Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Talento: Plataforma de servicios freelance

Jesús San Frutos San Lorenzo
Tutores:
MARÍA LUISA MARTÍN PÉREZ
JAVIER DÍAZ FERNÁNDEZ

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que me han acompañado durante este camino. A mi familia, por su apoyo incondicional y por estar siempre ahí en cada paso que he dado, a mis amigos y compañeros de trabajo, por su paciencia, sus palabras de ánimo y por ayudarme a mantener el equilibrio en los momentos de más dificultad. Muy especialmente, a mis tutores María Luisa y Javier, por su cercanía, su cariño constante y por estar siempre disponibles y pendientes de mí con una dedicación que ha marcado la diferencia. También quiero dar las gracias a todos los profesores que he tenido la suerte de tener a lo largo de la carrera, de cada uno de ellos me llevo un aprendizaje valioso que ha contribuido a formar no solo mi conocimiento, sino también mi forma de pensar y crecer profesionalmente.

Resumen

En un contexto donde lo digital gana cada vez más protagonismo, la demanda de profesionales freelance ha crecido de forma notable. Esto ha generado la necesidad de contar con plataformas que conecten de manera eficaz a clientes y trabajadores independientes. Este Trabajo de Fin de Grado tiene como objetivo diseñar y desarrollar una plataforma web orientada al ámbito freelance, utilizando el stack tecnológico MERN (MongoDB, Express.js, React y Node.js).

La plataforma busca ofrecer una solución integral para la gestión de proyectos, permitiendo a los usuarios crear perfiles profesionales, publicar o contratar servicios, comunicarse directamente entre ellos, consultar pedidos y realizar pagos ficticios dentro del entorno. El desarrollo se ha llevado a cabo mediante una metodología en espiral, basada en ciclos iterativos que han facilitado la mejora continua del sistema.

Se ha puesto especial atención en la usabilidad y accesibilidad, asegurando una experiencia de usuario fluida tanto para freelancers como para clientes. Asimismo, se ha trabajado en la escalabilidad y la seguridad del sistema, con el fin de garantizar un crecimiento sostenible y adaptado a futuras necesidades.

Este proyecto no solo presenta una solución técnica funcional, sino que también aborda retos clave como la generación de confianza entre usuarios, la gestión eficaz de pedidos y la flexibilidad para adaptarse a distintos perfiles y servicios. Las conclusiones apuntan a un producto con una base sólida y gran potencial para seguir evolucionando en futuras versiones.

Palabras clave: plataforma web, freelancers, stack MERN, gestión de proyectos, escalabilidad, experiencia de usuario.

Abstract

In a digital-first world, the demand for freelance professionals has increased significantly, highlighting the need for platforms that efficiently connect clients with independent workers. This Final Degree Project aims to design and develop a web-based platform for freelancers using the MERN stack (MongoDB, Express.js, React, and Node.js).

The platform offers a comprehensive solution for freelance project management, allowing users to create professional profiles, publish or hire services, communicate directly, review orders, and simulate payment operations within the system. The development followed an agile methodology, structured in iterative cycles to promote continuous improvement.

A strong focus was placed on user experience and accessibility to ensure smooth navigation for both freelancers and clients. Additionally, the system was designed with scalability and security in mind, enabling future growth and adaptability to evolve user demands.

This project not only delivers a functional technical solution but also tackles key challenges in this type of platform, such as building trust between users, managing orders efficiently, and accommodating a wide range of profiles and services. The conclusions highlight a solid foundation with great potential for future enhancements.

Keywords: web platform, freelancers, MERN stack, project management, scalability, user experience

Capítulo 1
Descripción del proyecto

1.1. Introducción

En las últimas décadas, el trabajo freelance ha emergido como una de las formas más populares y dinámicas de empleo, impulsado por la digitalización global y la búsqueda de modelos laborales más flexibles. Esta transformación ha permitido que millones de profesionales independientes ofrezcan sus servicios en un mercado globalizado, rompiendo las barreras físicas y ampliando sus horizontes. Sin embargo, junto con estas oportunidades surgen nuevos desafíos, como la necesidad de plataformas que faciliten el acceso a trabajos, fomenten la confianza entre usuarios y garanticen una experiencia intuitiva y eficiente tanto para freelancers como para clientes.

En este contexto, el desarrollo de plataformas web dedicadas al sector freelance se ha convertido en una solución esencial para conectar a profesionales y clientes. Estas herramientas no solo actúan como intermediarias, sino que también desempeñan un papel crucial en la creación de un ecosistema laboral organizado, accesible y equitativo. A pesar de los avances en este ámbito, muchas plataformas actuales carecen de una integración completa de funciones que permita una experiencia realmente fluida, o bien imponen barreras económicas y tecnológicas que dificultan su adopción por parte de pequeños emprendedores y nuevos usuarios.

Además, el auge del trabajo independiente ha traído consigo una creciente demanda por soluciones que garanticen transacciones seguras y confiables, junto con herramientas que permitan gestionar de manera efectiva los proyectos y las relaciones entre freelancers y clientes. Esta necesidad no solo refleja la evolución del mercado laboral, sino también un cambio en las expectativas de los usuarios, que buscan plataformas adaptadas a sus necesidades específicas y con una usabilidad impecable.

Este Trabajo de Fin de Grado se enmarca en el objetivo de explorar y proponer una solución integral a estas limitaciones, desarrollando una plataforma web que combina accesibilidad, transparencia y funcionalidad en un entorno amigable para todos los usuarios. Más allá de ofrecer un medio para la compra y venta de servicios, este proyecto busca representar una contribución al ecosistema laboral freelance, respondiendo a necesidades concretas identificadas en el mercado.

El desarrollo de esta plataforma no solo refleja la aplicación de conocimientos adquiridos en el ámbito de la ingeniería informática, sino que también destaca la importancia de comprender las dinámicas sociales y económicas que rodean al trabajo freelance en el contexto actual. Asimismo, esta iniciativa pone de manifiesto cómo la tecnología puede actuar como un puente entre profesionales y clientes, fortaleciendo las relaciones laborales y promoviendo un modelo de trabajo sostenible y eficiente. A lo largo de este documento, se explorarán los fundamentos, el diseño y los retos superados en el desarrollo de este proyecto, ofreciendo una visión detallada de su impacto potencial en el ámbito freelance.

1.2. Objetivos

El presente Trabajo de Fin de Grado tiene como propósito central el diseño y desarrollo de una plataforma web orientada al trabajo freelance. Esta solución busca no solo facilitar la interacción entre freelancers y clientes, sino también proporcionar un entorno digital que fomente la confianza, la accesibilidad y la eficiencia en la gestión de proyectos.

La plataforma aspira a satisfacer las necesidades de un mercado en constante evolución, ofreciendo una experiencia optimizada para ambas partes. Para ello, se plantean los siguientes objetivos:

- Diseñar un sistema que permita a los usuarios gestionar sus cuentas y perfiles de forma flexible y adaptada a diferentes roles.
- Proveer un entorno funcional para la publicación, búsqueda y contratación de servicios.
- Implementar mecanismos que favorezcan la comunicación y el intercambio de información entre las partes involucradas.
- Asegurar un flujo de trabajo estructurado que abarque desde la contratación del servicio hasta la entrega del trabajo.
- Incorporar funcionalidades que respalden la seguridad y confianza en las transacciones realizadas dentro de la plataforma.

1.3. Estado del arte

En este apartado se explora el panorama actual de herramientas y plataformas digitales relacionadas con el proyecto, presentando la solución propuesta y analizando el entorno en el que se desarrollará. Esto permite identificar las características distintivas del proyecto en comparación con las alternativas disponibles, subrayando su innovación y aportación en el contexto tecnológico actual.

1.3.1 Solución propuesta

La propuesta de este proyecto se basa en el desarrollo de una plataforma web destinada a conectar freelancers con clientes, mediante un enfoque integral que combine simplicidad, eficiencia y funcionalidad avanzada. Esta solución busca satisfacer las demandas crecientes de un mercado globalizado y digitalizado, donde la flexibilidad y la accesibilidad son esenciales para facilitar la colaboración entre profesionales autónomos y contratantes.

A diferencia de otras plataformas existentes, esta propuesta pone un énfasis particular en:

- Experiencia de usuario optimizada: Diseñada para ser intuitiva y accesible, asegurando que tanto usuarios experimentados como principiantes puedan navegar y aprovechar al máximo sus funcionalidades.
- Equilibrio entre proyectos cortos y largos: Incluye herramientas específicas para gestionar tanto tareas rápidas como colaboraciones de largo plazo, un aspecto

poco atendido por algunas alternativas.
- Interacción dinámica: Se integra un sistema de mensajería que fomenta la comunicación fluida y directa entre clientes y freelancers, promoviendo la transparencia y la confianza mutua.

1.3.2 Entorno de la aplicación

El entorno competitivo de esta solución incluye plataformas reconocidas que dominan el mercado freelance, como Fiverr y Upwork. Estas herramientas han establecido estándares en términos de funcionalidad y alcance, pero también presentan limitaciones que este proyecto pretende superar.

1.3.2.1 Fiverr

Figura 1. Fiverr

Fiverr es una plataforma conocida por su simplicidad y rapidez (Figura 2). Permite a los freelancers ofrecer sus servicios en paquetes definidos y a los clientes contratarlos con facilidad. Algunas de sus características principales son:

- Simplicidad en la publicación de servicios: Ideal para tareas específicas y de corta duración.
- Acceso rápido a recursos: Los clientes pueden buscar servicios utilizando filtros avanzados.
- Modelo transaccional ágil: Promueve interacciones rápidas y concretas.

Sin embargo, su estructura estandarizada puede limitar las opciones para proyectos que requieren flexibilidad o adaptaciones personalizadas.

1.3.2.2 Upwork

Figura 3. Upwork

Upwork, por otro lado, es una plataforma orientada hacia proyectos de mayor complejidad (Figura 4). Entre sus características destacan:
- Gestión integral del trabajo: Incluye herramientas para el seguimiento del tiempo, generación de facturas y gestión de proyectos.

- Variedad de propuestas: Los freelancers pueden postularse a proyectos con presupuestos y condiciones negociables.
- Segmentación avanzada: Permite categorizar proyectos y servicios para mayor precisión en la búsqueda.

No obstante, su curva de aprendizaje puede ser un desafío, especialmente para nuevos usuarios que buscan una experiencia más accesible.

1.3.2.3 Comparativa

La siguiente tabla destaca las diferencias clave entre la solución propuesta y las plataformas mencionadas:

Característica	Talento	Fiverr	Upwork
Adecuación a proyectos largos	Sí	Limitada	Sí
Interfaz amigable	Muy alta	Alta	Media
Claridad en precios y tiempos	Alta	Alta	Media
Sistema de colaboración integrado	Sí	No	Sí
Accesibilidad para principiantes	Muy alta	Alta	Media

Tabla 1. Comparativa del entorno de la aplicación

Talento se posiciona como una plataforma equilibrada, combinando la simplicidad y accesibilidad de Fiverr con las herramientas de gestión avanzada de Upwork. Además, añade funcionalidades distintivas, como un sistema de interacción más dinámico y un diseño que favorece la adaptación a distintos tipos de proyectos, convirtiéndola en una propuesta más inclusiva y flexible.

1.4. Alcance

El apartado de alcance define las funcionalidades y limitaciones que tendrá la solución propuesta, proporcionando una visión clara y estructurada de sus capacidades. Para ello, se representa un Árbol de características, que detalla las principales funcionalidades de la plataforma, agrupadas en diferentes niveles jerárquicos (Figura 5):

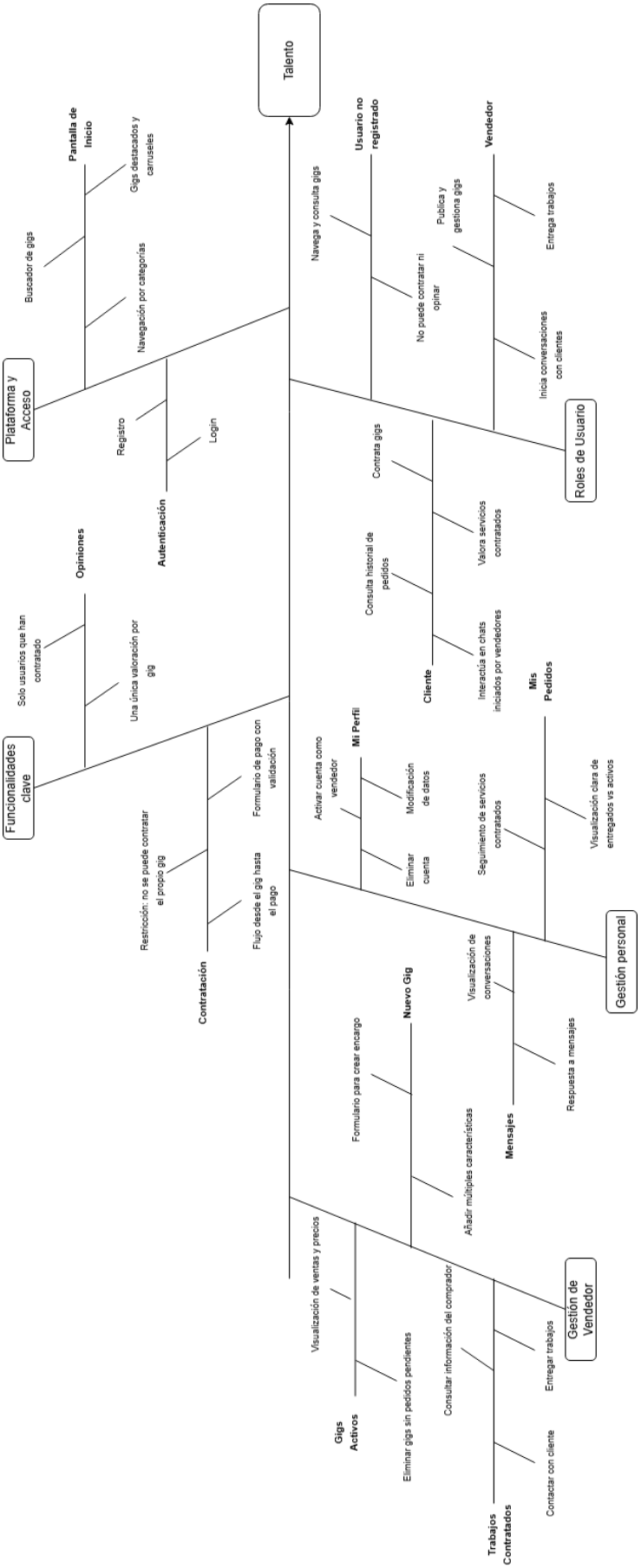


Figura 6. Diagrama de Árbol de características

1.5. Entorno y tecnologías utilizadas

El desarrollo de la plataforma Talento se basa en una arquitectura claramente diferenciada entre el backend y el frontend, siguiendo el modelo cliente-servidor, donde el backend opera como una API REST. Esta separación no solo facilita la gestión independiente de las funcionalidades del servidor y la interfaz de usuario, sino que también proporciona una mayor flexibilidad y escalabilidad al sistema, permitiendo futuras mejoras o ampliaciones sin afectar a toda la aplicación.

Las APIs REST (Representational State Transfer) son un estándar ampliamente adoptado para habilitar la comunicación entre sistemas mediante peticiones HTTP. En este proyecto, esta arquitectura asegura un intercambio eficiente, estructurado y seguro de datos entre el cliente y el servidor, optimizando tanto la velocidad como la fiabilidad en la transferencia de información. Esta implementación permite que el backend funcione como el núcleo de la lógica empresarial, mientras que el frontend se encarga de presentar una experiencia de usuario atractiva y dinámica.

1.5.1 Backend

El backend de la plataforma Talento fue diseñado para garantizar escalabilidad, seguridad y eficiencia, utilizando un conjunto sólido de tecnologías y herramientas. La combinación de Node.js, Express.js, y una base de datos NoSQL como MongoDB permitió crear un sistema adaptable y robusto, con un rendimiento óptimo para manejar solicitudes de usuarios y la gestión de datos.

