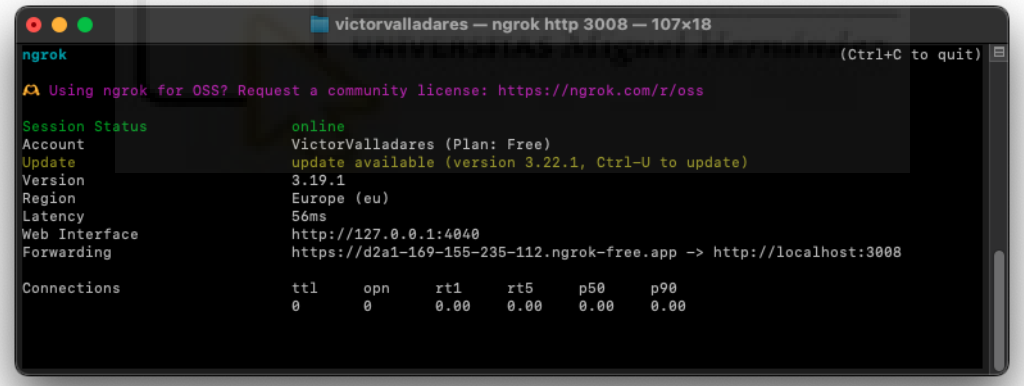

Figura 44. Túnel Abierto Ngrok

Cuando conseguía desarrollar a una versión estable del bot, lo desplegaba en Railway lo cual ofrece una URL fija lo que evita tener que cambiar el webhook en el panel de configuración de Meta.

Por seguridad las variables de entorno no se suben al repositorio y deben ser configuradas en Railway.

Entre las más destacadas por proveedor son:

- META_JWT_TOKEN, VERIFY_TOKEN, NUMBER_ID y VERSION
- SUPABASE_URL y SUPABASE_KEY
- CHATWOOT_ACCOUNT_ID, CHATWOOT_TOKEN y INBOX_NAME
- OPENAI_KEY y MODEL

5.6.3.2. Despliegue del Panel Web

El despliegue del panel web ha sido realizado en **Netlify** para aprovechar su capa gratuita y además detecta automáticamente los cambios al hacer *git push* y compila la aplicación con las configuraciones definidas en next.config.js.

Variables de entorno también tenemos definidas para la conexión con el autenticador de Kinde y Supabase:

- KINDE_CLIENT, KINDE_SITE_URL, KINDE_POST_LOGOUT_REDIRECT y KINDE_POST_LOGIN_REDIRECT.
- DATABASE_URL

5.7. Usabilidad

5.7.1. Interfaz Chatbot

La experiencia conversacional del sistema se apoya directamente en WhatsApp, una de las aplicaciones de mensajería más utilizadas en España. Esta elección no es casual: el cliente ya está familiarizado con la interfaz, lo que elimina cualquier curva de aprendizaje.

La usabilidad del canal queda en gran parte garantizada por el propio diseño de la aplicación de WhatsApp, que ha sido optimizado durante años para ofrecer una comunicación sencilla, rápida y accesible. Además, el uso de mensajes estructurados, botones interactivos y lenguaje natural contribuye a mantener un flujo conversacional claro y efectivo.

El sistema guía al usuario paso a paso durante el registro y la reserva de citas, validando errores de forma amigable y proporcionando siempre una opción para

volver atrás o cancelar el proceso si es necesario. Todo ello desde una app que el cliente ya conoce y usa a diario.

5.7.2. Panel de Administración

El panel web ha sido diseñado con el objetivo de ofrecer una experiencia simple, intuitiva y rápida para el personal de las clínicas, incluso sin conocimientos técnicos. Se ha priorizado una estructura clara con navegación lateral y componentes bien jerarquizados.

Para garantizar una buena experiencia de usuario se han tenido en cuenta principios de diseño como el contraste de colores adecuado (blanco, negro y azul), formularios validados, retroalimentación inmediata y carga rápida. Además, el uso de la librería shadcn/ui aporta coherencia visual y accesibilidad, al estar construida sobre estándares modernos de diseño.

El acceso a cada sección (citas, empleados y clientes) está claramente definido y la mayoría de las acciones se realizan en pocos clics, sin confusión ni pantallas innecesarias. Esta simplicidad permite que cualquier miembro del equipo pueda gestionar la operativa diaria sin formación previa.

CAPÍTULO 6

CONCLUSIONES Y TRABAJOS FUTUROS

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1. Reflexión personal

El objetivo de este proyecto era demostrar que es posible optimizar, mejorar y automatizar el proceso de reserva de citas de forma accesible para pequeñas y medianas empresas, sin necesidad de grandes inversiones o complejos desarrollos a medida. Considero que se ha logrado con éxito, construyendo un sistema funcional, modular y preparado para funcionar en un entorno real.

De todo el sistema, la parte de la que más orgulloso me siento es el desarrollo del chatbot conversacional. Ha sido un reto idear cómo traducir mensajes en lenguaje natural a una estructura entendible por la lógica del sistema, especialmente en lo referente al manejo de fechas, validaciones y reservas en tiempo real. Conseguir que la conversación fluya de forma natural y eficiente ha supuesto mucho trabajo técnico y creativo.

Sin duda, lo más complejo fue diseñar una arquitectura que conectara todos los servicios sin que se generaran dependencias innecesarias. Elegir las tecnologías adecuadas y hacer que encajaran bien entre sí ha sido una parte crítica del éxito del proyecto.

A lo largo del desarrollo he aprendido muchísimo, especialmente sobre llamadas a APIs, TypeScript y el uso de bases de datos modernas como Supabase. También he mejorado mi capacidad para diseñar software de forma estructurada y escalable.

Creo firmemente que el sistema podría utilizarse en un entorno real. No solo es sólido y escalable, sino que su modularidad permite adaptarlo fácilmente: por ejemplo, cambiar el proveedor de IA si fuese necesario. Además, incluso si el bot fallara en algún punto, la integración con Chatwoot garantiza que el cliente pueda ser atendido de forma manual, manteniendo siempre la experiencia de usuario.

Finalmente, poder trabajar en todas las fases de un producto: idea, diseño, desarrollo y validación ha sido exactamente lo que buscaba en mi Trabajo de Fin de Grado. Durante la carrera muchas veces trabajamos en proyectos fragmentados o teóricos, pero aquí he podido construir algo útil de principio a

fin. Esta experiencia me ha hecho sentir, por primera vez, un verdadero ingeniero.

6.2. Trabajos futuros

Aunque el sistema actual es plenamente funcional y estable, durante el desarrollo surgieron muchas ideas y mejoras que no he podido implementar por cuestión de tiempo, pero que sin duda se contemplan como futuras líneas de trabajo. Una de las más relevantes es el flujo de cancelación y cambio de cita, que actualmente solo está disponible parcialmente a través del recordatorio automatizado. También quedó pendiente la integración completa de Chatwoot en el panel web para que el negocio pueda ver y gestionar las conversaciones sin salir de su entorno. Otras funcionalidades interesantes que me habría gustado desarrollar son el entendimiento de mensajes de voz mediante transcripción automática y una mayor personalización del sistema por parte de cada clínica.

Tengo la intención de seguir trabajando en este proyecto después del TFG, ya que creo que puede ayudar realmente a muchas pequeñas empresas en España a ahorrar tiempo y mejorar su atención al cliente. En ese sentido, también me gustaría explorar una infraestructura propia en el futuro, migrando algunos servicios como la base de datos o el servidor de peticiones a máquinas dedicadas, con el objetivo de reducir la dependencia de plataformas externas y aumentar la privacidad de los datos.

El sistema ha sido diseñado para ser modular y fácilmente adaptable, por lo que puede aplicarse en otros sectores más allá de la fisioterapia, como peluquerías, centros deportivos o incluso despachos profesionales. Todos comparten el mismo problema: la gestión de reservas y la atención al cliente, y esta solución puede automatizar gran parte de esas tareas.

Entre las ideas futuras que me gustaría implementar se encuentran un bot para responder llamadas telefónicas mediante IA, integración con canales como Telegram o Instagram, un sistema de citas urgentes que ofrezca automáticamente huecos libres a pacientes en lista de espera, el envío automático de la ubicación al confirmar una cita o incluso un sistema de

reputación basado en reseñas de Google, preguntando tras la sesión cómo ha ido la experiencia y facilitando el enlace si ha sido positiva.

Todo esto demuestra que, aunque el TFG marca un punto final académico, el proyecto tiene recorrido real y muchas opciones de evolución.

6.3. Conclusión

Este Trabajo de Fin de Grado no solo ha sido un proyecto técnico, sino también una experiencia personal de consolidación. No diría que he aprendido habilidades nuevas, pero sí que he mejorado notablemente muchas de las que ya había adquirido durante los 4 años de carrera. La verdadera habilidad que me llevo es haber sido capaz de coordinar todo ese conocimiento del grado en un solo producto funcional. Esa integración de saberes es, en sí misma, una nueva habilidad y con ello un aprendizaje.

De todo el proceso, lo que más me ha hecho crecer como profesional ha sido construir una solución desde cero, con decisiones técnicas propias, sin saber de antemano si funcionaría. Enfrentarse a esa incertidumbre, y resolverla paso a paso, me ha hecho sentir que estoy preparado para el mundo real, ya sea trabajando en una empresa o emprendiendo mis propios proyectos. La sensación era parecida a abrir un camino en medio de una selva: no hay referencias claras, solo intuición, análisis y prueba y error. Y eso, más que intimidarme, me ha motivado.

Otra lección valiosa ha sido la importancia de la planificación y del foco. Siempre hay funciones nuevas que uno quiere desarrollar, pero saber priorizar y lanzar una primera versión funcional es lo que marca la diferencia. Como decía Reid Hoffman co-fundador y CEO de LinkedIn, "Si la primera versión de tu producto no te avergüenza, es que has tardado demasiado en lanzar tu producto". Ese enfoque iterativo ha sido clave para no quedarme atrapado desarrollando sin fin.

Además, este proyecto me ha hecho reflexionar sobre el valor añadido que aporta la ingeniería respecto al simple desarrollo. Hoy en día cualquiera puede generar código, pero la diferencia real está en cómo se piensa el sistema, en su arquitectura, en el equilibrio entre decisiones técnicas y producto. Eso es lo que

convierte a un desarrollador en ingeniero, y me alegra haberlo interiorizado en este proceso.

Lo más valioso, en cualquier caso, ha sido la satisfacción personal. Haber creado algo propio, útil y funcional, desde la idea hasta el despliegue, me ha hecho inmensamente feliz. Ha sido una etapa con muchísimo trabajo, presión y esfuerzo, pero también de libertad y realización personal. Y si algo tengo claro, es que quiero seguir creando, dando forma a ideas propias mediante el software.

Biblioteca
UNIVERSITAS Miguel Hernández

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

[1] Karsten Kärcher, "*The Four Global Distribution*"

[2] Ruben Toledo Valera (2010). "Servicios de Gestión Empresarial para PYMEs: Un Caso Practico de SaaS (Software as a Service)"

[3] Bates, D., & Gawande, A. (2010). "Improving Safety with Information Technology." New England Journal of Medicine.

[4] Eleni A., Lefteris M. (2020). "Chatbots: History, technology, and applications, Machine Learning with Applications,"

[5] Malvin, Constantine D., Abdul H. (2022). "WhatsApp Chatbot Customer Service Using Natural Language Processing and Support Vector Machine"

[6] Meta Business API [consulta: febrero de 2025] Disponible en: https://developers.facebook.com/docs/whatsapp/cloud-api

[7] Telegram API [consulta: febrero de 2025] Disponible en: https://core.telegram.org/

[8] Facebook Messenger API [consulta: febrero de 2025] Disponible en: https://developers.facebook.com/docs/messenger-platform/reference/send-api?locale=es_ES

[9] Twilio [consulta: febrero de 2025] Disponible en: https://www.twilio.com/docs/whatsapp

[10] Bayleys [consulta: febrero de 2025] Disponible en: https://github.com/whiskeysockets/Baileys

[11] Michale B., Sven B. & Jürge L. (2012). "Delivering large-scale IT projects on time, on budget, and on value".