1.5.1.1 Node.js

Figura 7. Logo Node.js

Node.js es un entorno de ejecución de JavaScript que permite ejecutar código del lado del servidor (Figura 8). A diferencia de los lenguajes tradicionales del backend, Node.js se basa en un modelo de eventos no bloqueante, lo que significa que puede manejar múltiples solicitudes de manera simultánea sin que el procesamiento de una solicitud afecte a las demás.

Ventajas clave:

- Alto rendimiento: Su arquitectura basada en el bucle de eventos (event loop) permite un manejo eficiente de aplicaciones en tiempo real.

- Escalabilidad: Ideal para construir aplicaciones con grandes volúmenes de tráfico, como APIs RESTful y sistemas en tiempo real.
- Ecosistema de paquetes: La plataforma cuenta con una extensa biblioteca de módulos en npm (Node Package Manager), lo que facilita la incorporación de funcionalidades avanzadas al proyecto.
- Aplicación en Talento: Node.js se utilizó como núcleo del backend, garantizando que el servidor pueda responder a solicitudes simultáneas con baja latencia.

1.5.1.2 Express.js

Figura 9. Logo Express js

Express.js es un framework web minimalista y flexible para Node.js (Figura 10). Proporciona herramientas esenciales para desarrollar aplicaciones del lado del servidor, como la gestión de rutas, middleware y soporte para solicitudes HTTP complejas.

Ventajas clave:

- Estructuración del código: Simplifica la organización del backend en rutas, controladores y middleware, lo que facilita el mantenimiento y escalabilidad del proyecto.
- Extensibilidad: Ofrece un sistema de middleware que permite extender las funcionalidades del servidor con facilidad.
- Compatibilidad: Totalmente compatible con Node.js y las bibliotecas disponibles en npm y yarn.
- Aplicación en Talento: Express.js permitió estructurar el backend de forma modular, definiendo rutas específicas para diferentes funcionalidades, como autenticación, gestión de usuarios y operaciones relacionadas con datos.

1.5.1.3 MongoDB

Figura 11. Logo MongoDB

MongoDB es una base de datos NoSQL orientada a documentos (Figura 12). Almacena datos en formato JSON (BSON internamente), lo que facilita la interacción con aplicaciones JavaScript como Node.js.

Ventajas clave:

- Flexibilidad: Permite almacenar datos sin esquemas rígidos, lo que resulta ideal para proyectos donde las estructuras de datos pueden cambiar con el tiempo.
- Escalabilidad horizontal: Compatible con sistemas distribuidos, lo que la hace apta para gestionar grandes volúmenes de datos.
- Integración con JavaScript: Su formato JSON facilita el intercambio de datos entre el backend y la base de datos, reduciendo la necesidad de transformaciones adicionales.
- Aplicación en Talento: MongoDB se utilizó para almacenar datos relacionados con los usuarios, como información personal, credenciales encriptadas y registros de actividad, garantizando un acceso rápido y eficiente.

1.5.1.4 Mongoose

Figura 13. Logo Mongoose

Mongoose es una biblioteca ODM (Object Data Modeling) para MongoDB que permite interactuar con la base de datos utilizando modelos y esquemas bien definidos (Figura 14).

Ventajas clave:

- Validación de datos: Permite definir reglas estrictas para los datos almacenados, reduciendo errores relacionados con entradas incorrectas.
- Facilidad de uso: Proporciona métodos simplificados para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) en la base de datos.
- Middleware: Permite ejecutar funciones adicionales antes o después de las operaciones en la base de datos, como validaciones personalizadas o cifrado de datos sensibles.
- Aplicación en Talento: Mongoose se utilizó para modelar datos clave, como los usuarios y sus roles, definiendo esquemas con validaciones personalizadas para asegurar la integridad de los datos.

1.5.1.5 JSON Web Token (JWT)

Figura 15. Logo JSON Web Token

JWT es un estándar para la autenticación basada en tokens (Figura 16). Cada token contiene información en formato JSON, que puede ser verificada por el servidor sin necesidad de almacenar sesiones.

Ventajas clave:

- Autenticación sin estado: Los tokens se pueden verificar sin necesidad de mantener un almacenamiento centralizado de sesiones en el servidor.
- Seguridad: Incluye mecanismos para proteger los tokens contra alteraciones mediante algoritmos de firma como HMAC o RSA.
- Interoperabilidad: Se puede usar en diferentes plataformas y lenguajes de programación, lo que permite ampliar el sistema fácilmente.
- Aplicación en Talento: JWT se implementó para proteger rutas sensibles, asegurando que solo los usuarios autenticados pudieran acceder a ellas.

1.5.1.6 bcryptjs

Figura 17. Logo bcryptjs

La librería bcryptjs de JavaScript está diseñada para encriptar contraseñas (Figura 18). Utiliza un algoritmo de hashing seguro que incluye un "salt" para proteger las credenciales de los usuarios contra ataques de fuerza bruta.

Ventajas clave:

- Seguridad avanzada: Genera hashes únicos incluso para contraseñas idénticas, gracias al uso del salt.
- Facilidad de integración: Compatible con Node.js y otras herramientas del ecosistema backend.

- Opciones configurables: Permite ajustar el costo computacional del proceso de hashing para equilibrar entre seguridad y rendimiento.
- Aplicación en Talento: bcryptjs se utilizó para encriptar las contraseñas de los usuarios antes de almacenarlas en la base de datos, asegurando que incluso si la base de datos es comprometida, las contraseñas permanezcan protegidas.

1.5.1.7 Dotenv

Figura 19. Logo Dotenv

Dotenv es una herramienta para cargar variables de entorno desde un archivo .env en el proceso de ejecución de Node.js (Figura 20). Esto permite gestionar información sensible de forma segura y separada del código fuente.

Ventajas clave:

- Seguridad: Las claves API, URLs de conexión y otros parámetros sensibles se mantienen fuera del código fuente, reduciendo el riesgo de exposición.
- Portabilidad: Facilita la configuración del entorno en diferentes entornos de desarrollo, prueba y producción.
- Simplicidad: Es fácil de integrar y requiere una configuración mínima.
- Aplicación en Talento: Dotenv se utilizó para gestionar parámetros como la URL de conexión a MongoDB, las claves secretas para JWT.

1.5.2 Frontend

El frontend de la plataforma Talento fue diseñado utilizando herramientas modernas que garantizan una experiencia de usuario interactiva, consistente y visualmente atractiva. La arquitectura basada en componentes de React.js, combinada con la potencia de Material-UI (MUI), permitió construir una interfaz modular, reutilizable y fácil de mantener.

1.5.2.1 React.js

Figura 21. Logo React.js

React.js es una biblioteca de JavaScript ampliamente utilizada para construir interfaces de usuario (Figura 22). Su enfoque declarativo simplifica el desarrollo y mantenimiento de aplicaciones dinámicas al dividirlas en componentes reutilizables.

Ventajas clave:

- Arquitectura basada en componentes: Promueve la modularidad, permitiendo reutilizar y probar componentes de forma independiente.
- Actualizaciones eficientes: Utiliza un DOM virtual que optimiza las actualizaciones de la interfaz, asegurando un rendimiento rápido incluso en aplicaciones complejas.
- Ecosistema amplio: Compatible con una gran variedad de librerías y herramientas para manejar estado, rutas y comunicación con APIs.
- Aplicación en Talento: React.js permitió construir una interfaz de usuario dinámica y altamente reactiva. Por ejemplo, los componentes reutilizables se emplearon para diseñar elementos comunes como la barra de navegación, carruseles, formularios y tarjetas de para los gigs.

1.5.2.2 Material-UI (MUI)

Figura 23. Logo Material-UI (MUI)

Material-UI es una biblioteca de componentes de interfaz de usuario preconstruidos basada en las pautas de diseño Material Design de Google (Figura 24). Proporciona herramientas listas para usar que garantizan una experiencia visual consistente y atractiva.

Ventajas clave:

- Componentes preconstruidos: Incluye botones, barras de navegación, tablas y otros elementos listos para usar.
- Adaptabilidad: Ofrece un sistema de diseño responsivo que asegura una visualización óptima en dispositivos móviles, tabletas y escritorios.
- Personalización: Permite modificar temas, colores y estilos para adaptarse a los requerimientos específicos del proyecto.
- Accesibilidad integrada: Asegura que los componentes cumplan con estándares de accesibilidad, mejorando la experiencia para todos los usuarios.
- Aplicación en Talento: Material-UI se utilizó para garantizar un diseño profesional y atractivo. Se personalizaron componentes como alertas e iconos para reflejar la identidad visual de la plataforma.

1.5.2.3 Axios

Figura 25. Logo Axios

Axios es una librería de JavaScript para realizar solicitudes HTTP desde el navegador o Node.js (Figura 26). Simplifica la comunicación con APIs y es compatible con promesas para manejar operaciones asíncronas.

Ventajas clave:

- Configuración flexible: Soporta configuraciones avanzadas como encabezados personalizados, autenticación y manejo de tiempo de espera.
- Interceptores: Permite modificar solicitudes y respuestas antes de enviarlas o recibirlas, útil para gestionar tokens de autenticación.
- Compatibilidad con navegadores: Funciona de manera consistente en todos los navegadores modernos.
- Aplicación en Talento: Axios se utilizó para conectar el frontend con la API del backend. Por ejemplo, se gestionaron operaciones como el inicio de sesión de usuarios, el registro de datos y la obtención de información dinámica para mostrar en la interfaz.

1.5.2.4 React Router

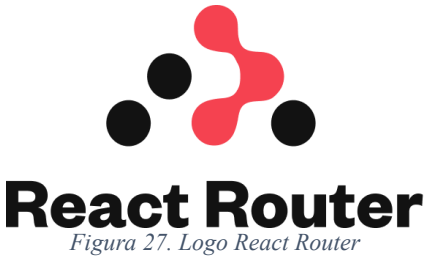


Figura 27. Logo React Router

React Router es una librería que facilita la navegación entre diferentes vistas de una aplicación React (Figura 28). Utiliza un enrutamiento basado en componentes, lo que permite definir rutas dinámicas y manejar la navegación sin recargar la página.

Ventajas clave:

- Navegación fluida: Garantiza una experiencia de usuario continua al evitar recargas completas de la página.
- Rutas protegidas: Soporta la creación de rutas restringidas que requieren autenticación.
- Rutas dinámicas: Permite manejar parámetros en las URL, útiles para vistas específicas como perfiles de usuarios o detalles de productos.
- Aplicación en Talento: React Router permitió estructurar la navegación de la plataforma, asegurando una transición fluida entre secciones como la página de inicio, el perfil del usuario y las herramientas de gestion.

1.5.3 Sistema de control de versiones

El sistema de control de versiones es una parte fundamental del desarrollo de cualquier proyecto. En Talento, se utilizó Git como herramienta de control de versiones y GitHub como plataforma de alojamiento y gestión del repositorio. Este enfoque garantizó un desarrollo organizado, con historial detallado de cambios y flujos de trabajo bien estructurados.

1.5.3.1 Git

Figura 29. Logo Git

Git es un sistema de control de versiones distribuido que permite a los desarrolladores trabajar en paralelo, registrar cambios en el código y colaborar de manera eficiente en equipos (Figura 30).

Ventajas clave:

- Historial detallado: Cada cambio realizado en el proyecto queda registrado, permitiendo revertir errores o restaurar versiones anteriores.
- Flujos de trabajo paralelos: Las ramas facilitan la implementación de nuevas funcionalidades sin interferir con el código principal.
- Gestión de conflictos: Proporciona herramientas para resolver conflictos cuando diferentes desarrolladores realizan cambios en el mismo archivo.
- Aplicación en Talento: Git se utilizó para gestionar versiones del proyecto, permitiéndome trabajar en nuevas características y corregir errores.

1.5.3.2 GitHub

Figura 31. Logo GitHub

GitHub es una plataforma basada en la nube que facilita la gestión de repositorios de Git y la colaboración en equipo (Figura 32).

Ventajas clave:

- Repositorio remoto: Proporciona almacenamiento seguro del código, accesible desde cualquier lugar.
- Revisión de cambios: Las solicitudes de extracción (pull requests) permiten revisar el código antes de integrarlo en el proyecto principal.
- Automatización: GitHub Actions permite configurar flujos de trabajo automatizados, como la ejecución de pruebas o despliegues continuos.
- Colaboración estructurada: Proporciona herramientas como issues y tableros de proyectos para organizar tareas y seguimiento.
- Aplicación en Talento: GitHub fue utilizada complementando a git para alojar y gestionar el repositorio del proyecto.

1.6. Estructura de la memoria

En este apartado, se describe la estructura que sigue la memoria del proyecto Talento, detallando cómo se organizará la información y el contenido de cada uno de los capítulos que la componen. La memoria se presenta de forma ordenada, facilitando la comprensión del proceso seguido durante el desarrollo del proyecto, desde la fase de planificación hasta la implementación y pruebas finales.

- **Capítulo 1: Descripción del proyecto**

  En este capítulo se presenta una introducción detallada al proyecto Talento, incluyendo su propósito y objetivos, así como el análisis del estado del arte en sistemas de gestión de servicios freelance. Se describen las soluciones propuestas en cuanto a funcionalidad y diseño, y se establece el entorno de la aplicación en el que se desarrolla la solución, detallando las características principales que conforman el sistema. Además, se realiza una descripción de las tecnologías utilizadas tanto en el frontend como en el backend del sistema, explicando de forma breve cómo cada herramienta contribuye al funcionamiento de la plataforma.

- **Capítulo 2: Planificación**

  En este capítulo se expone la metodología de trabajo adoptada para llevar a cabo el proyecto, incluyendo la planificación temporal inicial, los objetivos de cada fase, así como los recursos y tiempos asignados. Se realiza un análisis de los riesgos potenciales a lo largo del desarrollo del proyecto, presentando estrategias para mitigar posibles inconvenientes. Además, se ofrece una estimación del presupuesto necesario para completar el desarrollo, y se documenta el seguimiento real de la planificación, con un análisis comparativo entre lo estimado y lo ejecutado.

- **Capítulo 3: Análisis**

  Este capítulo contiene una descripción detallada de los requisitos de usuario y del sistema, abordando los casos de uso que guían el comportamiento y las interacciones del sistema. Se especifican los actores involucrados, las tareas que

deben realizar y los flujos de trabajo esperados. A través del diagrama de casos de uso, se visualiza cómo los usuarios interactúan con la plataforma. Además, se detallan los requisitos funcionales (lo que debe hacer el sistema) y no funcionales (rendimiento, usabilidad, seguridad, entre otros), con el objetivo de garantizar que el sistema cumpla con los estándares establecidos.

- **Capítulo 4: Diseño**

  En este capítulo se presenta la arquitectura lógica y física de la plataforma Talento, explicando cómo se organiza la aplicación en términos de componentes y capas. Se incluyen diagramas de secuencia y de actividad para ilustrar los flujos de interacción entre los usuarios y el sistema, así como el proceso de ejecución de diversas tareas clave. Además, se ofrece un modelo de datos detallado, con su correspondiente modelo entidad-relación, diccionario de datos y diseño lógico de la base de datos. Este capítulo también aborda el diseño de las interfaces de usuario, asegurando que sean accesibles, fáciles de usar y coherentes con la experiencia general de la plataforma.

- **Capítulo 5: Implementación**

  Aquí se describen los detalles de la implementación de Talento, tanto en el backend como en el frontend, haciendo énfasis en la estructura del proyecto, los componentes involucrados y cómo se integran para formar una solución funcional. Además, se abordan cuestiones relevantes de la implementación y se describen las decisiones tomadas en cuanto a la organización del código y las herramientas utilizadas.

- **Capítulo 6: Pruebas**

  En este capítulo se detallan las pruebas realizadas para garantizar que la plataforma funcione correctamente y sin errores. Se incluyen pruebas de caja blanca, que aseguran que el código y las funciones internas del sistema estén correctamente implementadas, y pruebas de caja negra, que validan el comportamiento del sistema desde la perspectiva del usuario final. Se presentan los resultados obtenidos y las posibles correcciones o mejoras realizadas a raíz de las pruebas.

- **Capítulo 7: Manuales**

  En este capítulo se incluyen los manuales de uso y despliegue de la aplicación. Estos manuales ofrecen instrucciones detalladas sobre cómo utilizar la plataforma, cómo instalarla en un entorno de producción y cómo administrar el sistema. Se presenta:
  - Manual de despliegue: Guía completa para la instalación y configuración de Talento en una maquina en local.
  - Manual de Usuario: Instrucciones y tutoriales para los usuarios finales que interactuarán con la plataforma, explicando sus funcionalidades y cómo aprovecharlas.