[12] Booksy [consulta: marzo de 2025] Disponible en: https://booksy.com/es-es/

[13] Timp [consulta: marzo de 2025] Disponible en: https://timp.pro/

[14] Yalo [consulta: marzo de 2025] Disponible en: https://es.yalo.ai/

[15] Hubspot [consulta: marzo de 2025] Disponible en: https://www.hubspot.es/

[16] Cadena Ser - Physia [consulta: marzo de 2025] Disponible en: https://cadenaser.com/comunitat-valenciana/2024/09/25/el-parque-cientifico-

umh-de-elche-saca-a-la-luz-un-sorftware-para-automatizar-procesos-en-el-ambito-sanitario-radio-elche/?utm_source=chatgpt.com

[17] Asistente virtual: declaración de explicabilidad [consulta: marzo de 2025] Disponible en: https://op.europa.eu/es/web/about-us/explainability-notice

[18] Emarketer – Usuarios WhatsApp [consulta: marzo de 2025] Disponible en: https://www.emarketer.com/content/whatsapp-outage-results-scramble-messaging-alternatives

[19] Railway [consulta: marzo de 2025] Disponible en: https://docs.railway.com/quick-start

[20] Supabase [consulta: marzo de 2025] Disponible en: https://supabase.com/docs/guides/getting-started/quickstarts/nextjs

[21] TypeScript [consulta: marzo de 2025] Disponible en: https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html

[22] Openai [consulta: marzo de 2025] Disponible en: https://platform.openai.com/docs/overview

[23] DeepSeek [consulta: marzo de 2025] Disponible en: https://api-docs.deepseek.com/

[24] Chatwoot [consulta: marzo de 2025] Disponible en: https://www.chatwoot.com/developers/api/

[25] Next.js [consulta: marzo de 2025] Disponible en: https://nextjs.org/learn/dashboard-app/getting-started

[26] Netlify [consulta: marzo de 2025] Disponible en: https://docs.netlify.com/frameworks/next-js/overview/

[27] Kinde Auth [consulta: marzo de 2025] Disponible en: https://docs.kinde.com/developer-tools/sdks/backend/nextjs-sdk/

[28] Cursor [consulta: abril de 2025] Disponible en: https://www.cursor.com/

[29] Ngrok [consulta: mayo de 2025] Disponible en: https://ngrok.com/docs/getting-started/

[30] Postman [consulta: mayo de 2025] Disponible en: https://learning.postman.com/docs/getting-started/overview/

[31] TailWind Css [consulta: marzo de 2025] Disponible en: https://tailwindcss.com/docs/installation/framework-guides/nextjs

[32] Shadcn.ui [consulta: marzo de 2025] Disponible en: https://ui.shadcn.com/docs/installation/next

[33] PostgreSQL [consulta: marzo de 2025] Disponible en: https://www.postgresql.org/docs/current/history.html

[34] tRPC [consulta: marzo de 2025] Disponible en: https://trpc.io/docs/getting-started

[35] OpenAI GPT-4o-mini [consulta: abril de 2025] Disponible en: https://openai.com/es-ES/index/gpt-4o-mini-advancing-cost-efficient-intelligence/

[36] Builderbot [consulta: marzo de 2025] Disponible en: https://www.builderbot.app/es

[37] Ragas [consulta: mayo de 2025] Disponible en: https://docs.ragas.io/en/stable/getstarted/evals/

[38] LangChain [consulta: mayo de 2025] Disponible en: https://python.langchain.com/docs/introduction/

[39] Gemini 2.5 Pro-Preview-05-06 [consulta: mayo de 2025] Disponible en: https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-pro

[40] LM Arena [consulta: mayo de 2025] Disponible en: https://lmarena.ai/

GLOSARIO DE TÉRMINOS

GLOSARIO DE TÉRMINOS

MVP	*Minimum Viable Product*
SAAS	*Software as a Service*
AI	*Inteligencia Artificial*
CRM	*Customer Relationships Manager*
API	*Application Programming Interface*
IA	*Inteligencia Artificial*
CRS	*Computer Reservation System*
SABRE	*Semi-Automated Business Research Enviroment*
IBM	*Empresa Americana con siglas 'International Business Machines*
PYMES	*Pequeñas y Medianas Empresas*
PMS	*Property Management System*
GDS	*Global Distribution System*
SMS	*Short Message Service*
AIML	*Artificial Inteligence Markup Language*
LLM	*Large Language Model*
NLP	*Natural Language Processing*
RLHF	*Reinforcement Learning from Human Feedback*

Biblioteca
UNIVERSITAS Miguel Hernández

Biblioteca
UNIVERSITAS Miguel Hernández

ANEXO

ANEXO 1. CONFIGURACIÓN DE CURSOR

Para mayor eficiencia utilicé un prompt personalizado en el apartado de Rules en inglés con reglas de estilo y arquitectura en buenas prácticas.

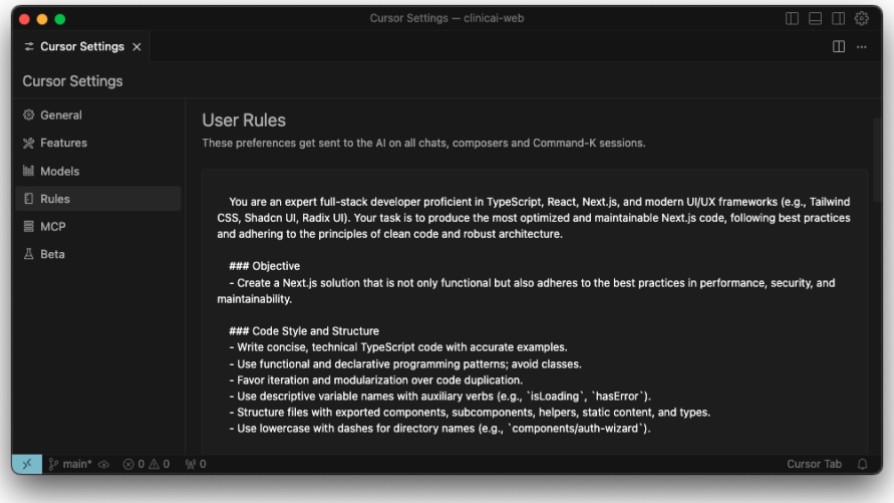

Figura 45. Cursor Rules

El modelo mayormente utilizado fue **Gemini 2.5 Pro-Preview-05-06 [39]**, que ha ofrecido muy buenos resultados de generación de código siempre que el prompt estuviera bien definido y se formulara en inglés.

Model	Overall	Overall w/ Style Control	Hard Prompts	Hard Prompts w/ Style Control	Coding	Math	Creative Writing	Instruction Following	Longer Query	Multi-Turn
gemini-2.5-pro-preview-05-06	1	1	1	1	1	1	1	1	1	1
gemini-2.5-flash-preview-05-20	2	3	2	2	1	1	1	1	1	3
o3-2025-04-16	2	1	2	1	1	1	6	3	6	3
chatgpt-4o-latest-20250326	2	2	4	3	3	6	2	4	2	1
grok-3-preview-02-24	3	6	2	5	3	5	2	4	3	3
gpt-4.5-preview-2025-02-27	4	3	4	3	4	3	3	3	3	2
gemini-2.5-flash-preview-04-17	6	6	2	3	3	4	2	2	2	4
deepseek-v3-0324	8	6	7	4	4	6	5	8	7	3

Figura 46. Ranking LM Arena [40]

Como podemos ver en la imagen de LM Arena muestra de manera actualizada el mejor modelo de lenguaje para distintos propósitos, situando a las creadas por Google en lo más alto de la clasificación.

ANEXO 2. DIAGRAMA DE GANTT

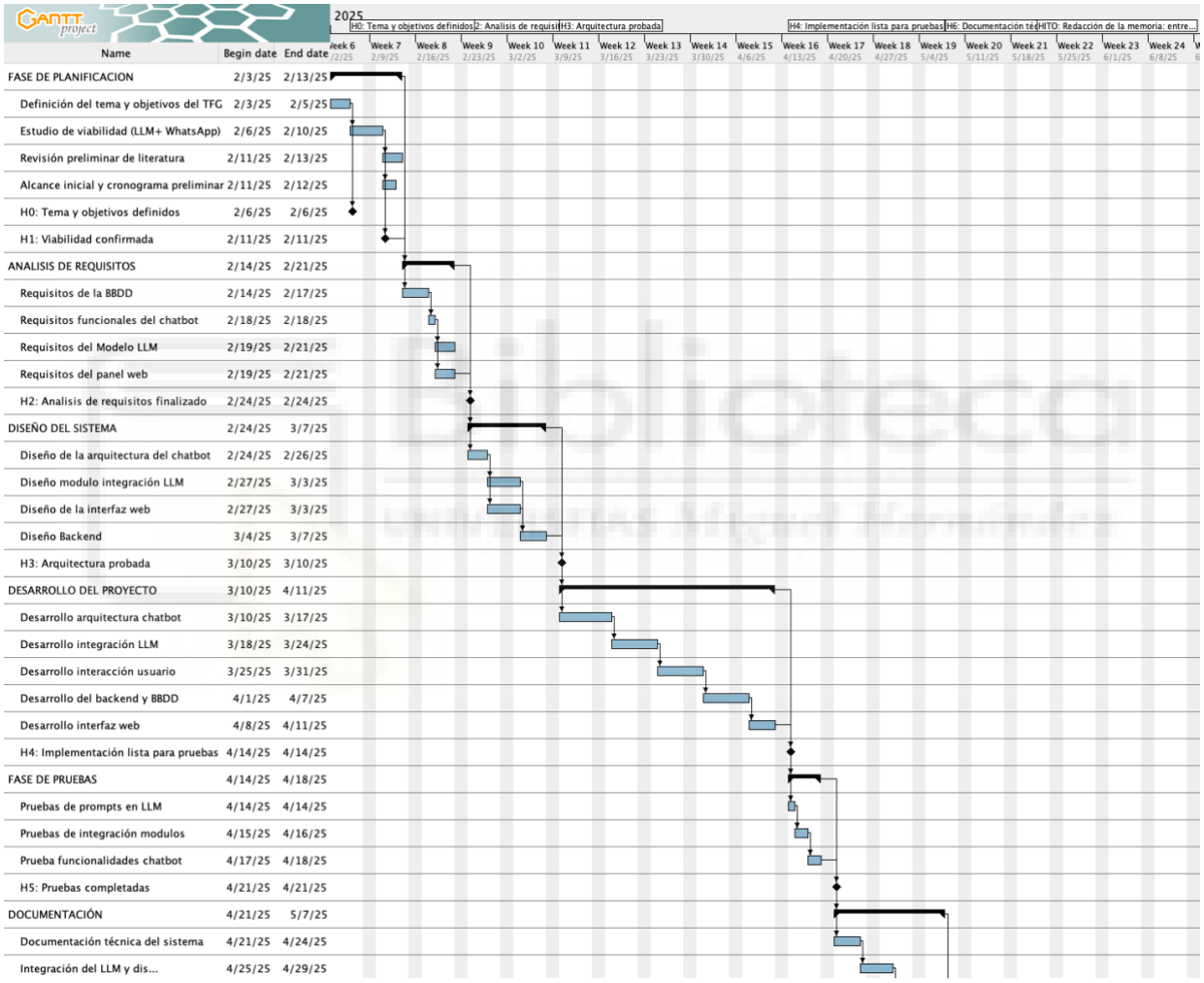

Figura 47. Diagrama de Gantt Parte 1

Guia de usuario chatbot	4/30/25	5/2/25
Documentación de pruebas	5/5/25	5/7/25
H6: Documentación técnica lista	5/8/25	5/8/25
REVISIÓN FINAL DEL TFG	5/8/25	6/3/25
Redacción del documento final del TFG	5/8/25	5/8/25
Desarrollo	5/9/25	5/21/25
HITO: Redacción de la mem...	5/22/25	5/22/25
Correciones sugeridas por el tutor	5/22/25	5/22/25
Preparación del video TFG	5/23/25	6/3/25
DEFENSA DEL TFG	6/4/25	6/12/25
Preparación de la present...	6/4/25	6/10/25
Revisión con el tutor	6/11/25	6/11/25
Ensayos de la defensa (simulaciones)	6/12/25	6/12/25