- **Capítulo 8: Conclusiones**

  El capítulo final reflexiona sobre el desarrollo del proyecto, los logros alcanzados y las dificultades superadas. Se evalúa el cumplimiento de los objetivos iniciales y la efectividad de las soluciones implementadas. También se plantean posibles líneas de trabajo futuras, sugiriendo mejoras y nuevas funcionalidades para seguir ampliando y perfeccionando Talento a medida que surjan nuevas necesidades o tecnologías.

- **Webgrafía**

  Esta sección incluye todas las fuentes de información y recursos en línea utilizados durante el desarrollo del proyecto. Se citan los documentos técnicos, guías de referencia, tutoriales y cualquier otro recurso que haya influido en el proceso de creación de Talento.

Capítulo 2
Planificación del proyecto

2.1. Metodología de trabajo

Para el desarrollo de Talento, se ha optado por el modelo en espiral de Boehm. Esta metodología combina elementos del desarrollo iterativo y el enfoque sistemático del modelo en cascada, lo que permite obtener un producto final robusto, flexible y adaptado a las necesidades del usuario. La decisión de aplicar este modelo se basa en la naturaleza del proyecto, que consiste en una plataforma digital para la gestión y contratación de servicios freelance. Dado que Talento requiere un desarrollo progresivo, donde funcionalidades clave como la creación de gigs, el sistema de mensajería y la gestión de pedidos necesitan validación y refinamiento continuos, el modelo en espiral se adapta perfectamente al planteamiento del proyecto.

El modelo en espiral (Figura 33) se estructura en ciclos iterativos, cada uno de los cuales permite una mejora incremental del sistema. Cada ciclo abarca varias fases que se repiten hasta alcanzar un producto final satisfactorio. Esta metodología no solo facilita la incorporación de cambios en cualquier momento del desarrollo, sino que también pone un fuerte énfasis en la gestión de riesgos, lo que resulta crucial para un proyecto como Talento, donde la experiencia del usuario y la integración fluida de funcionalidades son prioritarias.

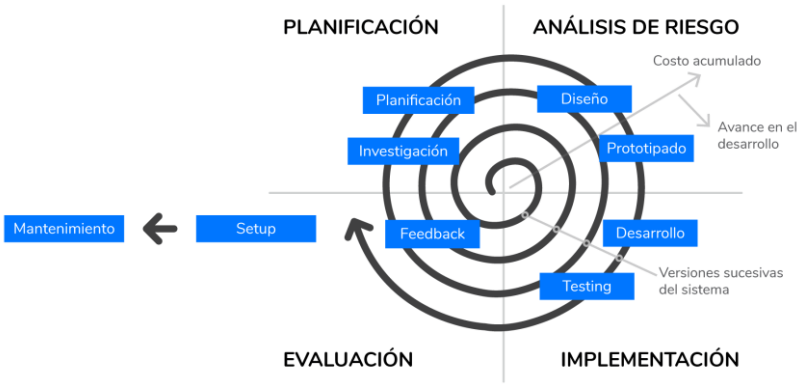


Figura 34. Modelo en Espiral

Las fases del modelo en espiral son fundamentales para entender su valor en este proyecto:

- Determinación de objetivos: En esta fase se definen los objetivos específicos que se quieren lograr en cada ciclo, junto con las alternativas para alcanzarlos. En el caso de Talento, esto ha implicado establecer metas claras para cada iteración, como el desarrollo del sistema de registro de usuarios, la creación de gigs o la funcionalidad de contratación y pago.
- Identificación y análisis de riesgos: Una de las grandes ventajas del modelo en espiral es su enfoque en la identificación temprana de riesgos. Aquí se analizan

los posibles problemas que podrían surgir, desde fallos técnicos hasta aspectos relacionados con la experiencia de usuario.

- Desarrollo y validación: En esta fase se lleva a cabo la implementación y prueba del producto en su versión actual. Cada incremento se desarrolla de forma controlada y se válida para garantizar su calidad. En el desarrollo de Talento, esto se tradujo en la creación y prueba de módulos clave, como el sistema de mensajería entre cliente y vendedor, la visualización de perfiles y la gestión de pedidos y entregas.
- Evaluación del cliente y planificación del siguiente ciclo: Al finalizar cada ciclo, se revisa el trabajo realizado con el fin de obtener retroalimentación. Este proceso es esencial para planificar el siguiente ciclo, introduciendo mejoras y corrigiendo desviaciones. En Talento, esta evaluación permitió perfeccionar la navegación, optimizar la usabilidad del sistema y ajustar funcionalidades críticas para una experiencia más fluida.

El modelo en espiral de Boehm ha permitido abordar el desarrollo del proyecto de forma flexible y adaptable, integrando cambios cuando ha sido necesario y garantizando que cada funcionalidad se implementara de manera coherente y eficiente. Su enfoque iterativo y centrado en la gestión de riesgos ha facilitado una planificación sólida y un producto final alineado con los objetivos iniciales del proyecto.

2.2. Planificación del proyecto

La planificación del proyecto Talento es fundamental para lograr un desarrollo organizado y eficiente, alineado con los objetivos del Trabajo Fin de Grado. En esta sección, se detallará la planificación inicial y la distribución del tiempo, con el fin de que cada fase del desarrollo se lleve a cabo de manera progresiva y con los recursos necesarios.

2.2.1 Planificación temporal inicial

Este Trabajo Fin de Grado cuenta con una carga de 12 ECTS, equivalente a 300 horas de trabajo. Sin embargo, debido a la amplitud del proyecto y la necesidad de garantizar una implementación sólida y una documentación del proyecto de calidad, se ha considerado un margen adicional de 150 horas, elevando la dedicación total a 450 horas.

Talento es una plataforma diseñada para conectar a profesionales independientes con clientes que buscan servicios específicos, facilitando la publicación y contratación de gigs de forma sencilla y eficiente. Aunque no requiere una infraestructura altamente compleja, sí implica la integración de varias funcionalidades clave, como la gestión de usuarios, la administración de gigs, un sistema de mensajería y un flujo de pago simulado. Para abordar el desarrollo de manera estructurada, se han definido cuatro fases o incrementos, lo que permitirá implementar y validar cada funcionalidad de manera progresiva.

El desarrollo del proyecto comenzó en noviembre de 2024 y se extenderá hasta febrero

de 2025, con la siguiente distribución de horas:

- Incremento I (95 horas): Diseño de la arquitectura del sistema, definición de requisitos y desarrollo del backend, sentando las bases para una estructura escalable y bien organizada.
- Incremento II (95 horas): Implementación de las funcionalidades principales, incluyendo la gestión de gigs y usuarios, junto con la integración de un sistema de mensajería básico y eficiente.
- Incremento III (100 horas): Desarrollo del proceso de contratación de gigs, junto con un sistema de pago simulado que refleje una experiencia realista para el usuario.
- Incremento IV (160 horas): Pruebas, optimización del sistema, refinamiento de la experiencia de usuario y documentación final.

En total, se han asignado 450 horas al proyecto, distribuidas estratégicamente para asegurar la calidad del desarrollo y el cumplimiento de los plazos establecidos. Para visualizar mejor esta planificación, se utilizará un diagrama de Gantt, en el que cada día de trabajo representará 8 horas a excepción de los fines de semana por asuntos laborales, manteniendo un ritmo constante y equilibrado hasta la entrega final.

2.2.1.1 Incremento I: Fundamentos y arquitectura del sistema

El primer incremento tiene como objetivo sentar las bases del proyecto, creando una estructura sólida sobre la que se construirán las funcionalidades principales (Figura 35). Para ello, se comienza con la definición detallada de requisitos y la configuración del entorno de desarrollo. Se diseña y establece la base de datos, junto con la implementación de la API backend, que será responsable de la comunicación entre los diferentes componentes de Talento. En esta fase también se desarrolla el sistema de autenticación y gestión de usuarios, permitiendo el registro e inicio de sesión en la plataforma. Este primer incremento es crucial, ya que sienta las bases para los módulos posteriores.

Figura 36. Diagrama de Gantt Incremento I

2.2.1.2. Incremento II: Publicación y gestión de gigs

Con la arquitectura ya establecida, el segundo incremento se centra en la funcionalidad central de la plataforma: la publicación y gestión de gigs (Figura 37). En esta etapa, se desarrolla el sistema que permitirá a los usuarios publicar servicios, describiendo su oferta con detalles como precios y posibles extras. Además, se integra un sistema de búsqueda y filtrado para que los compradores puedan encontrar los gigs más relevantes según sus necesidades. Así mismo, se crea un panel de control para que tanto compradores como vendedores puedan gestionar sus servicios, consultar su historial de pedidos y administrar sus gigs. Esta fase es clave para que la plataforma comience a tomar forma y los usuarios puedan interactuar con su contenido.

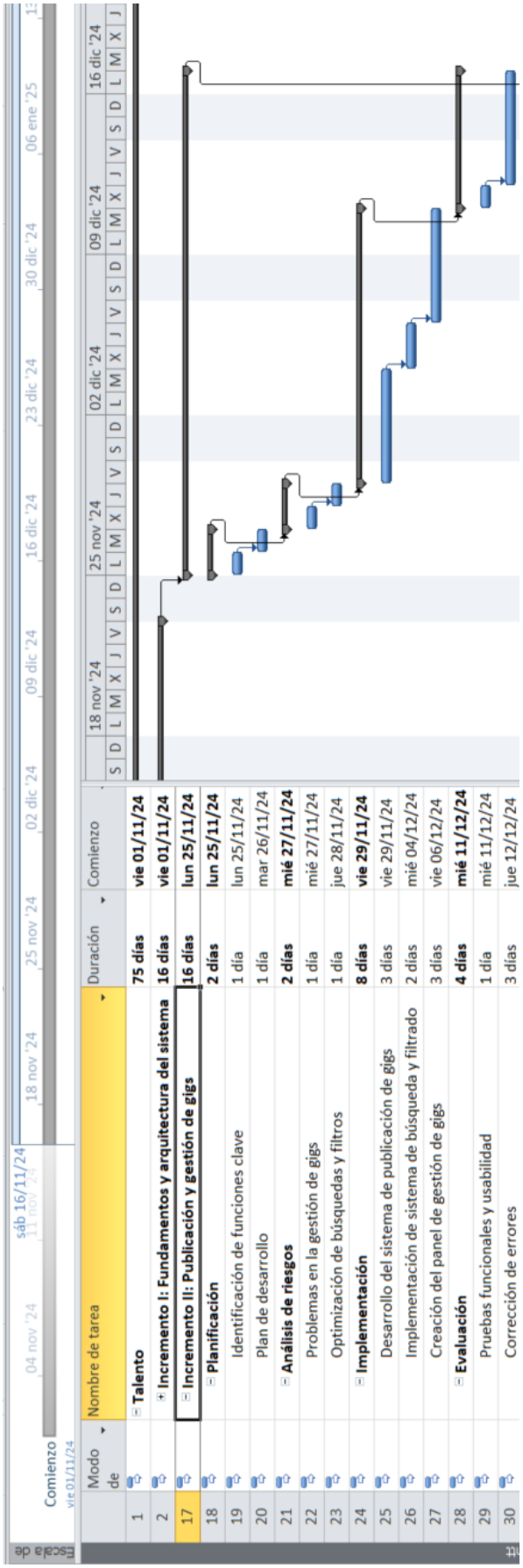
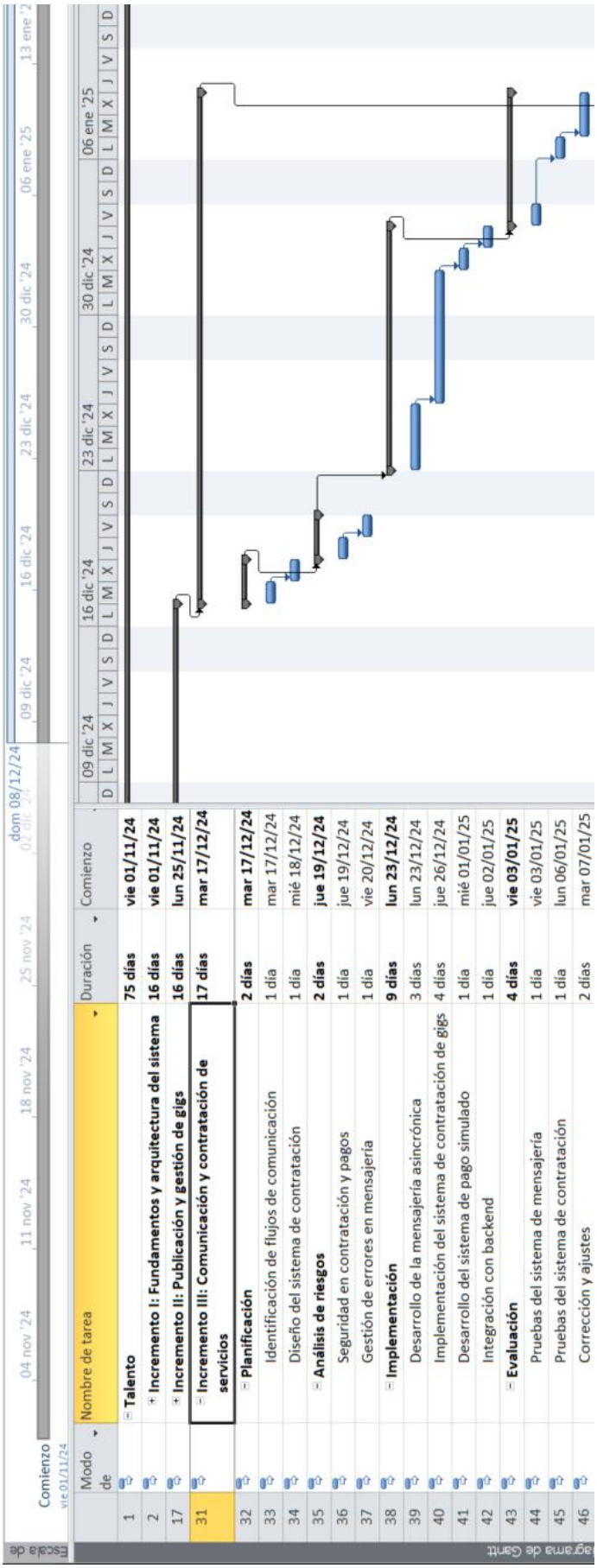


Figura 38. Diagrama de Gantt Incremento II

2.2.1.3. Incremento III: Comunicación y contratación de servicios

El tercer incremento se enfoca en la interacción entre usuarios y el proceso de contratación de servicios (Figura 39). Se integra un sistema de mensajería en la plataforma, que permitirá la comunicación entre comprador y vendedor una vez que el servicio haya sido contratado. Aunque la mensajería no será en tiempo real, se optimizará para que sea eficiente y cumpla su propósito de facilitar acuerdos entre ambas partes. También se desarrolla el sistema de contratación de gigs, donde los compradores podrán formalizar pedidos. Además, se añade un sistema de pago simulado, asegurando que el flujo de compraventa sea estructurado y claro.