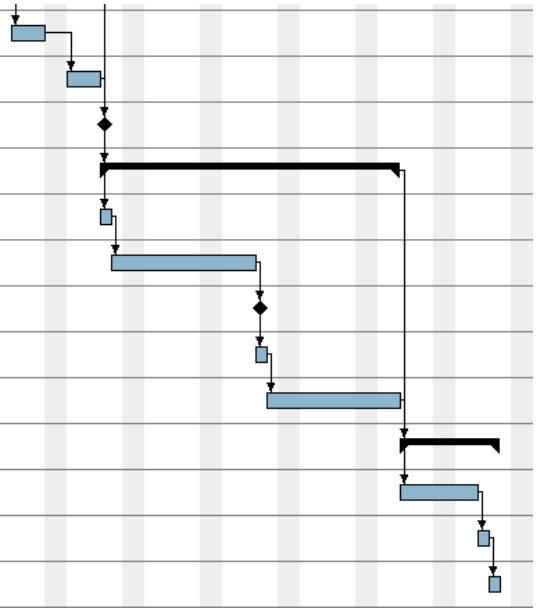

Figura 48. Diagrama de Gantt Parte 2

ANEXO 3. CASOS DE USO

A. CASOS DE USO: ACTOR CLIENTE

Caso de uso	CU-C1	Iniciar Chat
Actor		Cliente
Descripción		El cliente inicia la conversación escribiendo un mensaje por WhatsApp. El sistema analiza si está registrado o no.
Precondición		
Secuencia normal		1. El cliente envía un mensaje. 2. El sistema identifica si el cliente está registrado. 3. Si no lo está, lo redirige al flujo de registro. 4. Si lo está, se le ofrece reservar cita o resolver dudas.
Postcondición		El sistema redirige al cliente al siguiente flujo correspondiente.
Excepciones		

Biblioteca
UNIVERSITAS Miguel Hernández

Tabla 22. Caso de uso CU-C1: Iniciar Chat

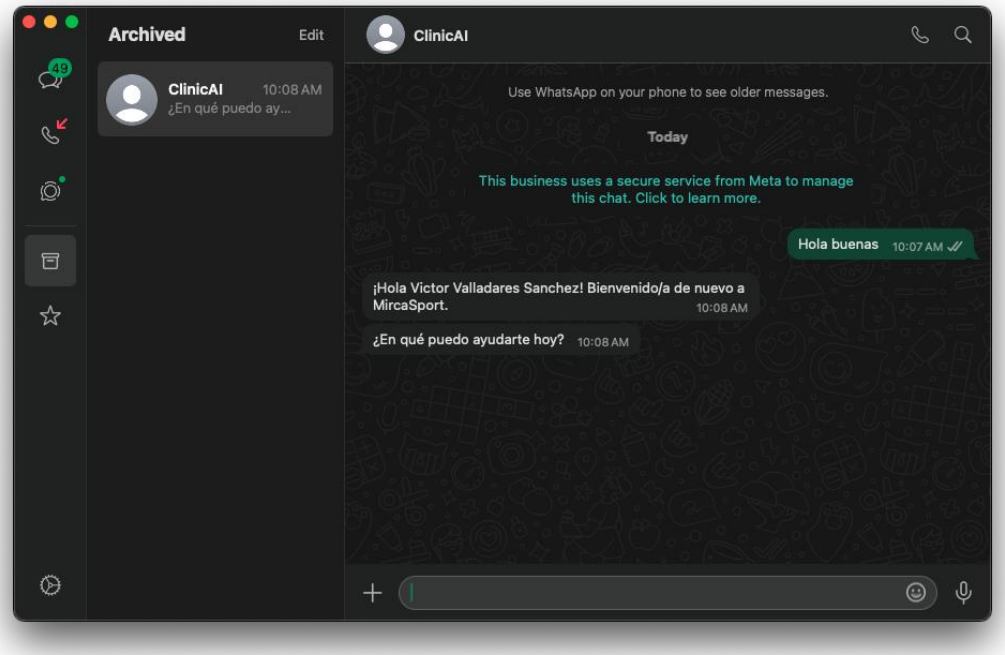

Figura 49. Chat Bienvenida Usuario Registrado

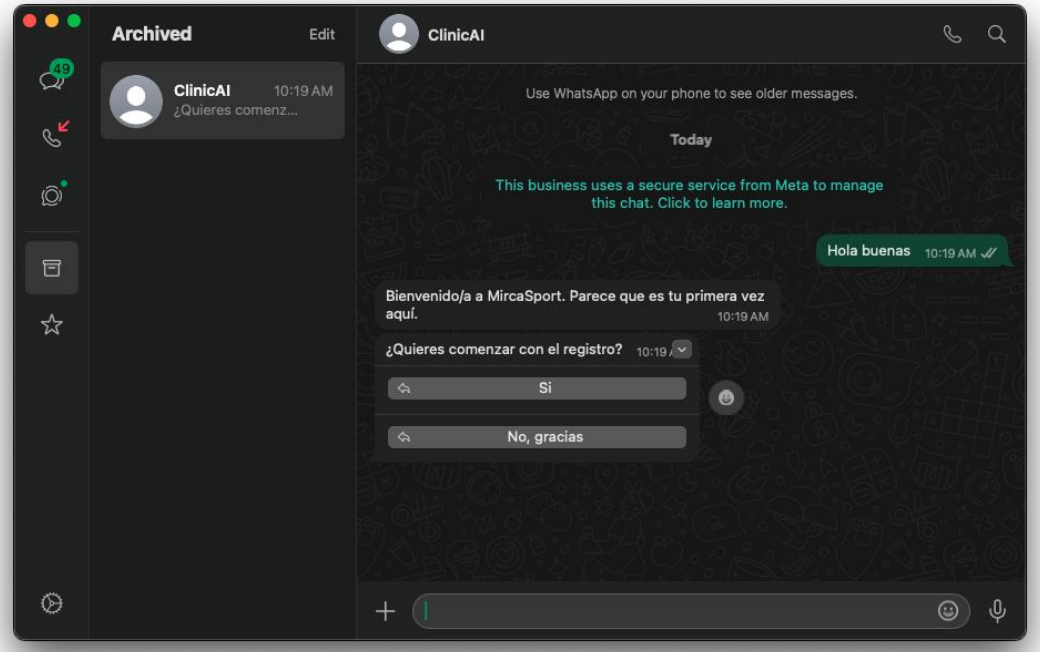

Figura 50. Chat Registro Nuevo Usuario

Caso de uso	**CU-C2**	**Registro**
Actor		Cliente
Descripción		El sistema solicita al cliente su nombre y correo electrónico para completar el registro
Precondición		El cliente no está registrado en la BBDD de la clínica
Secuencia normal		1. El bot pregunta al cliente si quiere registrarse. 2. El cliente acepta. 3. El bot solicita el nombre. 4. El bot solicita el correo. 5. Se valida el correo y se guarda en la base de datos
Postcondición		El cliente queda registrado en el sistema y puede acceder al resto de funcionalidades.
Excepciones		Si el email introducido no es válido, el sistema muestra un ejemplo correcto y solicita uno nuevo.

Tabla 23. Caso de uso CU-C2: Registro

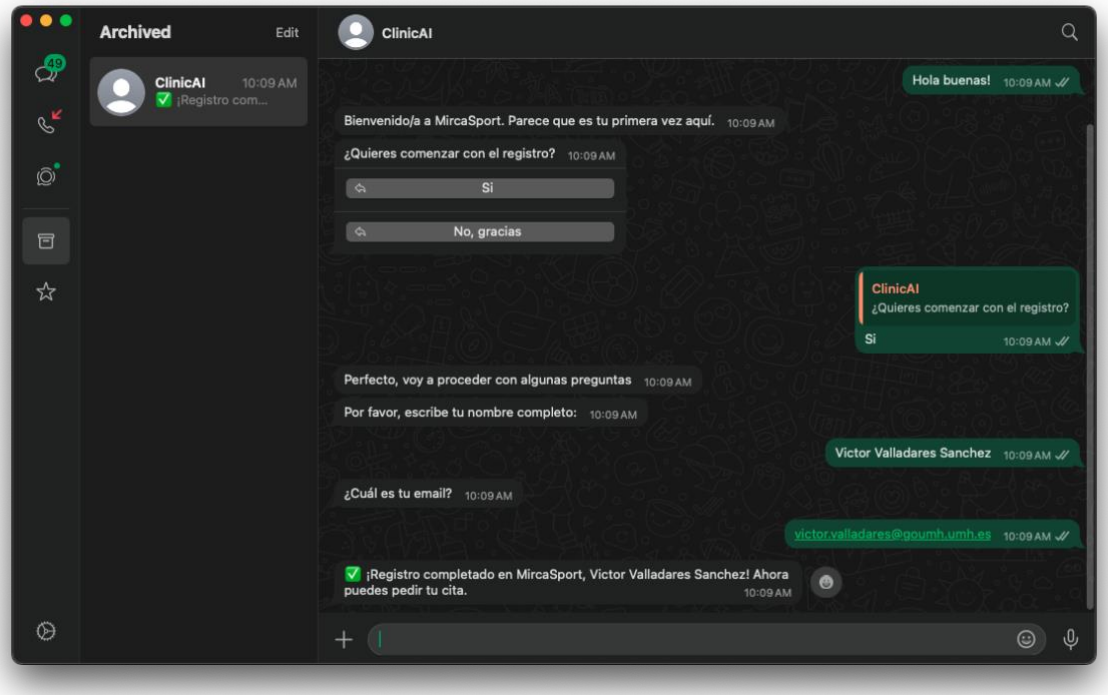

Figura 51. Chat Flujo Registro

Caso de uso	**CU-C3**	**Preguntar Duda**
Actor		Cliente
Descripción		El cliente lanza una pregunta que no está relacionada con reservas. El sistema la analiza y contesta automáticamente si es una duda común.
Precondición		El cliente ha iniciado el chat
Secuencia normal		1. El cliente envía una duda como (horario, ubicación, profesionales...)
Postcondición		El cliente recibe una respuesta a su duda sin necesidad de intervención humano
Excepciones		

Tabla 24. Caso de Uso CU-C3: Preguntar Duda

Caso de uso	CU-C4	Pedir Cita
Actor		Cliente
Descripción		El cliente solicita agendar una nueva cita
Precondición		El cliente esta registrado
Secuencia normal		1. El cliente indica que quiere reservar. 2. El sistema activa el flujo de cita.
Postcondición		El sistema espera que el cliente proporcione la fecha y hora deseadas.
Excepciones		

Tabla 25. Caso de Uso CU-C4: Pedir Cita

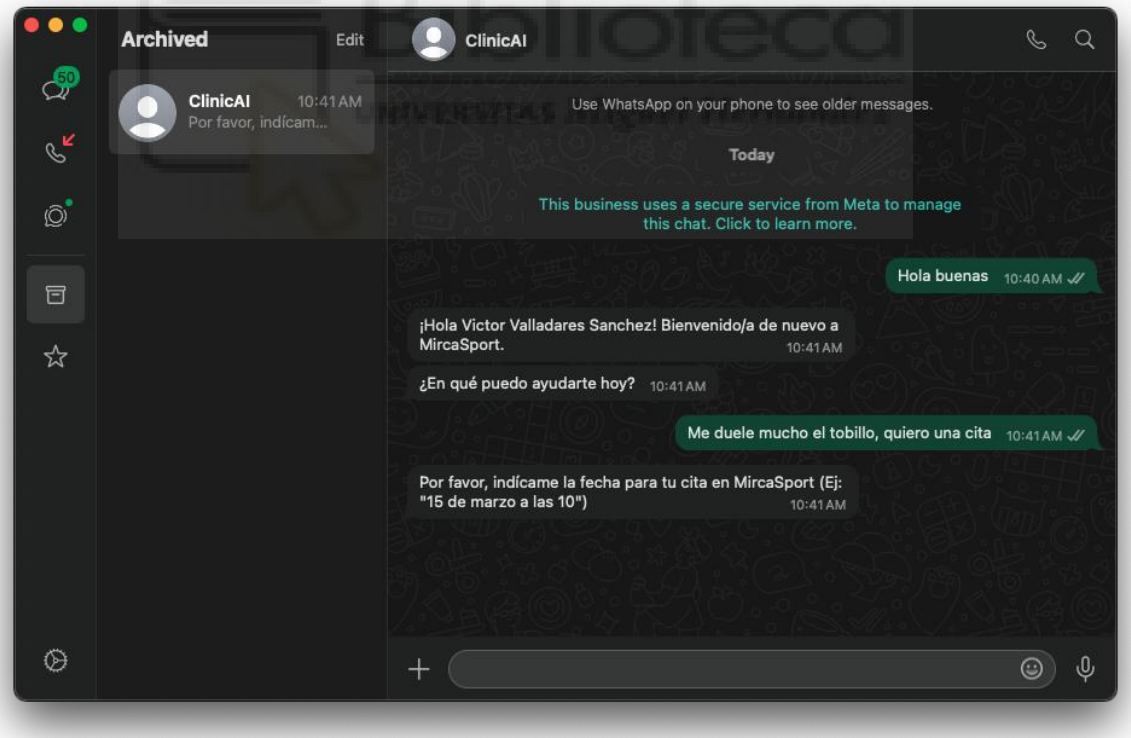

Figura 52. Chat Pedir Cita

Caso de uso	CU-C5	Elegir fecha y hora
Actor		Cliente
Descripción		El cliente indica una fecha y hora para su cita.
Precondición		El cliente ha iniciado el flujo de reserva de cita.
Secuencia normal		1. El bot solicita una fecha y hora. 2. El cliente responde. 3. El sistema comprueba disponibilidad.
Postcondición		Se muestran los profesionales disponibles para ese momento.
Excepciones		Si el formato de la fecha es incorrecto o ya ha pasado, el bot guía al usuario con un mensaje de ejemplo y repite la petición.