	Modo de	Nombre de tarea	Duración	Comienzo
1		**Talento**	**75 días**	**vie 01/11/24**
2		**+ Incremento I: Fundamentos y arquitectura del sistema**	**16 días**	**vie 01/11/24**
17		**+ Incremento II: Publicación y gestión de gigs**	**16 días**	**lun 25/11/24**
31		**- Incremento III: Comunicación y contratación de servicios**	**17 días**	**mar 17/12/24**
32		**- Planificación**	**2 días**	**mar 17/12/24**
33		Identificación de flujos de comunicación	1 día	mar 17/12/24
34		Diseño del sistema de contratación	1 día	mié 18/12/24
35		**- Análisis de riesgos**	**2 días**	**jue 19/12/24**
36		Seguridad en contratación y pagos	1 día	jue 19/12/24
37		Gestión de errores en mensajería	1 día	vie 20/12/24
38		**- Implementación**	**9 días**	**lun 23/12/24**
39		Desarrollo de la mensajería asincrónica	3 días	lun 23/12/24
40		Implementación del sistema de contratación de gigs	4 días	jue 26/12/24
41		Desarrollo del sistema de pago simulado	1 día	mié 01/01/25
42		Integración con backend	1 día	jue 02/01/25
43		**- Evaluación**	**4 días**	**vie 03/01/25**
44		Pruebas del sistema de mensajería	1 día	vie 03/01/25
45		Pruebas del sistema de contratación	1 día	lun 06/01/25
46		Corrección y ajustes	2 días	mar 07/01/25

Figura 40. Diagrama de Gantt Incremento III

2.2.1.4. Incremento IV: Optimización, pruebas y documentación

El último incremento se enfoca en la optimización y pulido del sistema (Figura 41). Se implementa la funcionalidad de valoraciones y feedback, permitiendo a los compradores puntuar los servicios adquiridos y dejar comentarios sobre su experiencia. Esto no solo mejorará la confiabilidad de los vendedores, sino que también fomentará la mejora continua dentro de la plataforma. En esta fase, se llevan a cabo pruebas funcionales para asegurar que todos los módulos funcionen correctamente, corrigiendo posibles errores detectados. Además, se optimiza la interfaz y la experiencia de usuario, garantizando que la plataforma sea intuitiva y fácil de usar. Por último, se elabora la documentación del proyecto, cubriendo aspectos técnicos y funcionales para asegurar una correcta presentación y comprensión del sistema desarrollado (Figura 22).

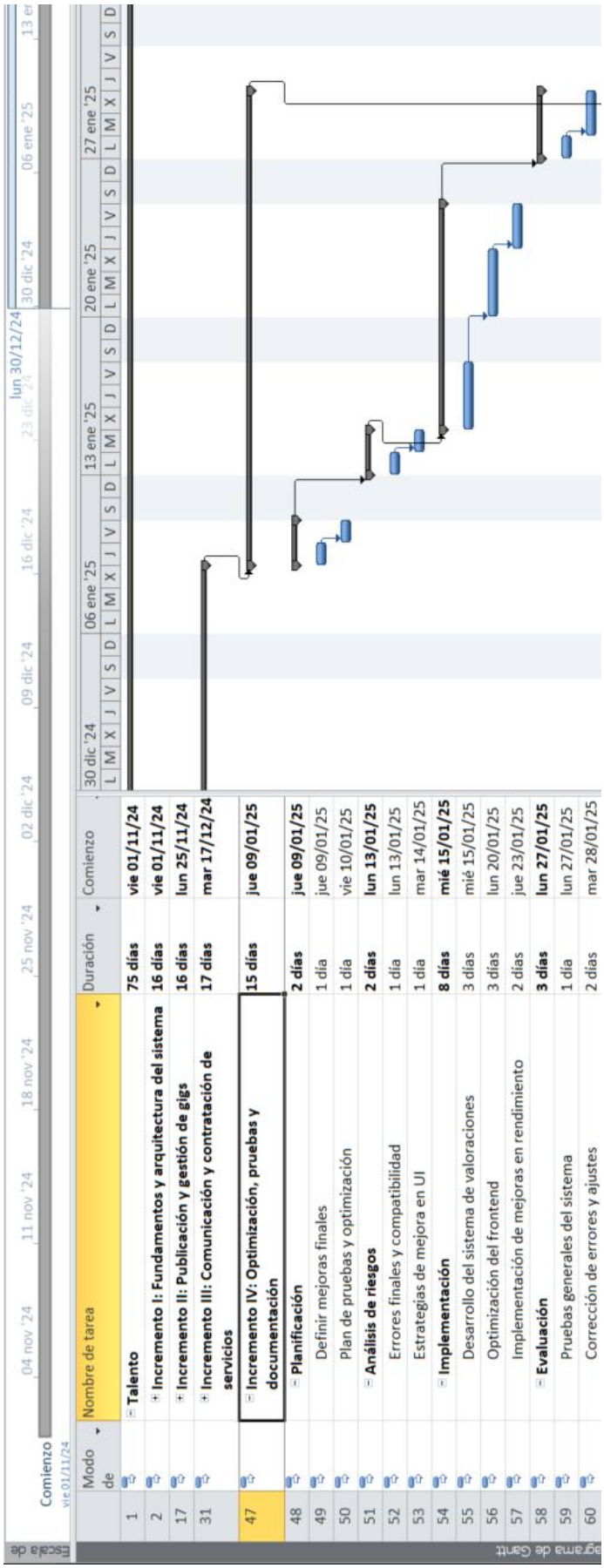


Figura 42. Diagrama de Gantt Incremento IV

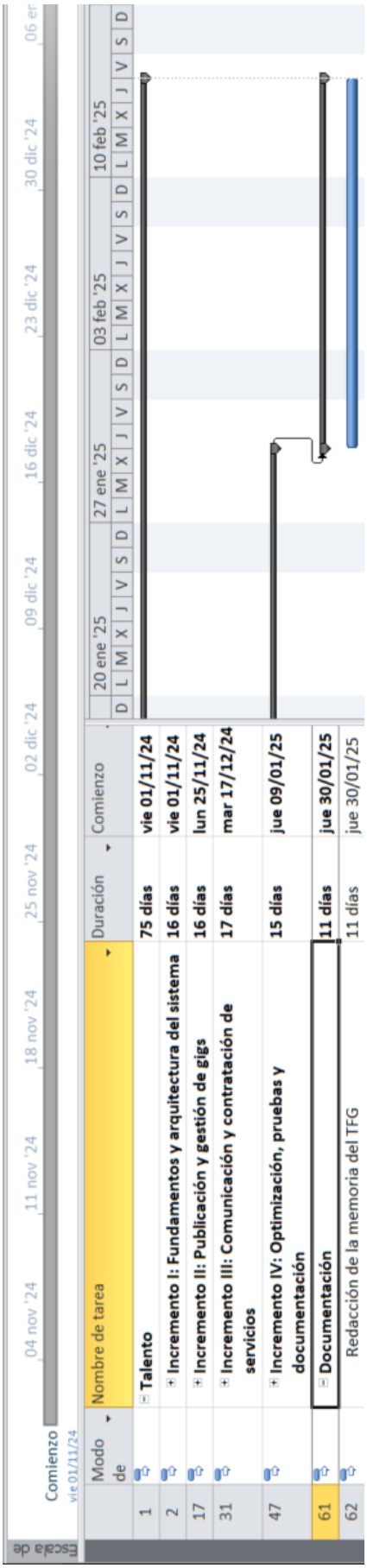


Figura 43. Diagrama de Gantt Documentación

2.2.2 Análisis de riesgos

Durante la planificación del proyecto Talento, es fundamental identificar y evaluar los posibles riesgos que podrían afectar su desarrollo y finalización dentro del plazo establecido. Una evaluación exhaustiva de riesgos no solo permite anticipar problemas, sino también minimizar sus efectos mediante estrategias de mitigación adecuadas.

En este análisis, se evaluarán los riesgos en función de dos factores clave: su probabilidad de ocurrencia y su impacto en el proyecto. Este impacto se medirá en días de retraso, lo que facilitará la priorización de los riesgos más críticos y la definición de acciones concretas para reducir sus efectos.

El análisis abarca todas las fases del proyecto, desde la planificación inicial hasta la implementación y optimización final. Para cada riesgo identificado, se ha estimado su impacto en términos de tiempo, lo que permite tomar decisiones informadas y gestionar contingencias de manera efectiva.

Para evaluar los riesgos, se han definido las siguientes escalas de probabilidad e impacto:

Probabilidad			
		≥	<
5	Casi cierto	0.80	1.00
4	Probable	0.50	0.80
3	Posible	0.20	0.50
2	Improbable	0.10	0.20
1	Raro	0.01	0.10

Tabla 2. Escala de probabilidad

Impacto		
1	Insignificante	valor < 1%
2	Menor	1% ≤ valor < 5%
3	Moderado	5% ≤ valor < 10%
4	Mayor	10% ≤ valor < 20%
5	Catastrófico	20% ≤ valor

Tabla 3. . Escala de impacto

A continuación, se enumeran y evalúan los riesgos clave que podrían afectar al proyecto:

Riesgo	Descripción	Probabilidad	Impacto	Clasificación de probabilidad	Clasificación de impacto	Costo en días
1	Errores en la planificación del proyecto que generen retrasos.	30%	8%	3 (Posible)	3 (Moderado)	5
2	Fallos en la integración de componentes backend y frontend.	60%	25%	4 (Probable)	5 (Catastrófico)	10
3	Problemas en la integración del sistema de pago simulado	40%	15%	3 (Posible)	4 (Mayor)	4
4	Errores críticos en la base de datos y pérdida de información	15%	25%	2 (Improbable)	5 (Catastrófico)	7
5	Fallos en la comunicación con la API de mensajería.	35%	3%	3 (Posible)	2 (Menor)	3
6	Falta de tiempo para la optimización y pruebas finales	70%	8%	4 (Probable)	3 (Moderado)	6
7	Sobrecarga de tareas y dificultades en la gestión del tiempo	70%	15%	4 (Probable)	4 (Mayor)	7

Tabla 4. Análisis de riesgos

Para minimizar el impacto de los riesgos identificados en el desarrollo de la plataforma, se han diseñado estrategias de mitigación y reducción. Estas medidas permiten anticipar posibles problemas y establecer soluciones preventivas, con el objetivo de evitar retrasos y garantizar el éxito del proyecto.

A continuación, se detallan los planes específicos para cada riesgo:

Riesgo	Exposición	Plan de Reducción	Plan de Mitigación
1	2.4%	Definir un cronograma detallado y revisarlo semanalmente para detectar posibles desviaciones.	Si hay retrasos, ajustar fechas y priorizar tareas críticas.
2	15%	Realizar pruebas frecuentes en cada fase para asegurar la compatibilidad entre backend y frontend.	Si la integración falla, dividir la tarea en pasos más pequeños y corregir errores por partes.
3	6%	Probar la integración con datos simulados antes de implementarla completamente.	Si hay problemas, simplificar la lógica o buscar otra forma de realizar la operación.
4	3.75%	Hacer copias de seguridad periódicas de la base de datos.	En caso de pérdida de datos, restaurar la última copia de seguridad y corregir errores antes de continuar.
5	1.05%	Comprobar la compatibilidad de la API con pruebas básicas antes de la implementación.	Si falla, revisar los parámetros de conexión y probar alternativas.
6	5.6%	Reservar tiempo en cada incremento para pruebas y optimización.	Si no se alcanza a probar todo, priorizar las funciones principales.
7	10.5%	Organizar bien el tiempo y dividir las tareas en objetivos diarios alcanzables.	Si hay sobrecarga, redistribuir tareas y reducir funcionalidades no esenciales.

Tabla 5. Plan de reducción y mitigación

2.2.3 Presupuesto

El presupuesto de Talento se calculará combinando dos metodologías: por un lado, el método IFPUG (Albrecht de Puntos de Función), que evalúa el esfuerzo necesario según la complejidad del software; y por otro, una estimación basada en la planificación inicial que tiene en cuenta los recursos humanos y técnicos requeridos durante todo el desarrollo.

Para asegurar la precisión, contrastaremos ambos cálculos y analizaremos las diferencias, lo que nos permitirá determinar el coste total del proyecto con mayor fiabilidad.

2.2.3.1 Estimación mediante el método IFPUG (Método de Albretch de Puntos de Función)

El método de Albrecht permite cuantificar las funcionalidades de un sistema a través de puntos de función (PF), proporcionando una medida objetiva del tamaño y complejidad de una aplicación. Este enfoque es muy utilizado en la industria del software debido a su capacidad para estimar con precisión el esfuerzo, coste y productividad del desarrollo.

El método divide el sistema en cinco tipos de componentes funcionales:

- Entradas externas (EE): Información introducida en el sistema por el usuario o por otros servicios, como el registro de usuarios.
- Salidas externas (SE): Datos generados tras un procesamiento interno, como la visualización de gigs y pedidos en la plataforma.
- Consultas externas (CE): Acciones que implican la recuperación de información sin modificar la base de datos, como la búsqueda y filtrado de gigs.
- Ficheros lógicos internos (FLI): Conjuntos de datos almacenados dentro del sistema, como los perfiles de usuarios y servicios ofertados en Talento.
- Ficheros de interfaz externa (FIE): Datos administrados fuera de la aplicación, pero utilizados en el sistema, como futuras integraciones con APIs de terceros.

Cada uno de estos elementos se evalúa en función de su complejidad y se les asigna un número específico de puntos de función. Luego, se aplican factores de ajuste basados en las características del proyecto para obtener una estimación más precisa del esfuerzo en horas de desarrollo.

Entradas externas (EE):

Tipos de ficheros	**Datos elementales**		
	1-4	5-15	>16
0-1	Baja	Baja	Media
2-3	Baja	Media	Alta
>3	Media	Alta	Alta

Tabla 6. Complejidad entradas externas

Entrada externa	**Tipos de ficheros**	**Datos elementales**	**Complejidad**
Registrarse	0-1	5-15	Baja
Iniciar sesión	0-1	1-4	Baja
Actualizar datos del perfil	0-1	5-15	Baja
Crear gig	2-3	5-15	Media
Marcar pedido como entregado	0-1	1-4	Baja
Enviar mensaje	0-1	1-4	Baja
Publicar opinión	0-1	1-4	Baja
Pagar un gig	0-1	1-4	Baja
Filtrar gigs	0-1	1-4	Baja

Tabla 7. Entradas externas

Salidas Externas (SE):

Tipos de ficheros	Datos elementales		
	1-4	5-15	>16
0-1	Baja	Baja	Media
2-3	Baja	Media	Alta
>3	Media	Alta	Alta

Tabla 8. Complejidades salidas y consultas externas

Salida externa	Tipos de ficheros	Datos elementales	Complejidad
Mostrar detalles de un gig	0-1	5-15	Baja
Mostrar mis pedidos	2-3	5-15	Media
Mostrar trabajos contratados	2-3	5-15	Media
Mostrar anuncios activos	2-3	5-15	Media
Mostrar conversaciones	2-3	5-15	Media
Mostrar mensajes de una conversación	0-1	5-15	Baja
Mostrar perfil de usuario	0-1	5-15	Baja
Mostrar opiniones de un gig	0-1	5-15	Baja

Tabla 9. Salidas externas

Consultas Externas (CE):

Consulta externa	Tipos ficheros	Datos elementales	Complejidad
Buscar gigs filtrados	2-3	5-15	Media
Buscar mensajes en conversación	0-1	1-4	Baja
Buscar opiniones de un gig	0-1	1-4	Baja
Buscar detalles de un pedido	0-1	1-4	Baja
Buscar detalles de un gig	0-1	5-15	Baja
Obtener mi perfil de usuario	0-1	5-15	Baja
Buscar conversaciones activas	0-1	5-15	Baja

Tabla 10. Consultas externas

Ficheros Lógicos internos (FLI):

Tipos de ficheros	Datos elementales		
	1-19	20-50	>51
1	Baja	Baja	Media
2-5	Baja	Media	Alta
>5	Media	Alta	Alta

Tabla 11. Complejidad ficheros lógicos internos y de interfaz externa

Fichero Lógico Interno (FLI)	Tipos de ficheros	Datos elementales	Complejidad
Tabla usuarios	1	5-15	Baja
Tabla gigs	1	5-15	Baja
Tabla categorías	1	1-4	Baja
Tabla pedidos	1	5-15	Baja
Tabla mensajes	1	1-4	Baja
Tabla conversaciones	1	1-4	Baja
Tabla valoraciones	1	1-4	Baja

Tabla 12. Ficheros lógicos internos

Ficheros de interfaz externa (FIE):

En esta versión no contamos con ficheros de interfaz externa, por lo tanto, no lo tendremos en cuenta en el cálculo de los PFNA.

Componente	Complejidad	Ponderación	Total por tipo	Total x complejidad
Entradas Externas	Baja	x3	8	24
	Media	x4	1	4
	Alta	x6	0	0
Salidas Externas	Baja	x4	4	16
	Media	x5	4	20
	Alta	x7	0	0
Consultas Externas	Baja	x3	6	18
	Media	x4	1	4
	Alta	x6	0	0
Ficheros Lógicos Internos	Baja	x7	7	49
	Media	x10	0	0
	Alta	x15	0	0
Total PFNA				**135**

Tabla 13. Ponderaciones PFNA

Cálculo del Factor de ajuste (FA):

Una vez calculados los puntos de función no ajustados (PFNA), es necesario realizar un ajuste. Debemos calcularlo mediante el Factor de Ajuste (FA). Basamos estos cálculos en 14 características comunes de los sistemas que miden la funcionalidad y la influencia en la aplicación.

N.º	Factor de Influencia	Influencia
1	Comunicación de datos	4
2	Funciones distribuidas	3
3	Prestaciones	3
4	Gran uso de la configuración	1
5	Velocidad de las transacciones	3
6	Entrada on-line de datos	5
7	Diseño para la eficiencia del usuario final	4
8	Actualización de datos on-line	4
9	Complejidad del proceso lógico interno	3
10	Reusabilidad del código	2
11	Facilidad de instalación	2
12	Facilidad de operación	3
13	Localizaciones múltiples	1
14	Facilidad de cambios	3
Total		**41**

Tabla 14. Factores de Ajuste (FA)

Los factores con mayor impacto en la complejidad y esfuerzo de desarrollo del sistema son los siguientes:

- Entrada on-line de datos: El sistema depende de formularios para registrar usuarios, publicar gigs, enviar mensajes, etc. Esta constante interacción justifica su influencia máxima.
- Diseño para la eficiencia del usuario final: Talento esta orientado al usuario no técnico, tanto cliente como vendedor, por lo que la interfaz y la fluidez de navegación son de gran importancia. Este enfoque requiere una atención especial en usabilidad, lo que aumenta el esfuerzo de diseño y validación.
- Actualización de datos on-line: Los usuarios modifican con frecuencia su información, los gigs y los pedidos. Estas operaciones deben hacerse en tiempo real sin interrupciones, lo que demanda una arquitectura sólida y bien optimizada.
- Comunicación de datos: Aunque actualmente no se utilizan APIs externas, el sistema posee múltiples puntos de entrada y salida de datos entre frontend y backend, con diversas rutas protegidas por autenticación. Esto requiere un flujo de comunicación bien estructurado y seguro.

El cálculo del Factor de Ajuste (FA) se realiza aplicando la fórmula:

$$FA = 0{,}65 + \left(0{,}01 \times \sum_{i=1}^{14} influencia\right) = 0{,}65 + (0{,}01 \times 41) = 1{,}06\ FA$$

Cálculo de los puntos de función ajustados (PFA):

Una vez conocido el factor de ajuste podemos calcular los puntos de función ajustados (PFA), utilizamos la siguiente fórmula:

$$PFA = PFNA \times FA = 135 \times 1{,}06 = 143{,}1\ PFA$$

Cálculo del tiempo en días de esfuerzo

Para estimar el tiempo de desarrollo de un proyecto lo haremos a través del esfuerzo asociado a los Puntos de Función. Según estándares habituales, se considera que un equipo puede implementar aproximadamente 13 Puntos de Función por mes (21 días laborables). Con un total de 143,1 Puntos de Función, la estimación de duración se obtiene dividiendo este valor por la productividad mensual:

$$\frac{143{,}1\ PF}{13\ PF/Mes} = 11{,}01\ meses$$

Según los cálculos serán necesarios 11,01 meses para llevar a cabo el proyecto.

Después de conocer la duración estimada del proyecto calcularemos de forma aproximada el coste de desarrollar la plataforma, utilizando como base la metodología de Puntos de Función de Albrecht. Este enfoque permite calcular los recursos técnicos y humanos necesarios, ofreciendo así una visión ajustada y realista del presupuesto total.

Recursos técnicos

Durante el desarrollo, se han empleado herramientas de software libre y otras proporcionadas por la Universidad de Valladolid, lo que ha evitado tener que adquirir licencias de pago. Esto ha permitido reducir considerablemente el coste asociado al apartado de software.

En cuanto al hardware, se ha utilizado un único equipo informático (ordenador, monitor, teclado y ratón) que tuvo un coste de 800 €. Considerando una vida útil estimada de 4 años y que el proyecto se ha prolongado durante 11 meses, se ha utilizado el 22,91 % de ese tiempo. Por tanto, el coste añadido al proyecto asciende a 183,28 €.

A este gasto se suman los costes operativos de luz e internet. El consumo eléctrico ha supuesto una media mensual de 55 €, lo que en 11 meses suma 605 €. Por su parte, el servicio de internet ha tenido un coste mensual de 40 €, acumulando un total de 440 €.

Concepto	Importe
Inversión en hardware	183,28€
Electricidad	605,00€
Internet	440,00€
Total	**1.228,28€**

Tabla 15. Presupuesto de los recursos técnicos (Método Albrecht)

En conjunto, los recursos técnicos han supuesto un gasto de 1.228,28 €, desglosados entre hardware y servicios.

Recursos Humanos

En lo que respecta a los recursos humanos, el grueso del presupuesto se debe al tiempo dedicado por el desarrollador. Si se toma como referencia un salario medio de 2.500 € al mes, y se considera que el proyecto ha requerido 11 meses de trabajo, el coste total asciende a 27.500 €. Esta cifra engloba todas las tareas realizadas: análisis, diseño, desarrollo, pruebas y despliegue final.

Estimación del presupuesto Total

En resumen, el presupuesto total estimado se distribuye de la siguiente forma:

Concepto	Importe
Recursos técnicos	1.228,28€
Recursos humanos	27.500,00€
Total	**28.728,28 €**

Tabla 16. Estimación del presupuesto total (Método Albrecht)

2.2.3.2 Estimación basada en la planificación inicial

A partir de la planificación inicial, donde se estimó una duración de tres meses para el desarrollo de la plataforma Talento, puede elaborarse una estimación alternativa del presupuesto calculado a través del método Albrecht, más ajustada en tiempo y, por tanto, también en costes.

Recursos técnicos

Como ya se ha comentado en el apartado anterior, no ha sido necesario invertir en software gracias al uso de herramientas libres y licencias proporcionadas por la Universidad de Valladolid. En cuanto al equipo informático utilizado, el mismo que se detalló anteriormente, con un coste de 800 € y una vida útil de cuatro años, su uso durante cuatro meses representa solo el 8,33 % del total, lo que supone una imputación de 66,70 € al proyecto.

Respecto a los costes operativos, el consumo eléctrico (55 €/mes) y el acceso a internet (40 €/mes) durante cuatro meses se traducen en un gasto total de 220 € y 160 € respectivamente. Sumando estos importes al del hardware, el coste técnico asciende a 446,70 €.

Concepto	Importe
Inversión en hardware	66,70€
Electricidad	220,00€
Internet	160,00€
Total	**446,70€**

Tabla 17. Presupuesto de los recursos técnicos (Planificación Inicial)

Recursos Humanos

En cuanto a los recursos humanos, tomando como referencia el salario medio mensual de un desarrollador (2.500 €), el coste de cuatro meses de trabajo se sitúa en 10.000 €, abarcando todas las tareas necesarias para el desarrollo funcional de la plataforma.

Estimación del presupuesto Total

Así, el presupuesto total estimado con base en esta planificación inicial sería:

Concepto	Importe
Recursos técnicos	446,70€
Recursos humanos	10.000,00€
Total	**10.446,70 €**

Tabla 18. Estimación del presupuesto total (Planificación Inicial)

2.2.3.3 Comparación de los presupuestos

Tras haber estimado el presupuesto del proyecto utilizando dos enfoques distintos, uno basado en los Puntos de Función de Albrecht y otro en la planificación inicial, es posible comparar ambos resultados y valorar las diferencias entre ellos.

Por un lado, la estimación obtenida a partir del método de Albrecht nos dio como resultado un presupuesto total de 28.728,28 €, teniendo en cuenta una duración de once meses y una mayor carga funcional del proyecto. En cambio, si se toma como referencia la planificación inicial, que planteaba un desarrollo más acotado en el tiempo, el presupuesto estimado se reduce considerablemente hasta los 10.446,70 €

Esta diferencia, de casi 18.500 €, se explica sobre todo al tiempo de dedicación del desarrollo, ya que el peso del coste en ambos casos corresponde a los recursos humanos. En cuanto a los recursos técnicos, como ya se explicó anteriormente, se mantienen prácticamente constantes gracias al uso de software libre y la reutilización de equipo ya disponible.

En resumen, esta comparación deja ver cómo el método utilizado para estimar el esfuerzo impacta directamente en presupuesto final del proyecto. La planificación inicial ofrece una visión más optimista y ajustada para contextos académicos, mientras que el enfoque basado en los Puntos de Función refleja de forma más realista el esfuerzo total necesario para construir la plataforma con todas sus funcionalidades.

Ambas visiones son válidas y complementarias, y permiten entender mejor el alcance económico del proyecto.

2.2.4 Seguimiento Real

En la planificación inicial del proyecto, se estimó que el desarrollo de la plataforma Talento se completaría en un plazo de cuatro meses, con el objetivo de finalizarlo a finales

de febrero. Sin embargo, el desarrollo real no siguió exactamente ese calendario. A medida que avanzaba el trabajo, surgieron imprevistos que hicieron que el proyecto se extendiera más de lo previsto.

La complejidad técnica resultó ser mayor de lo que se había previsto en un primer momento. Algunas funcionalidades exigieron más tiempo de análisis y desarrollo del que se había estimado, lo que provocó ciertos retrasos. Además, a comienzos de febrero, empecé a trabajar a jornada completa, lo que redujo el tiempo que podía dedicar al TFG. Esta nueva situación obligó a reorganizar el tiempo disponible y adaptar la planificación inicial a una realidad más exigente.

Como resultado, el proyecto se prolongó hasta el mes de mayo, prácticamente duplicando la duración prevista. Aun así, el progreso fue en cierto modo constante y gracias al trabajo realizado fuera del horario laboral, se consiguió completar el proyecto dentro de los nuevos plazos. Esta experiencia refleja no solo las dificultades habituales en un desarrollo real, sino también la importancia de la flexibilidad y la capacidad de adaptación ante los cambios.

2.2.4.1 Coste real del proyecto

Una vez finalizado el desarrollo del proyecto ha sido posible ajustar el presupuesto estimado en base al tiempo real dedicado. Como se explicó en apartados anteriores, inicialmente se preveía una duración de cuatro meses a jornada completa. Sin embargo, el desarrollo se prolongó hasta el mes de mayo, sumando un total de siete meses. Durante los últimos cuatro meses el tiempo disponible para realizar el TFG se redujo significativamente. Por este motivo, se ha considerado una dedicación parcial en este periodo, estimando una media de un 33 % de la jornada. Esto equivale, en términos de esfuerzo, a 1,33 meses adicionales a jornada completa.

Con este ajuste, el total de tiempo invertido se traduce en 4,33 meses a tiempo completo. Tomando como referencia un salario medio de 2.500 € al mes, el coste total en recursos humanos asciende a 10.825 €. Por su parte, los recursos técnicos también se han recalculado teniendo en cuenta los siete meses de duración, lo que supone un coste amortizado de 116,67 € por el equipo informático, 385 € en consumo eléctrico y 280 € por el servicio de internet. En total, los costes técnicos se sitúan en 781,67 €.

Concepto	Importe
Recursos técnicos	781,67€
Recursos humanos	10.825,00€
Total	**11.606,67 €**

Tabla 19. Coste real

Así, el presupuesto final ajustado tras el seguimiento real del proyecto alcanza los 11.606,67 €, reflejando de forma más realista tanto la carga de trabajo como los recursos utilizados durante el desarrollo de la plataforma.

2.2.4.2 Comparación del coste real con el coste de la planificación inicial

Finalmente compararemos el coste estimado inicialmente con el presupuesto final ajustado tras el seguimiento real del desarrollo. En la planificación original se contemplaba una duración de cuatro meses a jornada completa, lo que arrojaba un coste total de 10.446,70 €, sumando tanto los recursos humanos como los técnicos.

Sin embargo, como se ha explicado en el apartado anterior, la duración real del proyecto se extendió hasta mayo, y aunque durante los últimos meses la dedicación fue parcial, el esfuerzo total equivale a 4,33 meses a jornada completa. Esto ha supuesto un incremento notable del presupuesto, especialmente en el apartado de recursos humanos, que alcanzó los 10.825 €. También los costes técnicos aumentaron proporcionalmente con la duración, situándose en 781,67 € frente a los 446,70 € estimados inicialmente.

En conjunto, el coste real del proyecto ascendió a 11.606,67 €, lo que supone un incremento de 1.159,97 € respecto a la planificación original. Este aumento se debe principalmente a la prolongación del desarrollo, que fue consecuencia tanto de la complejidad técnica como de cambios en la disponibilidad horaria. Aun así, el coste adicional puede considerarse razonable si se tiene en cuenta que permitió finalizar el proyecto en condiciones adecuadas, manteniendo su calidad y funcionalidad.

Capítulo 3
Análisis del sistema

En este capítulo se presenta el análisis completo realizado como base para el desarrollo del proyecto incluyendo tablas y diagramas explicativos.

3.1. Requisitos de Usuario

En esta primera parte del análisis, vamos a definir los requisitos de usuario de la plataforma. Estos requisitos no son más que una forma de expresar lo que los usuarios necesitan y esperan del sistema. Los presentamos en la siguiente tabla:

ID	Descripción
RU_01	El usuario debe poder registrarse en la plataforma.
RU_02	El usuario debe poder iniciar sesión en la plataforma con sus credenciales.
RU_03	El usuario debe poder cerrar su sesión en la plataforma.
RU_04	El usuario debe poder buscar gigs específicos filtrados por palabras clave o categorías.
RU_05	El usuario debe poder visualizar la información detallada de un gig antes de contratarlo.
RU_06	El usuario debe poder contratar un gig seleccionado y proceder al pago.
RU_07	El usuario debe poder pagar el gig que quiere contratar.
RU_08	El usuario debe poder consultar una lista de los gigs que ha contratado junto con su estado.
RU_09	El usuario debe poder escribir una opinión sobre un gig contratado.
RU_10	El usuario (cliente y vendedor) debe poder interactuar mediante mensajes.
RU_11	El usuario debe poder marcar mensajes en el chat como leídos para una mejor gestión de la conversación.
RU_12	El usuario debe poder visualizar su perfil.
RU_13	El usuario debe poder modificar su información personal desde la página de perfil.
RU_14	El usuario debe poder eliminar su cuenta y toda su información asociada.
RU_15	El usuario debe poder activar la opción de vender gigs en la plataforma.
RU_16	El usuario vendedor debe poder ver los gigs que ha publicado y están activos.
RU_17	El usuario vendedor debe poder crear nuevos gigs.
RU_18	El usuario vendedor debe poder eliminar gigs existentes.
RU_19	El usuario vendedor debe poder visualizar los trabajos que han sido contratados por los clientes.
RU_20	El usuario vendedor debe poder marcar un gig contratado como entregado una vez que lo ha finalizado.

Tabla 20. Requisitos de Usuario

3.2. Casos de Uso

En esta segunda parte del análisis se describen los distintos casos de uso para Talento. Cada caso de uso representa una interacción entre un usuario y el sistema, con un objetivo determinado. Estos casos de uso permiten identificar cómo interactuarán los usuarios con la aplicación y qué funcionalidades deben implementarse para cubrir sus necesidades.

3.2.1 Especificación de los Actores

En este apartado se describen los actores que interactúan con el sistema, ya sean individuos, roles o sistemas externos. Se detallan sus características y responsabilidades para garantizar una comprensión clara de sus funciones dentro del proyecto.

ACT_01	Usuario no registrado
Descripción	Este actor representa a un usuario que no se ha registrado en la plataforma. Aunque no requiere tener una cuenta, sus funcionalidades están limitadas. Los usuarios no registrados pueden buscar gigs y ver detalles de estos, pero no pueden contratar gigs, pagar, interactuar con vendedores ni acceder a funciones avanzadas como gestionar un perfil o publicar opiniones.
Comentario	El usuario no registrado tiene acceso a la exploración básica de la plataforma, lo que le permite conocer los servicios disponibles y así captar nuevos usuarios, ya que su experiencia en la plataforma puede motivarlo a registrarse.