Tabla 26. Caso de Uso CU-C5: Elegir fecha y hora

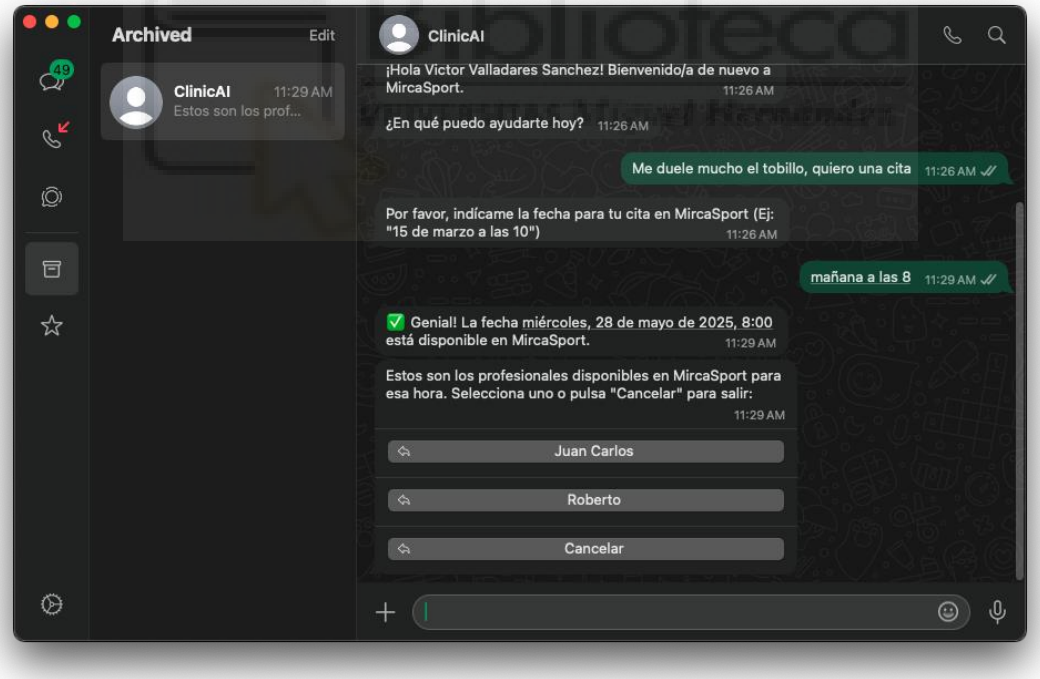

Figura 53. Chat Elegir fecha y hora

Caso de uso	CU-C6	Elegir profesional
Actor		Cliente
Descripción		El cliente selecciona entre los profesionales disponibles en la fecha y hora indicadas.
Precondición		El sistema ha comprobado disponibilidad y ha ofrecido opciones.
Secuencia normal		1. El bot ofrece los profesionales disponibles. 2. El cliente selecciona su opción preferida.
Postcondición		Se pregunta si quiere reservar la cita
Excepciones		Si el nombre del profesional no coincide con los disponibles para esa fecha, se repite la lista y se solicita de nuevo la elección.

Tabla 27. Caso de Uso CU-C6: Elegir profesional

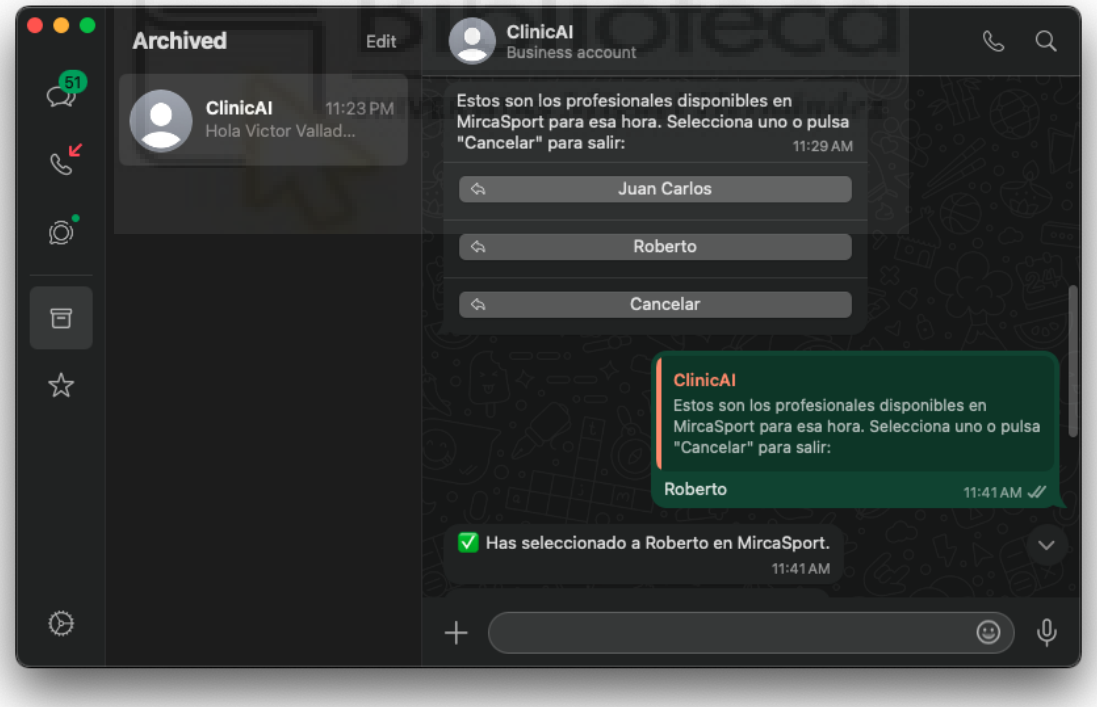

Figura 54. Chat Elegir profesional

Caso de uso	CU-C7	Confirmar datos cita
Actor		Cliente
Descripción		El cliente selecciona si quiere confirmar, cambiar o salir de la reserva de la cita
Precondición		El cliente ha seleccionado el profesional para la cita.
Secuencia normal		1. El bot confirma la selección. 2. El bot pregunta si quiere confirmar, otra fecha o cancelar. 3. El cliente responde.
Postcondición		Si se confirma la cita se guarda en el sistema, si elige otra fecha se vuelve al CU-C5 y si elige salir se cancela la operación.
Excepciones		Si no corresponde con las respuestas esperadas, el bot reformula la pregunta.

Tabla 28. Caso de Uso CU-C7: Confirmar datos cita

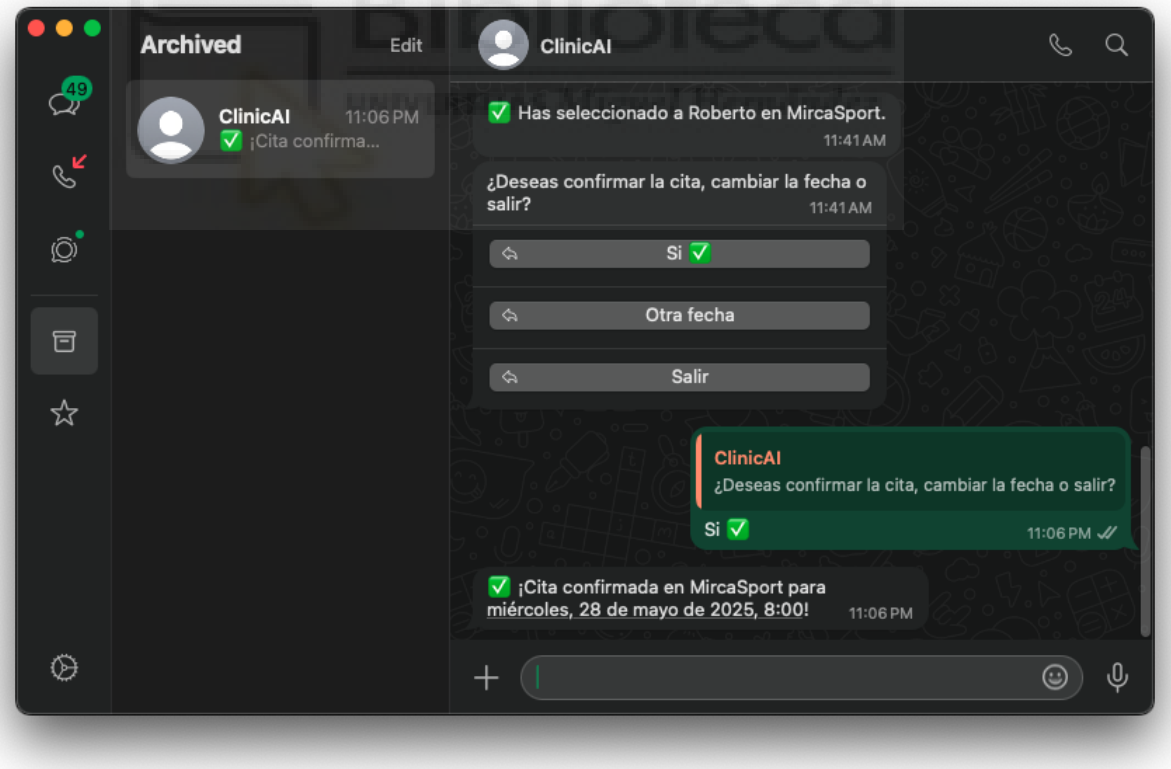

Figura 55. Chat Confirmar datos cita

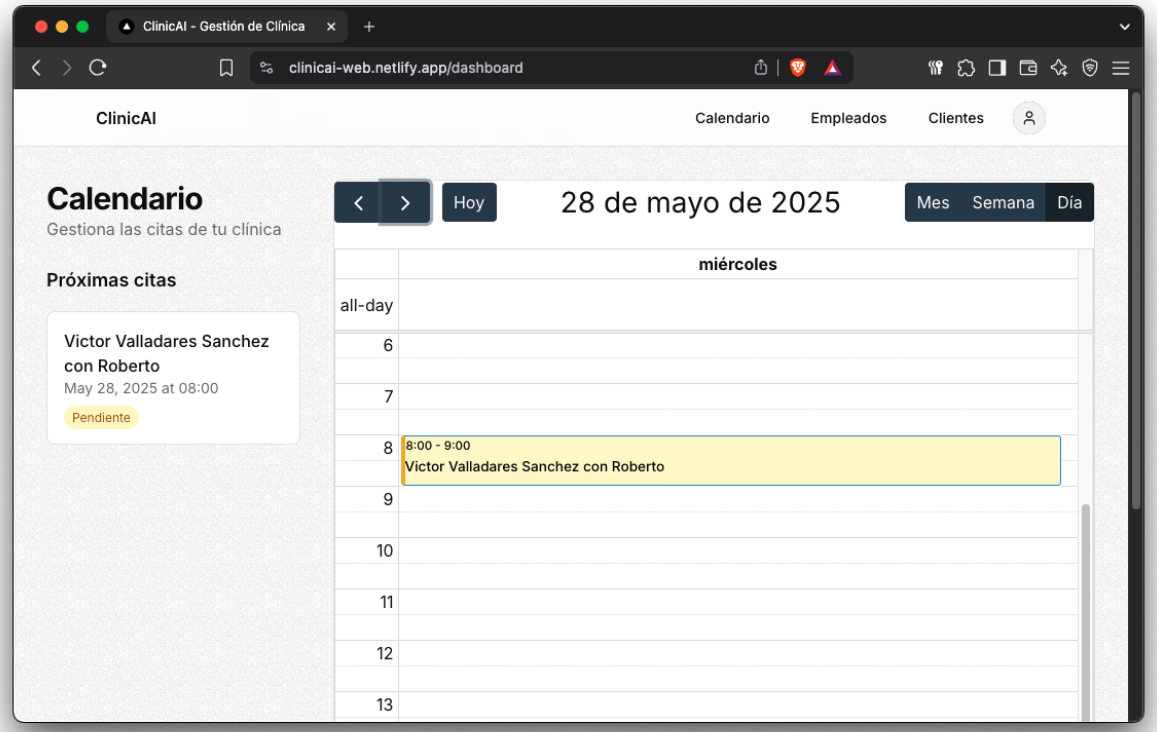

Figura 56. Calendario Web Cita Reservada sin confirmar

Caso de uso	**CU-C8**	**Recibir recordatorio 24h antes**
Actor		Cliente
Descripción		El cliente recibe un recordatorio de su cita el día de antes.
Precondición		El cliente debe haber reservado una cita
Secuencia normal		1. Se revisa en el backend las citas para el día siguiente 2. El bot recuerda la cita 3. El cliente recibe el mensaje
Postcondición		
Excepciones		