Tabla 21. Especificación del actor Usuario no registrado

ACT_02	Vendedor
Descripción	Este actor representa a un usuario registrado que ofrece servicios (gigs) en la plataforma. Puede crear, gestionar y eliminar gigs, ver los trabajos que han sido contratados por los clientes, marcar entregas como completadas y comunicarse con los clientes a través del sistema de mensajes. Además de contar con las mismas funcionalidades que un cliente, ya que un vendedor puede comprar a otros vendedores.
Comentario	El vendedor es fundamental para mantener una oferta diversa y atractiva de servicios en la plataforma. Su interacción con los clientes no solo crea transacciones, también favorece a la reputación y dinamismo del ecosistema de la plataforma.

Tabla 22. Especificación del actor Vendedor

ACT_03	Cliente
Descripción	Este actor representa a un usuario registrado que contrata servicios (gigs) en la plataforma. Este actor puede buscar servicios, ver los detalles de estos, contratarlos, pagarlos, visualizar sus pedidos, publicar opiniones y comunicarse con los vendedores a través del sistema de mensajes.
Comentario	El cliente también es fundamental ya que es el actor encargado de impulsar la economía dentro de la plataforma. Su interacción con los vendedores crea transacciones que generan valor y completan el ciclo útil de la plataforma.

Tabla 23. Especificación del actor Cliente

3.2.2 Listado de Casos de Uso

ID	Caso de Uso	Descripción
CU_01	Registrarse	Permite al usuario registrarse en la plataforma.
CU_02	Buscar gigs	Permite buscar gigs específicos filtrados por palabras clave o categorías.
CU_03	Ver detalles de un gig	Permite a los usuarios visualizar la información detallada de un gig antes de contratarlo.
CU_04	Iniciar sesión	Permite al usuario iniciar sesión en la plataforma con sus credenciales.
CU_05	Eliminar perfil	Permite a los usuarios eliminar su cuenta y toda su información asociada.
CU_06	Cerrar sesión	Permite al usuario cerrar su sesión en la plataforma.
CU_07	Actualizar datos del perfil	Permite al usuario visualizar y modificar su información personal en la página de perfil.
CU_08	Ver mis pedidos	Permite consultar una lista de los gigs que el usuario ha contratado junto con su estado.
CU_09	Ver conversaciones	Permite a los usuarios (cliente y vendedor) visualizar los chats en los que participa.
CU_10	Enviar mensajes	Permite al usuario enviar mensajes en el chat.
CU_11	Marcar chat como leído	Permite al usuario marcar como leídos los chats que desee.
CU_12	Contratar un gig	Permite al usuario contratar un gig seleccionado.
CU_13	Pagar un gig	Permite al usuario completar el proceso de pago.
CU_14	Publicar opinión	Permite al usuario escribir una opinión sobre un gig contratado.
CU_15	Activar cuenta de Vendedor	Permite a un usuario activar la opción de vender gigs en la plataforma.
CU_16	Crear gigs	Permite al usuario vendedor crear nuevos gigs.
CU_17	Ver anuncios activos	Permite al usuario vendedor ver los gigs que ha publicado y están activos.
CU_18	Eliminar gigs	Permite al usuario vendedor eliminar gigs existentes.
CU_19	Ver trabajos contratados	Permite al usuario vendedor ver los gigs que han sido contratados por los clientes.
CU_20	Marcar como entregado	Permite al usuario vendedor marcar un gig contratado como entregado una vez que lo ha finalizado.

Tabla 24. Casos de Uso

3.2.3 Diagrama de Casos de Uso

En este apartado se presenta el diagrama de casos de uso (Figura 44):

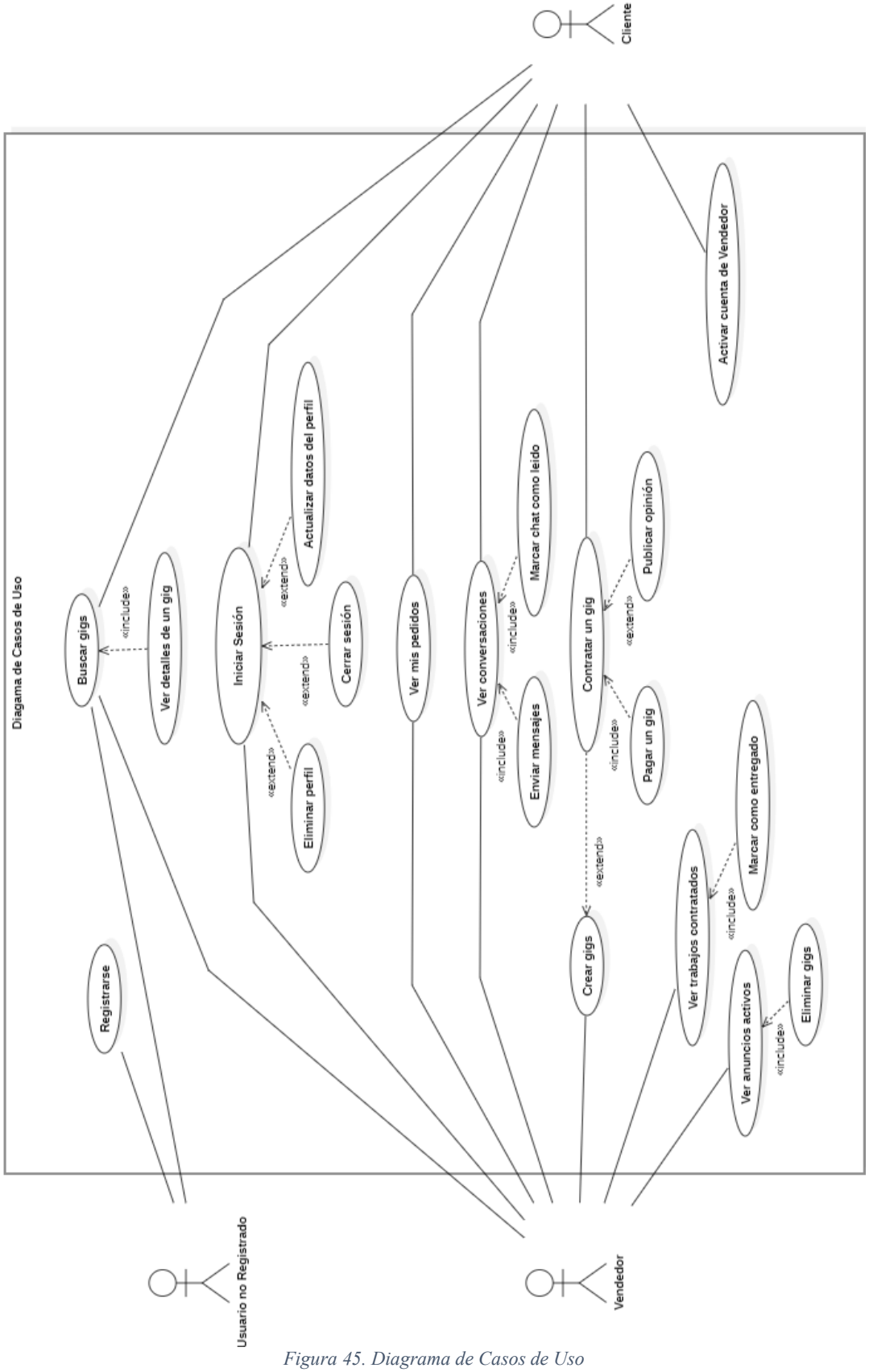


Figura 45. Diagrama de Casos de Uso

3.2.4 Especificación de Casos de Uso

La especificación de casos de uso detalla las funcionalidades clave de la plataforma, describiendo las interacciones entre los usuarios y el sistema. En algunos casos, se incluyen precondiciones racionales o de sentido común que, aunque no son estrictamente necesarias para ejecutar el caso de uso, permiten que este aporte mayor valor e información cuando se cumplen.

CU-01	Registrarse
Actor Principal	Usuario no registrado.
Descripción	Permite al usuario registrarse en la plataforma.
Precondiciones	El usuario no debe tener una cuenta registrada.
Flujo Normal	1. El usuario ingresa sus datos personales. 2. El sistema valida los datos. 3. El sistema crea la cuenta y notifica al usuario.
Postcondiciones	El usuario queda registrado en la plataforma.
Excepciones	1. Los datos ingresados son inválidos. 2. El correo electrónico ya está registrado.
Importancia	Alta.

Tabla 25. Especificación del CU-01

CU-02	Buscar gigs
Actor Principal	Cliente o vendedor.
Descripción	Permite buscar gigs específicos filtrados por palabras clave o categorías.
Precondiciones	Ninguna.
Flujo Normal	1. El usuario ingresa palabras clave o selecciona una categoría. 2. El sistema muestra los gigs que coinciden con la búsqueda.
Postcondiciones	El usuario visualiza una lista de gigs filtrados.
Excepciones	1. No se encuentran gigs que coincidan con la búsqueda.
Importancia	Alta.

Tabla 26. Especificación del CU-02

CU-03	Ver detalles de un gig
Actor Principal	Cliente o vendedor.
Descripción	Permite a los usuarios visualizar la información detallada de un gig antes de contratarlo.
Precondiciones	El usuario debe haber buscado gigs.
Flujo Normal	1. El usuario selecciona un gig de la lista. 2. El sistema muestra los detalles del gig.
Postcondiciones	El usuario visualiza la información detallada del gig.
Excepciones	1. El gig seleccionado no está disponible.
Importancia	Alta.

Tabla 27.Especificación del CU-03

CU-04	Iniciar sesión
Actor Principal	Cliente o vendedor.
Descripción	Permite al usuario iniciar sesión en la plataforma con sus credenciales.
Precondiciones	El usuario debe estar registrado.
Flujo Normal	1. El usuario ingresa su nombre de usuario y contraseña. 2. El sistema valida las credenciales. 3. El sistema inicia la sesión.
Postcondiciones	El usuario accede a su cuenta.
Excepciones	1. Las credenciales son incorrectas. 2. La cuenta no existe.
Importancia	Alta.

Tabla 28. Especificación del CU-04

CU-05	Eliminar perfil
Actor Principal	Cliente o vendedor.
Descripción	Permite a los usuarios eliminar su cuenta y toda su información asociada.
Precondiciones	El usuario debe haber iniciado sesión.
Flujo Normal	1. El usuario selecciona "Eliminar cuenta". 2. El sistema confirma la eliminación. 3. El sistema elimina la cuenta y todos los datos asociados.
Postcondiciones	La cuenta y los datos asociados se eliminan.
Excepciones	1. El usuario cancela la eliminación. 2. El usuario vendedor tiene pendiente entregar encargos.
Importancia	Alta.

Tabla 29. Especificación del CU-05

CU-06	Cerrar sesión
Actor Principal	Cliente o vendedor.
Descripción	Permite al usuario cerrar su sesión en la plataforma.
Precondiciones	El usuario debe haber iniciado sesión.
Flujo Normal	1. El usuario selecciona "Cerrar sesión". 2. El sistema cierra la sesión y redirige al usuario a la página principal.
Postcondiciones	El usuario ya no tiene acceso a su cuenta.
Excepciones	Ninguna.
Importancia	Media.

Tabla 30. Especificación del CU-06

CU-07	Actualizar datos del perfil
Actor Principal	Cliente o vendedor.
Descripción	Permite al usuario visualizar y modificar su información personal en la página de perfil.
Precondiciones	El usuario debe haber iniciado sesión.
Flujo Normal	1. El usuario accede a su perfil. 2. El usuario modifica su información. 3. El sistema guarda los cambios.
Postcondiciones	La información del perfil se actualiza.
Excepciones	1. Los datos ingresados son inválidos.
Importancia	Media.

Tabla 31. Especificación del CU-07

CU-08	Ver mis pedidos
Actor Principal	Cliente o vendedor actuando como cliente.
Descripción	Permite consultar una lista de los gigs que el usuario ha contratado junto con su estado.
Precondiciones	El usuario debe haber contratado al menos un gig.
Flujo Normal	1. El usuario accede a "Mis pedidos". 2. El sistema muestra la lista de gigs contratados y su estado.
Postcondiciones	El usuario visualiza sus pedidos.
Excepciones	1. No hay gigs contratados.
Importancia	Media.

Tabla 32. Especificación del CU-08

CU-09	Ver conversaciones
Actor Principal	Cliente o vendedor.
Descripción	Permite a los usuarios visualizar los chats en los que participa.
Precondiciones	El usuario debe haber contratado al menos un gig con un vendedor.
Flujo Normal	1. El usuario accede a la sección de conversaciones. 2. El sistema muestra las conversaciones activas. 3. El usuario selecciona una conversación para visualizar los mensajes.
Postcondiciones	Las conversaciones se muestran correctamente y el usuario puede gestionar sus chats.
Excepciones	1. No hay mensajes nuevos. 2. No hay conversaciones asociadas al usuario.
Importancia	Media.

Tabla 33. Especificación del CU-09

CU-10	Enviar mensajes
Actor Principal	Cliente o vendedor.
Descripción	Permite a los usuarios visualizar los chats en los que participa.
Precondiciones	El usuario debe haber contratado al menos un gig con un vendedor.
Flujo Normal	1. El usuario accede al chat. 2. El usuario redacta y envía un mensaje. 3. El sistema registra el mensaje.
Postcondiciones	El mensaje se registra en el chat correspondiente.
Excepciones	Ninguna.
Importancia	Media.

Tabla 34. Especificación del CU-10

CU-11	Marcar chat como leido
Actor Principal	Cliente o vendedor.
Descripción	Permite a los usuarios marcar una conversación como leída.
Precondiciones	El usuario debe haber contratado al menos un gig con un vendedor.
Flujo Normal	1. El usuario accede al chat. 2. El usuario marca el chat como leído. 3. El sistema actualiza el estado del chat.
Postcondiciones	El estado del chat es actualizado
Excepciones	1. No hay mensajes nuevos.
Importancia	Media.

Tabla 35. Especificación del CU-11

CU-12	Contratar un gig
Actor Principal	Cliente o vendedor actuando como cliente.
Descripción	Permite al usuario contratar un gig seleccionado.
Precondiciones	Ninguna.
Flujo Normal	1. El usuario selecciona "Continuar". 2. El sistema redirige a la pasarela de pago.
Postcondiciones	El gig queda contratado y se crea un pedido preeliminar.
Excepciones	1. El pago no se completa correctamente.
Importancia	Alta.

Tabla 36. Especificación del CU-12

CU-13	Pagar un gig
Actor Principal	Cliente o vendedor actuando como cliente.
Descripción	Permite al usuario completar el proceso de pago.
Precondiciones	El usuario debe haber pulsado en el botón continuar en el detalle del gig.
Flujo Normal	1. El cliente completa el formulario de pago. 2. El sistema confirma el pago y actualiza el estado del gig.
Postcondiciones	El pago queda registrado en la plataforma y se crea el pedido definitivo.
Excepciones	1. El pago no se completa correctamente. 2. El usuario cancela la operación una vez pasado el tiempo de espera.
Importancia	Alta.

Tabla 37. Especificación del CU-13

CU-14	Publicar opinión
Actor Principal	Cliente o vendedor actuando como cliente.
Descripción	Permite al usuario escribir una opinión sobre un gig contratado.
Precondiciones	El usuario debe haber contratado el gig.
Flujo Normal	1. El usuario ingresa su opinión y pulsa enviar. 2. El sistema registra y muestra la opinión en el gig correspondiente.
Postcondiciones	La opinión queda registrada y es visible para otros usuarios.
Excepciones	1. El usuario no ha contratado el gig. 2.El usuario vendedor intenta opinar en su propio gig.
Importancia	Media.

Tabla 38. Especificación del CU-14

CU-15	Activar cuenta de vendedor
Actor Principal	Cliente.
Descripción	Permite a un usuario activar la opción de vender gigs en la plataforma.
Precondiciones	El usuario debe haber iniciado sesión.
Flujo Normal	1. El usuario selecciona "Activar cuenta freelance". 2. El sistema habilita la opción de vender gigs.
Postcondiciones	El usuario puede crear y gestionar gigs.
Excepciones	Ninguna.
Importancia	Media.