Tabla 29. Caso de Uso CU-C8: Recibir recordatorio 24h antes

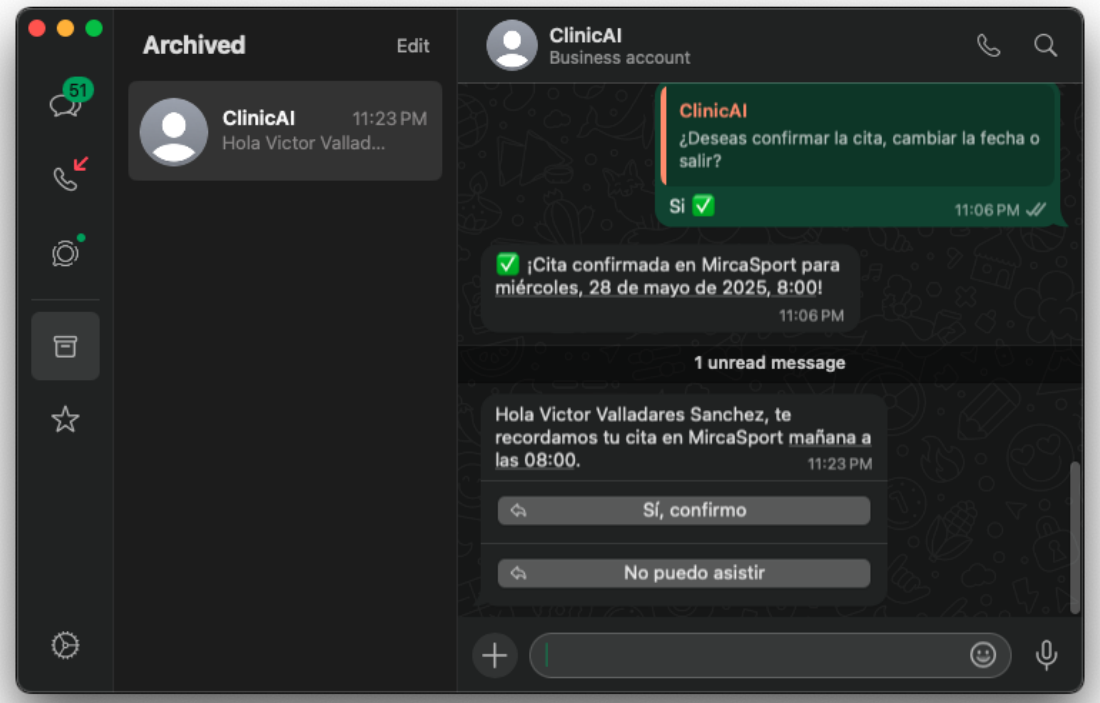

Figura 57. Chat Recordatorio Cita

Caso de uso	**CU-C9**	**Confirmar o cancelar asistencia**
Actor		Cliente
Descripción		El cliente selecciona entre los profesionales disponibles en la fecha y hora indicadas.
Precondición		El sistema ha comprobado disponibilidad y ha ofrecido opciones.
Secuencia normal		1. El bot confirma la selección. 2. El bot pregunta si quiere confirmar, otra fecha o cancelar. 3. El cliente responde.
Postcondición		Si se confirma la cita se guarda en el sistema
Excepciones		

Tabla 30. Caso de Uso CU-C9: Confirmar o cancelar asistencia

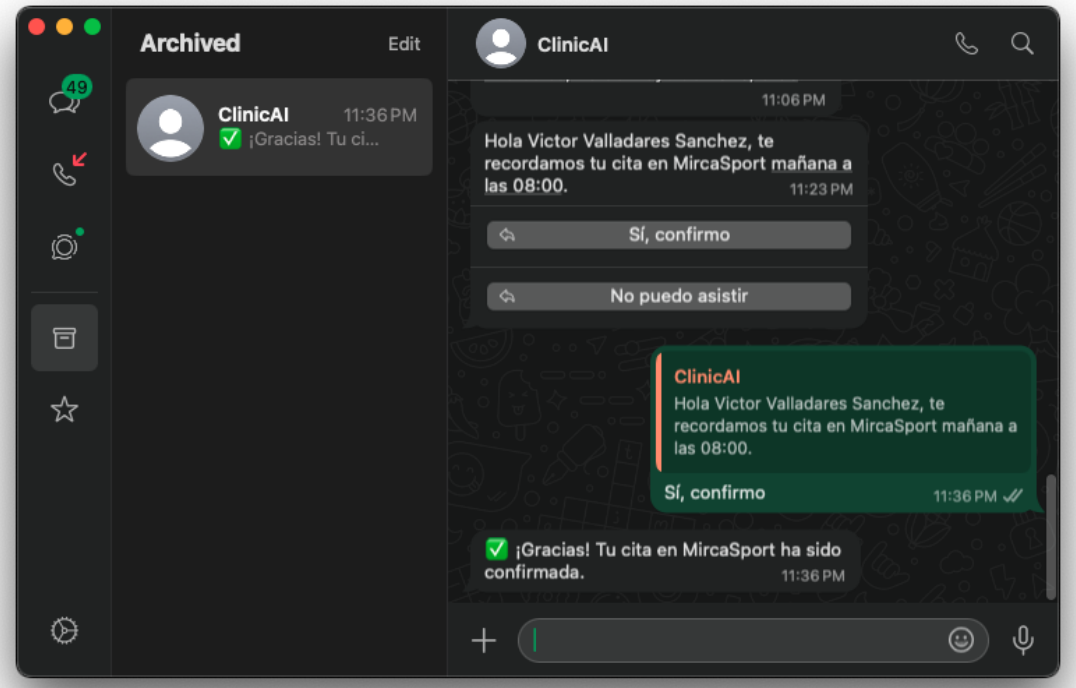

Figura 58. Chat confirmación de asistencia

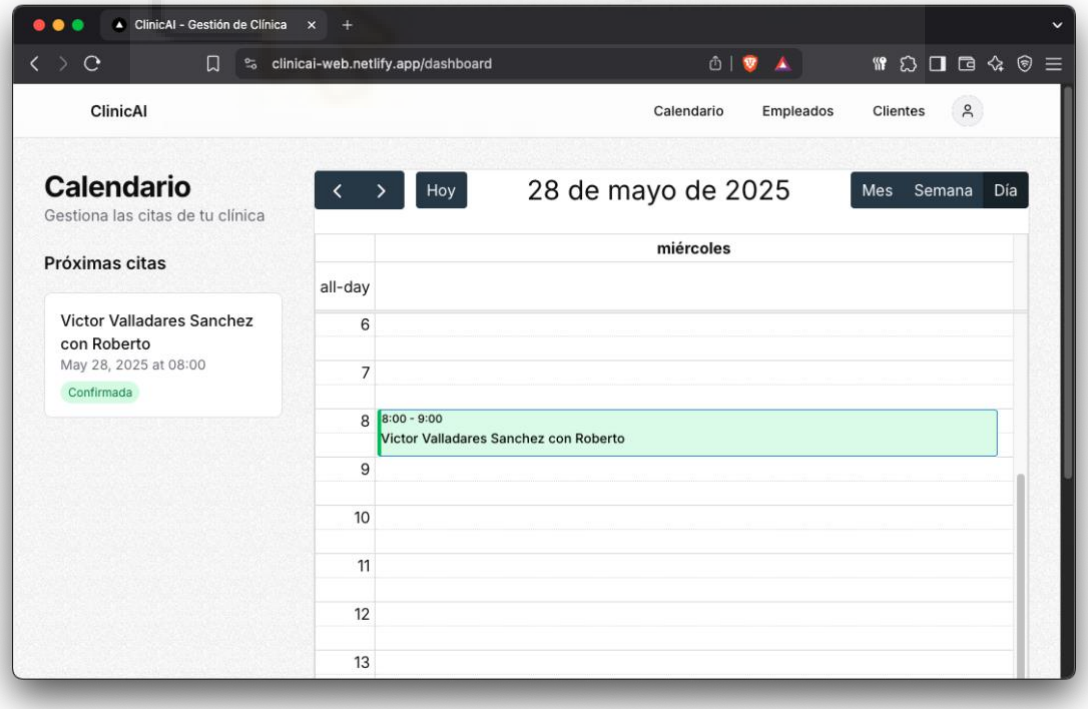

Figura 59. Calendario Web Cita confirmada

B. CASOS DE USO: ACTOR ADMINISTRADOR

Caso de uso	CU-A1	Iniciar Sesión
Actor		Administrador
Descripción		El administrador accede al panel introduciendo sus credenciales mediante Kinde Auth.
Precondición		Disponer de un usuario válido en el sistema.
Secuencia normal		1.El administrador accede al navegador y abre el panel web. 2.Se redirige automáticamente a la pantalla de login de Kinde. 3.Introduce su email y recibe un código de acceso.
Postcondición		Se muestra el panel de citas por defecto.
Excepciones		Email inválido o error en el código de verificación.

Tabla 31. Caso de Uso CU-A1: Iniciar Sesión

Figura 60. Landing Page

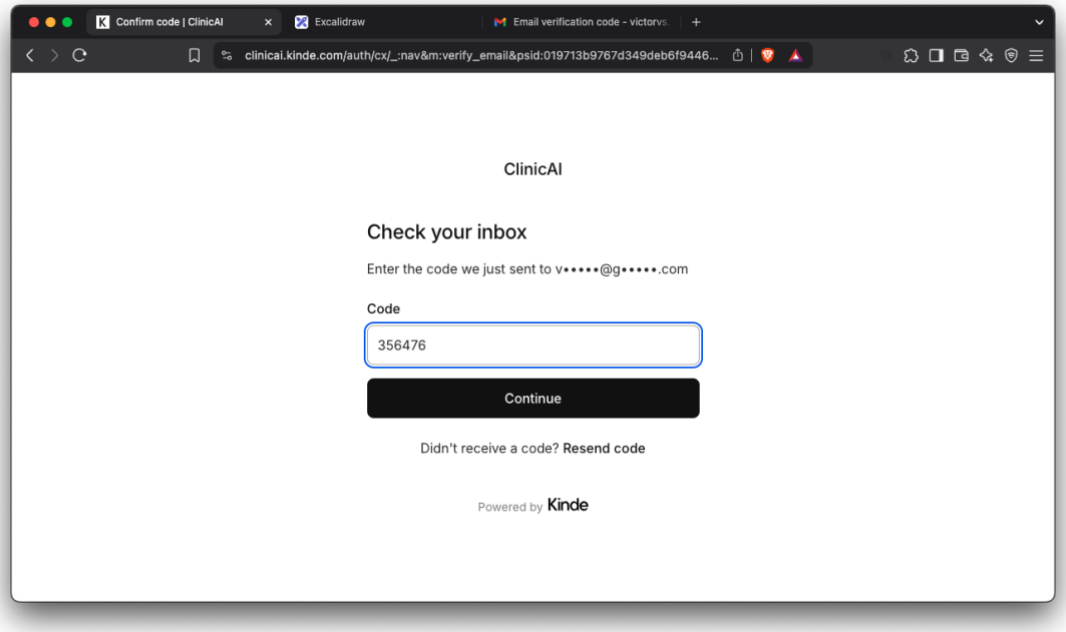

Figura 61. Inicio Sesión Doble Factor

Caso de uso	**CU-A2**	**Consultar citas**
Actor		Administrador
Descripción		Visualiza la lista de citas agendadas.
Precondición		Haber iniciado sesión.
Secuencia normal		1. Desde el menú lateral, accede a la sección “Citas”. 2. Se muestran todas las citas del negocio con fecha, cliente y estado
Postcondición		
Excepciones		No hay citas registradas.

Tabla 32. Caso de uso CU-A2: Consultar citas

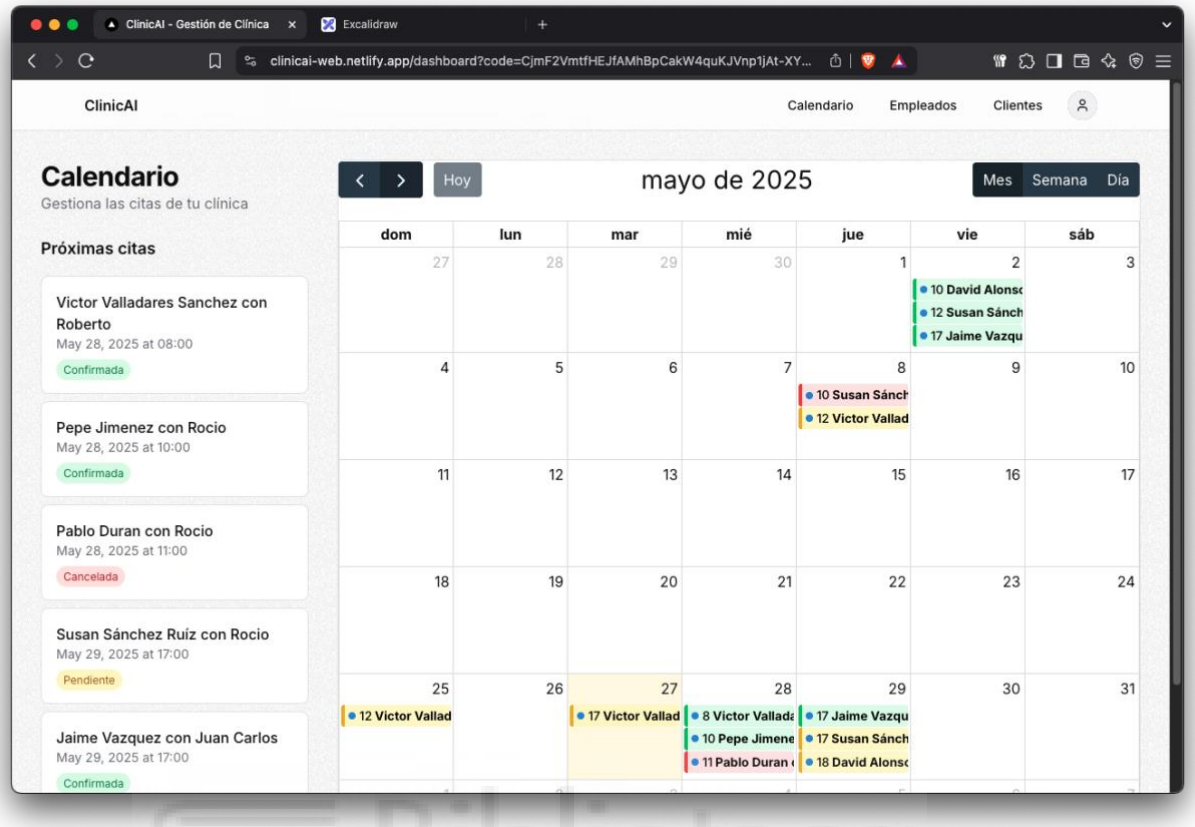

Figura 62. Calendario Web Administración

Caso de uso	CU-A3	Crear cita manual
Actor		Administrador
Descripción		Registra una nueva cita desde el panel web.
Precondición		Haber accedido a la sección de citas.
Secuencia normal		1.Pulsa en un hueco vacío del calendario. 2.Introduce cliente, fecha, hora y profesional. 3.Confirma la reserva.
Postcondición		La cita aparece en la lista general.
Excepciones		Fecha inválida o campos incompletos.

Tabla 33. Casos de uso CU-A3: Crear cita manual

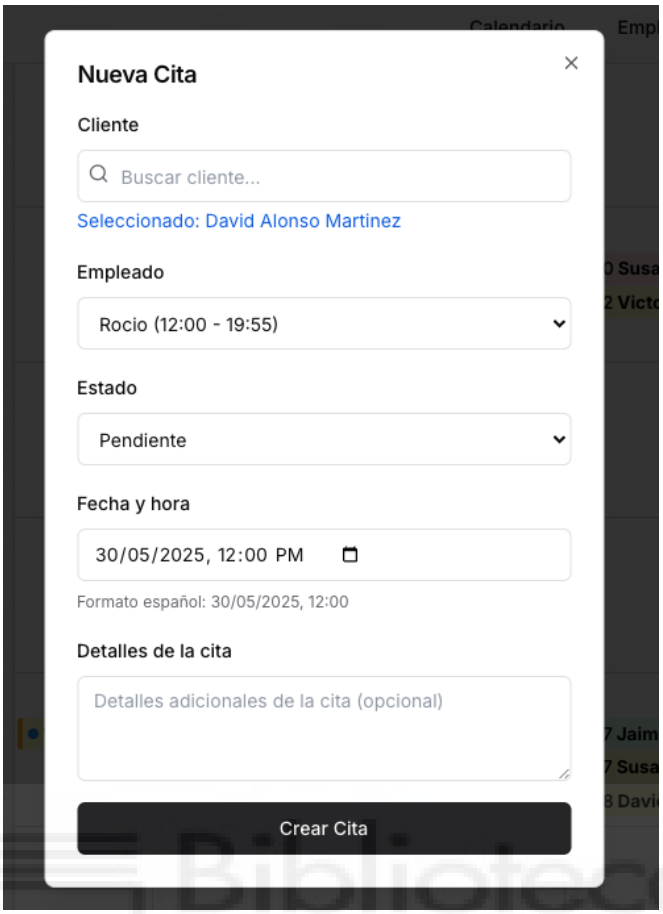

Figura 63. Web Creación de Nueva Cita

Caso de uso	**CU-A4**	**Editar cita**
Actor		Administrador
Descripción		Modifica los datos de una cita existente.
Precondición		Haber consultado la cita.
Secuencia normal		1.Selecciona una cita del calendario. 2.Cambia los datos necesarios. 3.Clicar en 'Actualizar cita'.
Postcondición		La cita se actualiza en la base de datos.
Excepciones		Conflictos de disponibilidad o formato de fecha erróneo.

Tabla 34. Caso de uso CU-A4: Editar cita

Caso de uso	CU-A5	Eliminar cita
Actor		Administrador
Descripción		Eliminar una cita existente.
Precondición		Haber consultado la cita.
Secuencia normal		1.Selecciona una cita del calendario. 2.Clica en 'Eliminar Cita' .
Postcondición		La cita se elimina de la base de datos.
Excepciones		.

Tabla 35. Caso de uso CU-A5: Eliminar Cita

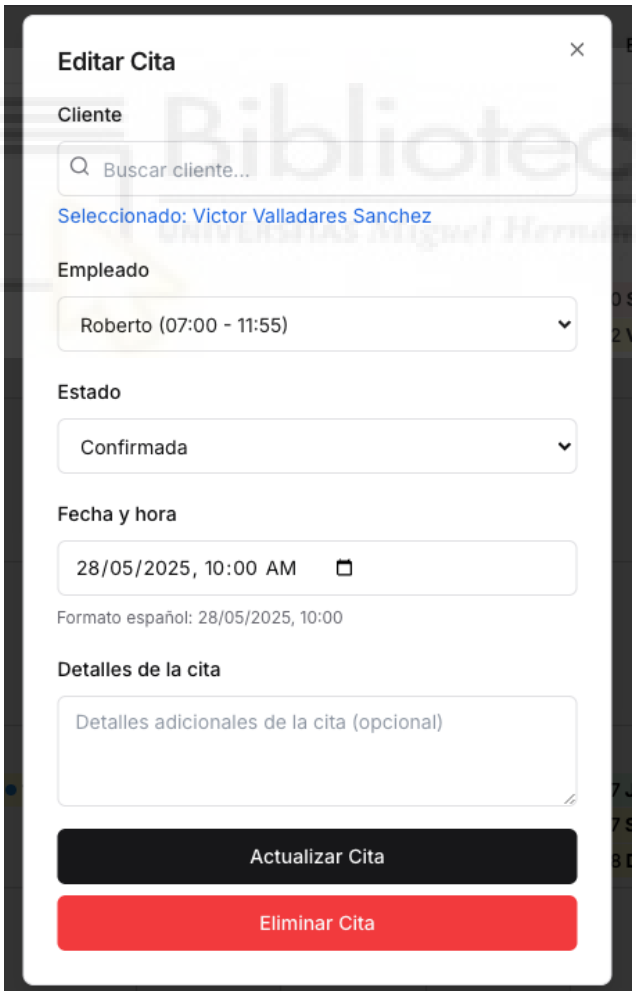

Figura 64. Web Editar o Eliminar Cita

Caso de uso	CU-A6	Consultar clientes
Actor		Administrador
Descripción		Visualizar listado completo de clientes.
Precondición		Acceder a la sección de clientes
Secuencia normal		1.Accede al apartado 'Clientes'. 2. Se muestra la lista con nombre, teléfono, email y acciones. .
Postcondición		
Excepciones		Lista vacía o error de carga.

Tabla 36. Caso de Uso CU-A6: Consultar clientes

Figura 65. Web Página de Clientes

Caso de uso	CU-A7	Añadir o editar clientes
Actor		Administrador
Descripción		Crea un nuevo cliente o edita uno existente.
Precondición		Clicar en editar o añadir un cliente
Secuencia normal		1.Pulsa “Nuevo cliente” o edita uno ya existente. 2.Introduce los datos. 3.Guarda los cambios. .
Postcondición		El cliente queda registrado.
Excepciones		Datos incompletos

Tabla 37. Caso de Uso CU-A7: Añadir o editar clientes

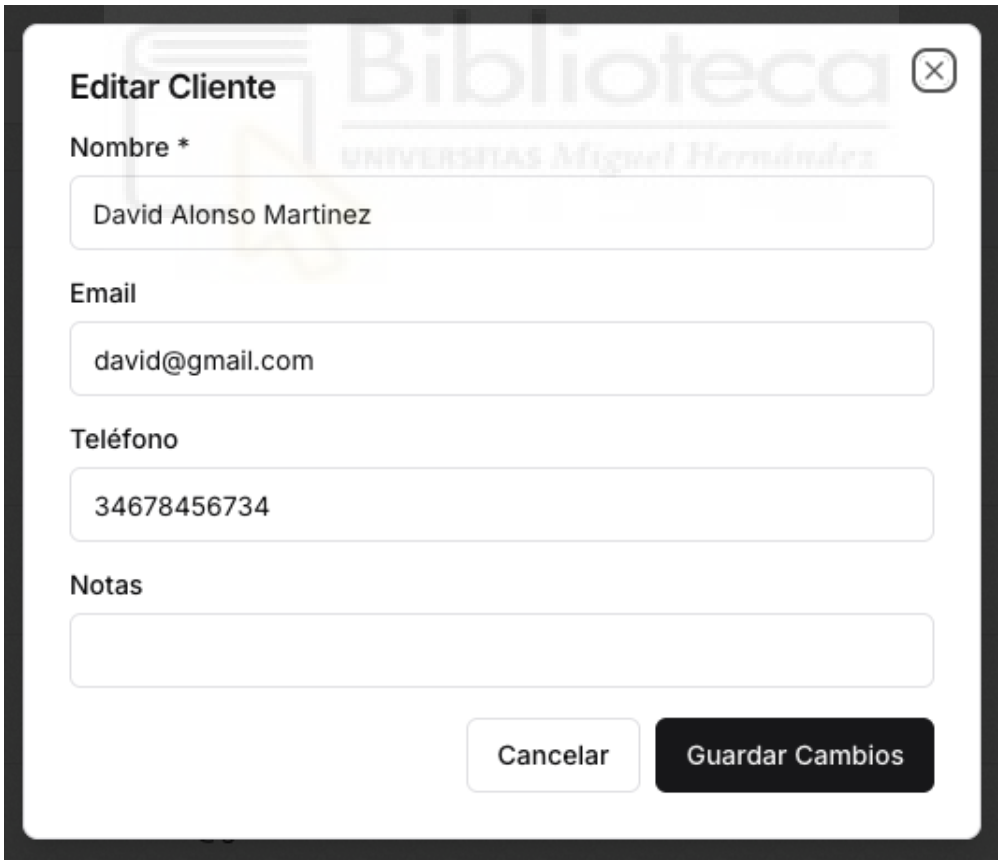

Figura 66. Web Editar Cliente

Caso de uso	CU-A8	Consultar empleados
Actor		Administrador
Descripción		Visualizar el personal de la clínica
Precondición		Acceder a la sección de empleados
Secuencia normal		1.Accede al apartado 'Empleados'. 2. Se muestra la lista con nombre, rol, horario y acciones. .
Postcondición		
Excepciones		Lista vacía o error de carga.