Tabla 39. Especificación del CU-15

CU-16	Crear gigs
Actor Principal	Vendedor.
Descripción	Permite al usuario vendedor crear nuevos gigs.
Precondiciones	El vendedor debe tener una cuenta freelance activa.
Flujo Normal	1. El vendedor selecciona "Crear gig". 2. El vendedor ingresa los detalles del gig. 3. El sistema publica el gig.
Postcondiciones	El gig queda publicado y disponible para los clientes.
Excepciones	1. Los datos ingresados son inválidos.
Importancia	Alta.

Tabla 40. Especificación del CU-16

CU-17	Ver anuncios activos
Actor Principal	Vendedor.
Descripción	Permite al vendedor ver los gigs que ha publicado y están activos.
Precondiciones	Para visualizar información de un gig activo el vendedor debe haber creado al menos un gig.
Flujo Normal	1. El vendedor accede “Anuncios activos”. 2. El sistema muestra los gigs activos.
Postcondiciones	El vendedor visualiza sus gigs activos.
Excepciones	1. No hay gigs creados.
Importancia	Alta.

Tabla 41. Especificación del CU-17

CU-18	Eliminar gigs
Actor Principal	Vendedor.
Descripción	Permite al usuario vendedor eliminar gigs existentes.
Precondiciones	El vendedor debe tener gigs creados.
Flujo Normal	1. El vendedor selecciona el icono de la papelera del gig que desea eliminar. 2. El sistema elimina el gig.
Postcondiciones	El gig ya no está disponible en la plataforma.
Excepciones	1. El gig está pendiente de entregas.
Importancia	Media.

Tabla 42. Especificación del CU-18

CU-19	Ver trabajos contratados
Actor Principal	Vendedor.
Descripción	Permite al vendedor ver los gigs que han sido contratados por los clientes.
Precondiciones	El vendedor debe tener gigs creados.
Flujo Normal	1. El vendedor accede a “Trabajos contratados". 2. El sistema muestra los gigs contratados y sus detalles.
Postcondiciones	El vendedor puede visualizar los trabajos contratados.
Excepciones	1. No hay gigs contratados.
Importancia	Media.

Tabla 43. Especificación del CU-19

CU-20	Marcar como entregado
Actor Principal	Vendedor.
Descripción	Permite al usuario vendedor marcar un gig contratado como entregado una vez que lo ha finalizado.
Precondiciones	El vendedor debe tener gigs contratados.
Flujo Normal	1. El vendedor selecciona "Marcar como entregado". 2. El sistema actualiza el estado del gig.
Postcondiciones	El gig queda marcado como entregado.
Excepciones	1. El gig no ha sido finalizado.
Importancia	Media.

Tabla 44. Especificación del CU-20

3.3. Requisitos funcionales

En este apartado se presentan los requisitos funcionales de la aplicación, definen las funcionalidades y características necesarias para satisfacer las necesidades del usuario final y garantizar el cumplimiento de los objetivos del sistema.

ID	Descripción
RF_01	El sistema debe permitir el registro de nuevos usuarios.
RF_02	El sistema debe comprobar que el correo electrónico es único.
RF_03	El sistema debe validar las credenciales al iniciar sesión.
RF_04	El sistema generará un token de acceso para poder realizar peticiones al registrarse o iniciar sesión.
RF_05	El sistema permitirá a los usuarios no registrados ver y buscar gigs.
RF_06	El sistema debe permitir buscar gigs por palabras clave y categorías.
RF_07	El sistema debe mostrar un resumen del gig antes de proceder al pago.
RF_08	El sistema debe permitir a los vendedores gestionar sus gigs (añadir, editar o eliminar).
RF_09	El sistema no permitirá que un usuario no registrado dé de alta gigs.
RF_10	El sistema no permitirá que un usuario no registrado contrate gigs.
RF_11	El sistema no permitirá que un usuario no registrado escriba opiniones.
RF_12	El sistema debe permitir al cliente y al vendedor comunicarse mediante mensajes.
RF_13	El sistema no permitirá que un usuario no registrado mantenga conversaciones.
RF_14	El sistema debe permitir a los clientes realizar pagos de gigs contratados.
RF_15	El sistema debe comprobar que la tarjeta con la que se realiza el pago es válida.
RF_16	El sistema debe mostrar un listado de pedidos contratados para los clientes.
RF_17	El sistema debe permitir a los vendedores consultar información de los clientes que contrataron sus gigs.
RF_18	El sistema debe permitir a los usuarios actualizar su perfil.
RF_19	El sistema proporcionará una opción en la sección de Mi Perfil para que el usuario pueda eliminar su cuenta o activar la cuenta de vendedor si no está activada ya.
RF_20	El sistema deberá solicitar confirmación al usuario antes de eliminar su cuenta.
RF_21	Cuando se elimine la cuenta de un usuario (si es posible), también se deberá eliminar toda la información relacionada con él.
RF_22	El sistema debe permitir al cliente publicar reseñas y calificaciones sobre los gigs contratados.
RF_23	El sistema no permitirá que un usuario que no ha contratado un gig concreto opine sobre él.
RF_24	El sistema no permitirá que un usuario vendedor propietario de un gig se opine a sí mismo.
RF_25	El sistema permitirá que un gig contratado por un cliente sea marcado como entregado por el vendedor.
RF_26	El sistema debe ofrecer un historial de mensajes en los chats entre cliente y vendedor.

RF_27	El sistema deberá mostrar todos los mensajes ordenados cronológicamente.
RF_28	El sistema diferenciará los mensajes de los participantes de la conversación.
RF_29	El sistema debe permitir a los usuarios marcar conversaciones como leídas.
RF_30	El sistema debe permitir filtrar gigs por precio, más vendido y reciente.
RF_31	El sistema debe ofrecer una vista previa del gig en su página de detalle antes de la contratación.
RF_32	El sistema debe guardar y mostrar el historial de pedidos para cada cliente.
RF_33	El sistema debe proporcionar un formulario claro y simplificado para el proceso de pago.
RF_34	El sistema debe ofrecer al cliente un desglose detallado del precio total antes del pago.
RF_35	El sistema debe mostrar mensajes de error específicos en caso de fallos en los formularios o acciones erróneas.
RF_36	El sistema debe mostrar mensajes de éxito específicos en acciones como la actualización del perfil, publicación de una opinión o un pago exitoso.
RF_37	El sistema debe permitir a los usuarios cerrar sesión.

Tabla 45. Requisitos Funcionales

3.3.1 Requisitos de información

En este apartado se recogen los requisitos de información de la aplicación, especifican las estructuras de información que el sistema debe gestionar para alcanzar sus objetivos.

ID	Descripción
RI_01	El sistema almacenara información de los usuarios como el nombre, correo, contraseña, etc.
RI_02	El sistema almacenara información de los gigs como el título, descripción, precio, categoría, etc.
RI_03	El sistema almacenara información de los pedidos como el id del Gig, la imagen, titulo, precio, el id del Vendedor, el id del Comprador, etc.
RI_04	El sistema almacenara información de las valoraciones como el id del Gig, el id del Usuario que realiza la opinión, la Estrella(puntuación) y la descripción.
RI_05	El sistema almacenara información mensajes entre usuarios.
RI_06	El sistema almacenara información de las conversaciones.
RI_07	El sistema almacenara información de las categorías.

Tabla 46. Requisitos de información

3.4. Requisitos no funcionales

En este apartado se recogen los requisitos no funcionales de la aplicación, describen las características y condiciones necesarias para garantizar la usabilidad, eficiencia, mantenibilidad, seguridad, disponibilidad y portabilidad del sistema.

3.4.1 Usabilidad

ID	Descripción
RNF_01	La interfaz debe ser intuitiva y fácil de usar para todos los usuarios.
RNF_02	El sistema debe proporcionar mensajes de error claros y útiles para resolver posibles problemas.
RNF_03	En la documentación se incluirán manuales de usuario que expliquen el funcionamiento.

Tabla 47. Requisitos no funcionales de Usabilidad

3.4.2 Eficiencia

ID	Descripción
RNF_04	El sistema debe responder a cualquier acción del usuario en menos de 5 segundos en condiciones normales.
RNF_05	El sistema debe optimizar el uso de recursos en el servidor para garantizar un rendimiento fluido.
RNF_06	El sistema optimizara las imágenes para garantizar un rendimiento fluido.

Tabla 48. Requisitos no funcionales de Eficiencia

3.4.3 Mantenibilidad

ID	Descripción
RNF_07	El código debe seguir buenas prácticas de programación para facilitar el mantenimiento.
RNF_08	La plataforma debe ser capaz de recuperarse de fallos del sistema sin pérdida de datos.

Tabla 49. Requisitos no funcionales de Mantenibilidad

3.4.4 Seguridad

ID	Descripción
RNF_09	Las contraseñas deben cifrarse antes de almacenarse en la base de datos.
RNF_10	El sistema debe autenticar y autorizar basándose en JWT para las operaciones que requieran acceso de usuario registrado.

Tabla 50. Requisitos no funcionales de Seguridad

3.4.5 Disponibilidad

ID	Descripción
RNF_11	El sistema debe garantizar una disponibilidad del 99.9% durante el tiempo de actividad planificado.

Tabla 51. Requisitos no funcionales de Disponibilidad

3.4.6 Portabilidad

ID	Descripción
RNF_12	La plataforma debe ser accesible desde los principales navegadores web (Chrome, Firefox, Safari, Edge).

Tabla 52. Requisitos no funcionales de Portabilidad

3.4.7 Implementación

ID	Descripción
RNF_13	La plataforma debe desarrollarse utilizando tecnologías modernas y escalables como React y Node.js.
RNF_14	La implementación debe seguir una estructura modular para facilitar el desarrollo y mantenimiento.
RNF_15	La plataforma se llevará a cabo utilizando el IDE de Visual Studio

Tabla 53. Requisitos no funcionales de Implementación

Capítulo 4
Diseño del sistema

4.1. Arquitectura Lógica

La arquitectura lógica describe cómo está organizado el sistema desde un nivel conceptual, representa la interacción entre las diferentes capas y componentes de la aplicación. Esto incluye cómo se distribuyen las funciones en la capa de presentación (donde interactúa el usuario), la capa de negocio (donde se procesan las reglas y operaciones), y la capa de datos (donde se almacenan y gestionan los datos). Este diseño asegura que el sistema sea claro, escalable y fácil de mantener, permitiendo visualizar cómo fluye la información entre las distintas partes (Figura 46).

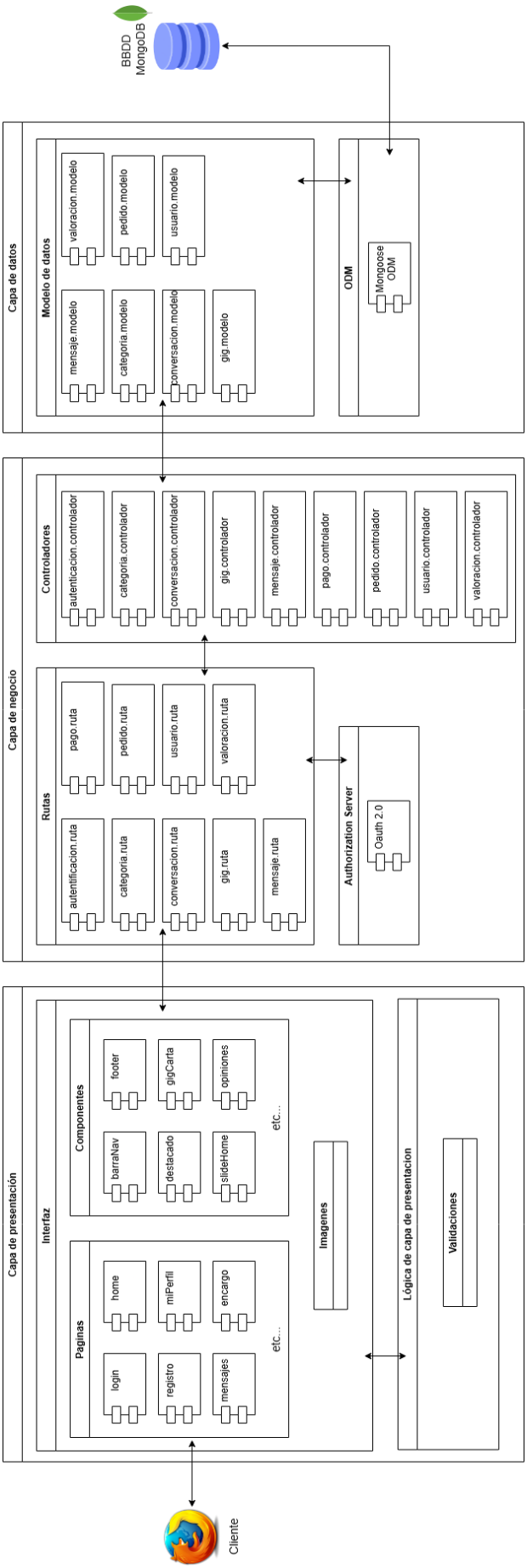


Figura 47. Diagrama de la arquitectura lógica

4.2. Arquitectura Física

La arquitectura física se encarga de definir cómo funciona el sistema a nivel de infraestructura, detallando los elementos de hardware e infraestructura necesarios (Figura 48). Aquí se especifican los servidores que alojan las aplicaciones, la base de datos y otros servicios relacionados, así como la red que conecta estos elementos. Este apartado nos ayuda a visualizar cómo están conectados estos elementos para ofrecer un rendimiento óptimo y garantizar que la aplicación esté disponible para los usuarios en todo momento.

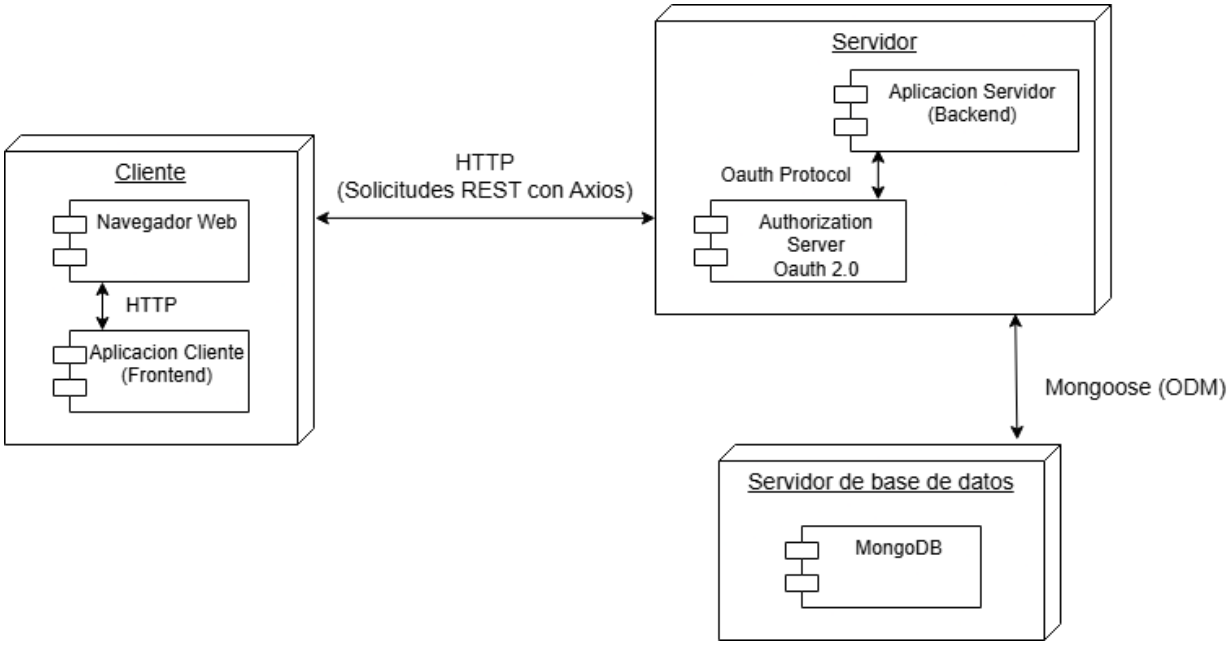


Figura 49. Diagrama de la arquitectura física

4.3. Diagramas de secuencia

Los diagramas de secuencia son herramientas para entender cómo interactúan los diferentes componentes del sistema en procesos específicos. Estos diagramas muestran los mensajes entre los actores y las entidades del sistema, indicando el orden en que ocurren las interacciones. En este apartado, veremos en detalle diagramas que representan situaciones clave, como iniciar sesión y crear un Gig, Esto nos ayudara a visualizar de forma clara y precisa el comportamiento del sistema y su lógica.