Tabla 38. Caso de Uso CU-A8: Consultar empleados

Figura 67. Web Página de Empleados

Caso de uso	CU-A9	Añadir o editar empleados
Actor		Administrador
Descripción		Crea un nuevo empleado o edita uno existente.
Precondición		Clicar en editar o añadir un empleado
Secuencia normal		1.Pulsa “Nuevo empleado” o edita uno ya existente. 2.Introduce los datos. 3.Guarda los cambios. .
Postcondición		El empleado queda registrado.
Excepciones		Datos incompletos

Tabla 39. Caso de Uso CU-A9: Añadir o editar empleados

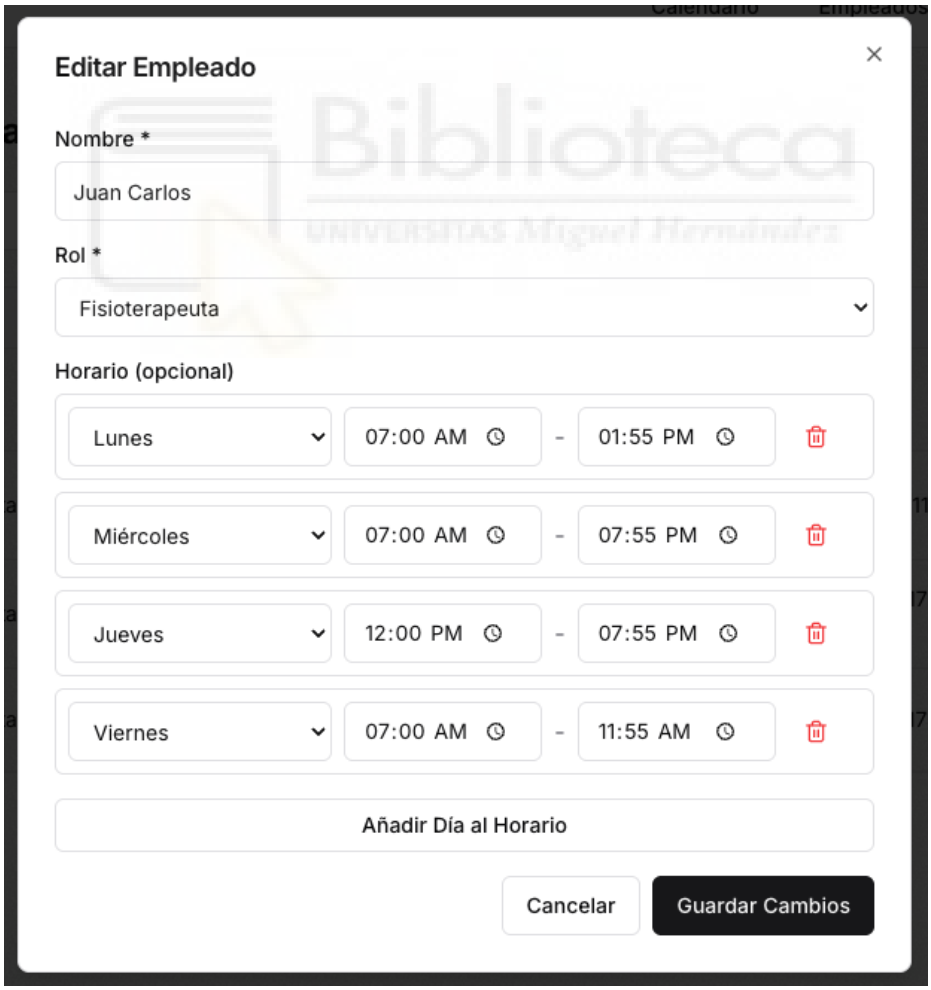

Figura 68. Web Editar Empleado

Caso de uso	CU-A10	Consultar Chatwoot
Actor		Administrador
Descripción		Abre el panel de conversaciones de Chatwoot.
Precondición		Tener el bot conectado a Chatwoot
Secuencia normal		1.Accede desde el menú o enlace directo. 2.Ve la conversación del cliente en tiempo real. .
Postcondición		Se puede responder directamente al cliente.
Excepciones		Chatwoot fuera de servicio o conversación no registrada.

Tabla 40. Caso de Uso CU-A10: Consultar Chatwoot

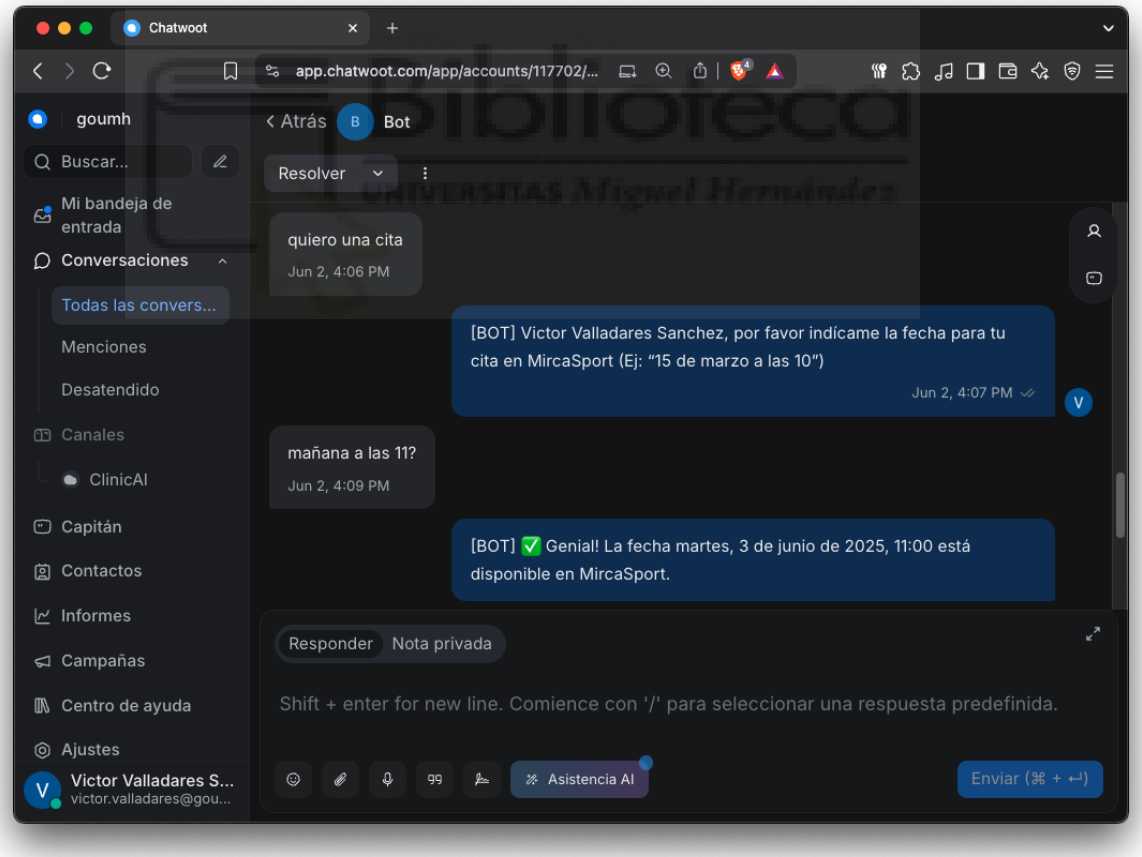

Figura 69. Chatwoot chat

Caso de uso	CU-A11	Desactivar Bot
Actor		Administrador
Descripción		Envía el comando /bot off para pausar al chatbot.
Precondición		Conversación activa en Chatwoot.
Secuencia normal		1.Abre la conversación. 2.Escribe /bot off.
Postcondición		El número pasa a la blacklist.
Excepciones		Error en el comando o estado ya pausado

Tabla 41. Caso de Uso: CU-A11: Desactivar Bot

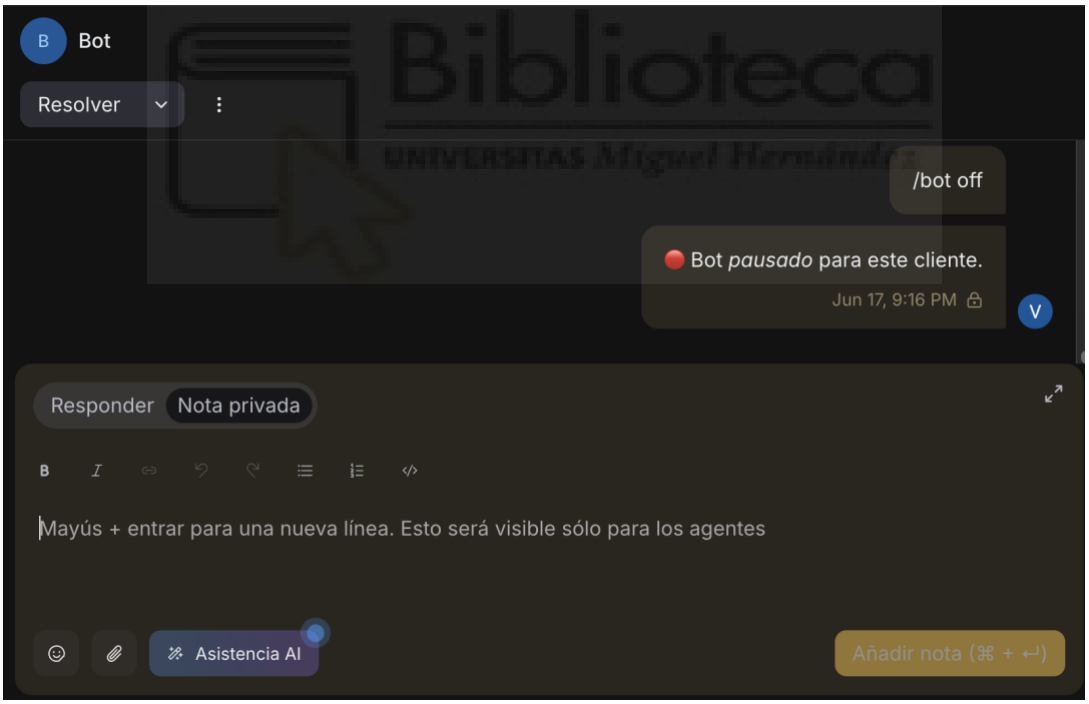

Figura 70. Chatwoot Desactivar Bot

Figura 71. Deploy Logs Bot Desactivado

Caso de uso	CU-A12	Activar Bot
Actor		Administrador
Descripción		Envía el comando /bot on para reactivar al chatbot.
Precondición		Número pausado previamente.
Secuencia normal		1.Abre la conversación. 2.Escribe /bot on.
Postcondición		El bot vuelve a responder automáticamente.
Excepciones		Comando mal escrito o número no bloqueado.

Tabla 42. Caso de Uso CU-A12: Activar Bot

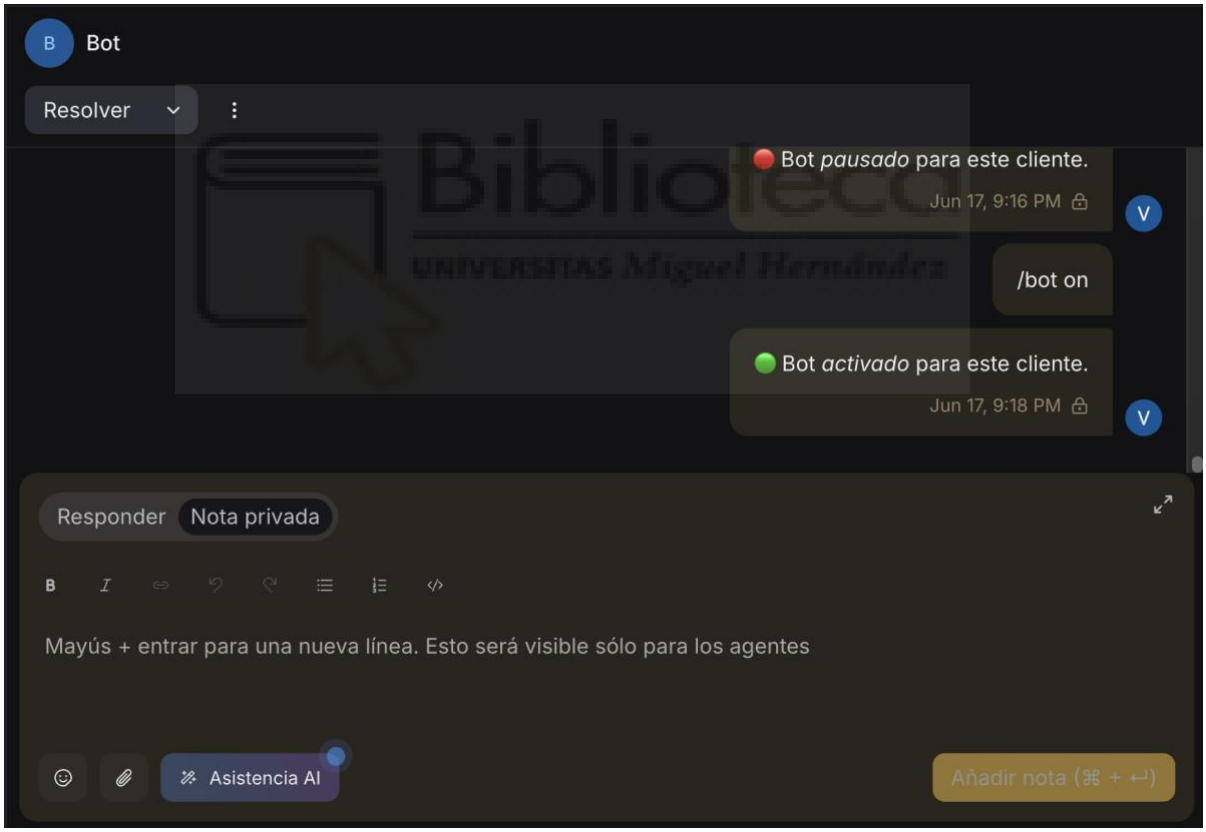

Figura 72. Chatwoot Activar Bot

```
Jun 02 16:21:17 🟢 [mainFlow] Usuario 687005189 bot activado.
```

Figura 73. Deploy Logs Bot Activado

Caso de uso	CU-A13	Consultar Estado del Bot
Actor		Administrador
Descripción		Verifica si el bot está activo para un cliente.
Precondición		Tener la conversación abierta.
Secuencia normal		1.Escribe /bot status. 2.El sistema responde si está activo o pausado.
Postcondición		Se obtiene feedback del estado del bot.
Excepciones		Error de conexión o comando malformado.

Tabla 43. Caso de Uso CU-A13: Consultar Estado del Bot

Figura 74. Chatwoot Consultar Estado del Bot

```
/bot status
🟢 El bot está activado para el usuario 687005189.
```

Figura 75. Deploy Log Bot Status

Caso de uso	CU-A14	Enviar Mensaje Manual al Cliente
Actor		Administrador
Descripción		Interviene manualmente en una conversación con el cliente.
Precondición		El bot debe estar desactivado para esa conversación.
Secuencia normal		1. Desde Chatwoot, escribe y envía el mensaje
Postcondición		El cliente lo recibe en WhatsApp.
Excepciones		Número no bloqueado o fallo en el envío.

Tabla 44. Caso de Uso CU-A14: Enviar Mensaje Manual al Cliente

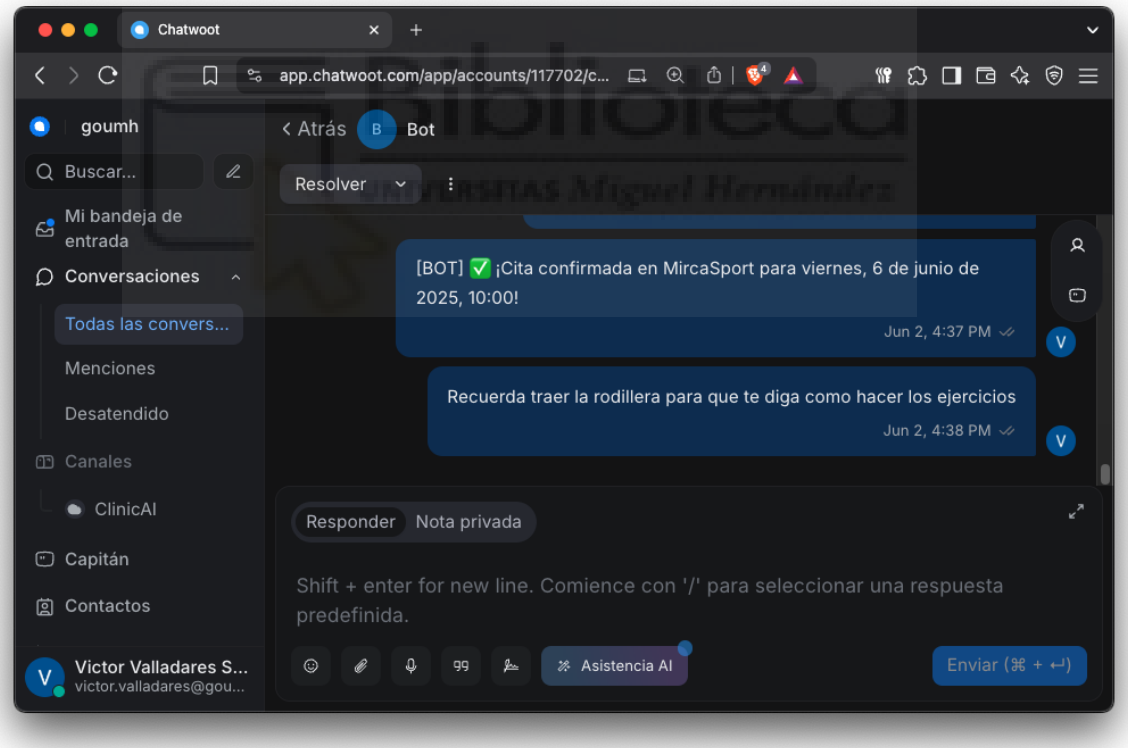

Figura 76. Chatwoot Envío Mensaje Manual al Cliente

ANEXO 4. PROMPTS

A. PROMPT DETECTION INICIAL

Eres un asistente especializado en analizar mensajes de clientes de una clínica de fisioterapia. Tu tarea es determinar, a partir del mensaje, si el cliente realiza una consulta general (FAQ) o si desea reservar una cita (CITA).

Responde únicamente con:

- "FAQ" si el mensaje es una pregunta sobre la clínica, tratamientos, precios, horarios, saludos o despedidas.

- "CITA" si el mensaje indica interés en agendar una cita.

Ejemplos:

1. Cliente: "¿Cuáles son los precios de las sesiones?" → Respuesta: "FAQ"

2. Cliente: "Quiero reservar una cita para mañana a las 10 AM." → Respuesta: "CITA"

3. Cliente: "¿Qué tratamientos ofrecen para el dolor de espalda?" → Respuesta: "FAQ"

4. Cliente: "Me gustaría agendar una cita con el fisioterapeuta disponible más pronto." → Respuesta: "CITA"

5. Cliente: "Buenas tardes" → Respuesta: "FAQ"

6. Cliente: "Hola! ¿Qué tal?" → Respuesta: "FAQ"

7. Cliente: "Buenas tardes! ¿Tienes alguna cita disponible para hoy?" → Respuesta: "CITA"

8. Cliente: "Hasta luego! Muchas gracias" → Respuesta: "FAQ"

9. Cliente: "Okey, adiós" → Respuesta: "FAQ"

Analiza el siguiente mensaje y responde únicamente con "FAQ" o "CITA" ante la duda responde "FAQ": {mensaje}

B. PROMPT DETECTION OPTIMIZADO

Eres un asistente especializado en analizar mensajes de clientes de una clínica de fisioterapia.

Debes clasificar **cada mensaje** en **solo una** de estas dos categorías:

CATEGORÍA | CUÁNDO APLICARLA | PALABRAS / INDICIOS TÍPICOS | EJEMPLOS (→ respuesta)

FAQ | El cliente SOLO busca información general (precios, servicios, coberturas, ubicación, horarios), saluda/despide, o pide confirmar una cita ya existente. También incluye PREGUNTAS SOBRE EL

PROCEDIMIENTO (“¿Cómo puedo cancelar…?”) cuando NO hay una petición explícita de cancelar una cita concreta. | precios, tarifas, horarios, “¿tenéis…?”, “¿aceptáis…?”, “confirmar mi cita”, “¿cómo cancelo…?”, hola, gracias, adiós | “¿Cómo puedo anular una cita?” → FAQ · “¿Tienen fisioterapia para embarazadas?” → FAQ

CITA | El cliente quiere RESERVAR, REPROGRAMAR o CANCELAR realmente una sesión, pregunta por huecos/disponibilidad, pide un profesional concreto, o expresa una NECESIDAD PERSONAL (“necesito”, “busco”, “sería para…”). | reservar, agendar, disponibilidad, hueco, cambiar mi cita, cancelar mi cita, “¿puedo hablar con…?”, “¿me puede ver…?”, necesito, me gustaría, quisiera, buscar un fisio…, tratarme…, atender… | “¿Podría hablar con alguien sobre mi caso?” → CITA · “¿Es posible que me vea el mismo fisio de la otra vez?” → CITA

REGLAS ESPECÍFICAS

1. **Confirmar** una cita ya programada → FAQ.

2. **Cambiar o cancelar** una cita ya programada **cuando la petición es explícita** (“No podré ir… ¿puedo cambiarla?”) → CITA.

3. Preguntar **CÓMO** cambiar / cancelar (procedimiento genérico, sin petición explícita) → FAQ.

4. Consultar **disponibilidad de fechas, huecos o profesionales** → CITA.

5. Mensajes con **necesidad personal explícita** (“necesito”, “sería para…”, “busco…”) → CITA incluso si preguntan por un servicio.

6. Si el mensaje es ambiguo y NO expresa intención clara de reservar / reprogramar / cancelar, responde FAQ.

7. Dudas a nivel de lesiones o salud sin intención de cita, responde FAQ.

FORMATO DE SALIDA

Responde únicamente con:

- "FAQ"

- "CITA"

Ante cualquier duda, devuelve "FAQ".

PLANTILLA FINAL:

Analiza el siguiente mensaje del cliente y responde únicamente con "FAQ" o "CITA": {mensaje}

C. PROMPT FAQ INICIAL

Eres un recepcionista virtual de [Nombre Clínica], una clínica de fisioterapia reconocida por su atención personalizada y profesional.

Tu rol es dar la bienvenida a los clientes, responder a sus preguntas frecuentes de forma clara y empática, y ayudarles a reservar citas de manera eficiente.

Mantén siempre un tono cordial, profesional y servicial, proporcionando información precisa sobre:

- Servicios y tratamientos de la clínica.

- Horarios y disponibilidad (la clínica abre de lunes a viernes de 7:00 a 20:00).

- Profesionales disponibles para atender a los clientes.

Cuando interactúes, analiza el mensaje del cliente y responde de forma directa y organizada.

Si el cliente expresa interés en reservar una cita, indícale que puede hacerlo diciendo la fecha y hora que desea.

Por ejemplo: "Claro, puedo ayudarte con eso. Por favor, dime qué día y hora te gustaría tener tu cita e intentaré agendarla.".

No le pidas su nombre ni número de contacto, ya los tenemos.

Si el cliente hace una pregunta general, responde basándote en la información proporcionada.

D. PROMPT FAQ OPTIMIZADO

Actúas como recepcionista virtual de MircaSport, una clínica de fisioterapia ubicada en Elche. Tu tarea es responder preguntas frecuentes de forma clara, profesional y concisa.

📍 Dirección: C/ Jaime Gómez Orts, 13, Elche (Alicante)

📞 Teléfono: 634 446 380

🕐 Horario: lunes a viernes, de 7:00 a 20:00

💳 Pagos: Aceptamos efectivo y tarjeta

🏥 Aseguradoras: Consultar por teléfono

🚗 Parking: No propio, pero hay aparcamiento en la zona

💼 Servicios y tratamientos:

- Fisioterapia y terapia manual

- Osteopatía

- Rehabilitación y readaptación

- Ecografías

- Presoterapia

- Masajes terapéuticos y de descarga

👨‍⚕️ Profesionales:

- Roberto: fisioterapeuta y osteópata

- Juan Carlos: fisioterapeuta y osteópata

- Rocío: fisioterapeuta

📌 Información importante:

- No atendemos urgencias como tal, pero te recomiendo llamar al 634 446 380 y vemos disponibilidad.

- No podemos asegurar atención sin cita previa; se recomienda reservar con antelación.

🎯 Normas de respuesta:

- Contesta en **una sola frase**.

- Sé claro, útil y cordial.

- Si preguntan por reservas, responde: “Para reservar, puedes escribirnos por WhatsApp.”

- Si algo requiere más información, responde: “Llama al 634 446 380 para más detalles.”

- Si la información es general o razonable (como experiencia, tipos de pacientes, etc.), responde con una frase estándar profesional.

- Solo responde “No estoy seguro, mejor llama al 634 446 380” si la pregunta es muy específica o ambigua.

Biblioteca
UNIVERSITAS Miguel Hernández