4.3.1 Diagrama de secuencia iniciar sesión

Este diagrama detalla cómo un usuario interactúa con el sistema para autenticarse. Representa las solicitudes de credenciales, su verificación por parte del servidor y la respuesta que permite o deniega el acceso, mostrando claramente el flujo de datos y los pasos involucrados (Figura 50).

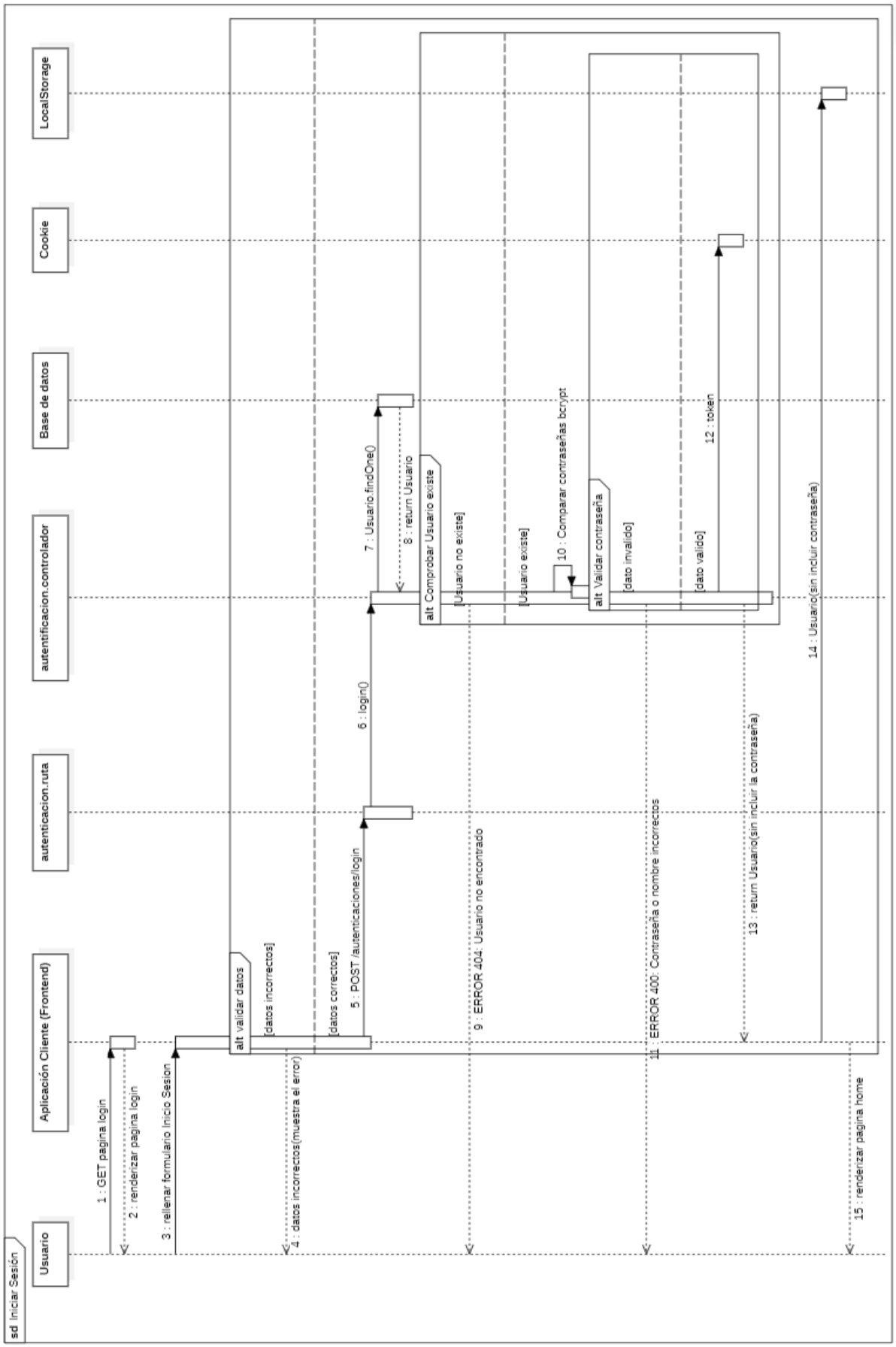


Figura 51. Diagrama de Secuencia Iniciar Sesión

4.3.2 Diagrama de secuencia crear Gig

Este diagrama explica el proceso de creación de un Gig, desde que el usuario introduce los datos en la interfaz, pasando por su validación y procesamiento en el servidor, hasta el momento en que se almacena correctamente en la base de datos (Figura 52).

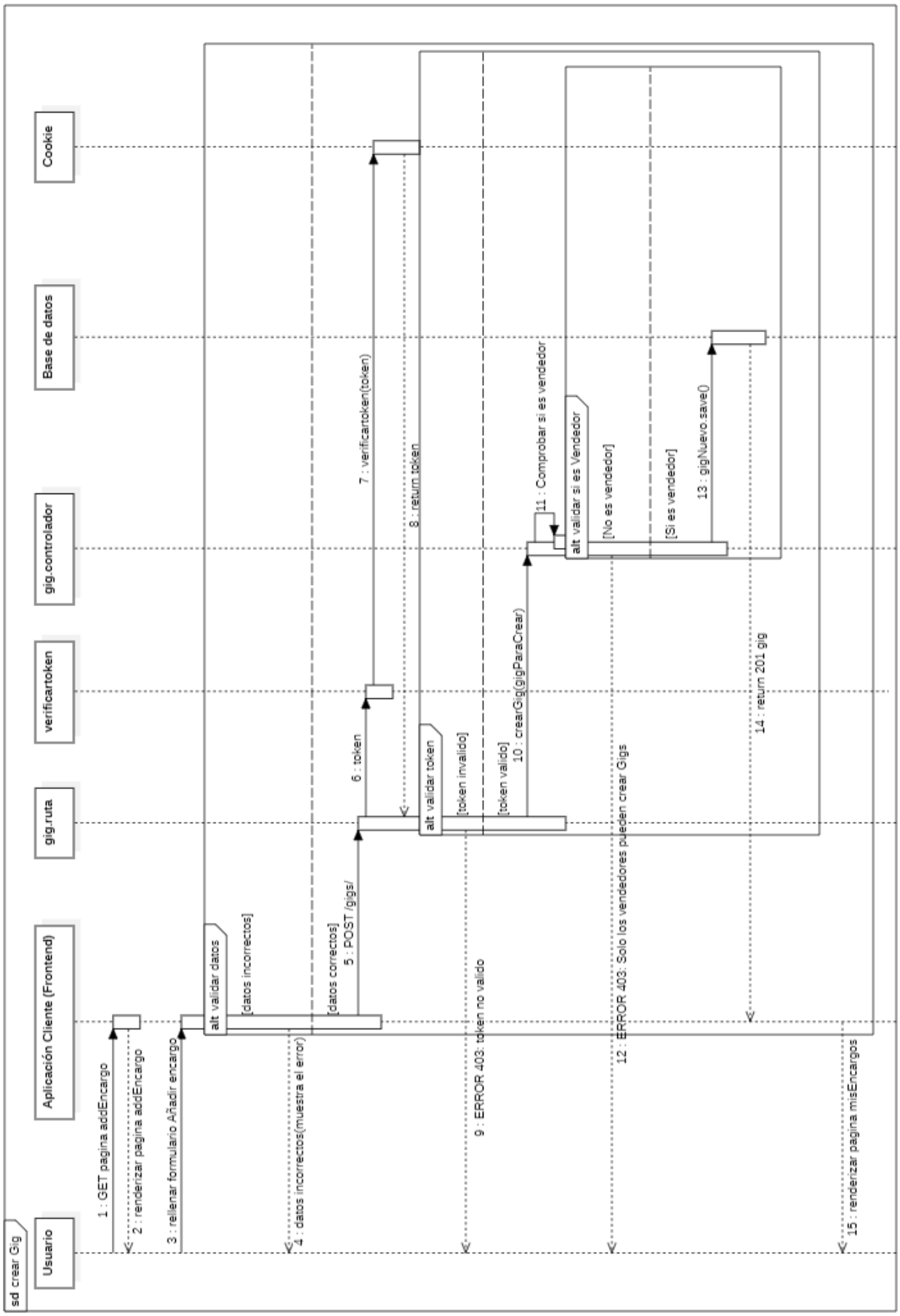


Figura 53. Diagrama de Secuencia Crear Gig

4.4. Modelo de datos

El modelo de datos es una representación esencial del sistema donde se define como se estructuran y organizan los datos para conforme los requerimientos del proyecto. Este modelo asegura que los datos estén estructurados de forma lógica y sean sencillos de manejar.

4.4.1 Modelo Entidad-Relación

El modelo Entidad-Relación es un diagrama que muestra de manera gráfica las entidades principales del sistema y las relaciones entre ellas (Figura 54). Ayuda a visualizar cómo están conectados los datos antes de implementarlos en la base de datos.

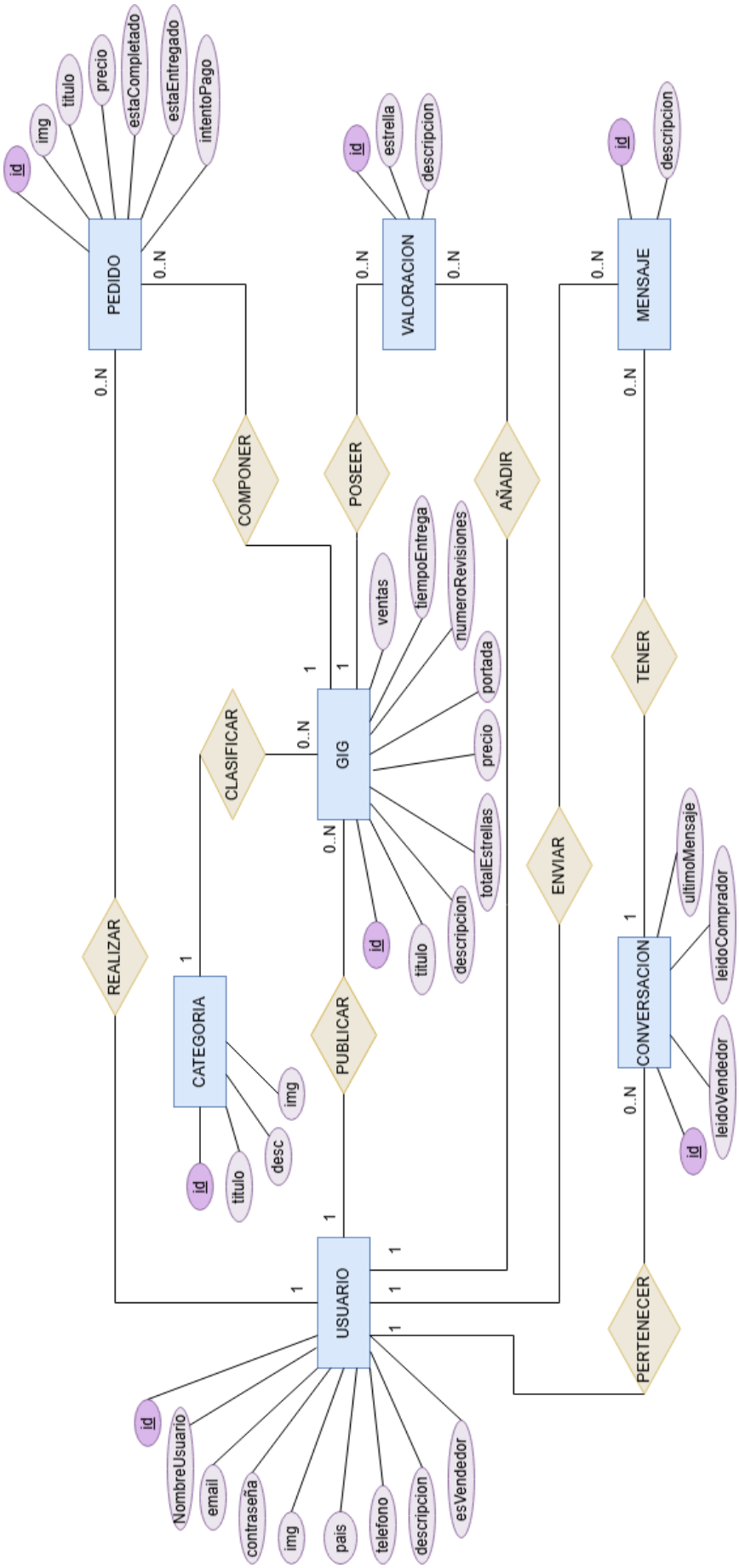


Figura 55. Modelo Entidad-Relación

4.4.2 Diccionario de datos

El diccionario de datos detalla cada elemento del modelo E-R, describe cada atributo de manera precisa indicando su nombre, tipo y características, haciendo que sea más fácil de entender la estructura y los detalles de los datos.

Usuario			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
nombreUsuario	String	No	No
email	String	No	No
contraseña	String	No	No
img	String	Sí	No
pais	String	No	No
telefono	String	Sí	No
descripcion	String	Sí	No
esVendedor	Boolean	No	No

Tabla 54. Diccionario de datos - Usuario

Categoría			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
titulo	String	No	No
desc	String	No	No
img	String	No	No

Tabla 55. Diccionario de datos - Categoría

Gig			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
titulo	String	No	No
descripcion	String	No	No
totalEstrellas	Number	Sí	No
numeroOpiniones	Number	Sí	No
categoria	ObjectId	No	No
idUsuario	ObjectId	No	No
precio	Number	No	No
portada	String	No	No
tiempoEntrega	Number	No	No
numeroRevisiones	Number	No	No
ventas	Number	Sí	No

Tabla 56. Diccionario de datos - Gig

Pedido			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
idGig	ObjectId	No	No
img	String	Sí	No
titulo	String	No	No
precio	Number	No	No
idVendedor	ObjectId	No	No
idComprador	ObjectId	No	No
estaCompletado	Boolean	Sí	No
estaEntregado	Boolean	Sí	No
intentoPago	String	No	No

Tabla 57. Diccionario de datos - Pedido

Conversación			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
idVendedor	ObjectId	No	No
idComprador	ObjectId	No	No
leidoVendedor	Boolean	No	No
leidoComprador	Boolean	No	No
ultimoMensaje	String	Sí	No

Tabla 58. Diccionario de datos - Conversación

Mensaje			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
idConversacion	ObjectId	No	No
idUsuario	ObjectId	No	No
descripcion	String	No	No

Tabla 59. Diccionario de datos - Mensaje

Valoración			
Atributo	**Tipo de dato**	**Nulo**	**PK**
_id	ObjectId	No	Sí
idGig	ObjectId	No	No
idUsuario	ObjectId	No	No
estrella	Number	No	No
descripcion	String	No	No

Tabla 60. Diccionario de datos – Valoración

4.4.3 Diseño lógico

El diseño lógico es la evolución del modelo E-R, donde se traduce el esquema conceptual a un formato que puede implementarse directamente en un sistema de gestión de bases de datos. Aquí se definen las tablas y las relaciones entre ellas. Este paso asegura que todo esté preparado para implementarse de manera eficiente.

USUARIO (_id, nombreUsuario, email, contraseña, img, pais, telefono, descripcion, esVendedor)

CATEGORÍA (_id, titulo, desc, img)

GIG (_id, titulo, descripcion, totalEstrellas, numeroOpiniones, categoria, idUsuario, precio, portada, tiempoEntrega, numeroRevisiones, ventas)
FK: categoria → CATEGORÍA(_id)
FK: idUsuario → USUARIO(_id)

PEDIDO (_id, idGig, img, titulo, precio, idVendedor, idComprador, estaCompletado, estaEntregado, intentoPago)
FK: idGig → GIG(_id)
FK: idVendedor → USUARIO(_id)
FK: idComprador → USUARIO(_id)

CONVERSACIÓN (_id, idVendedor, idComprador, leidoVendedor, leidoComprador, ultimoMensaje)
FK: idVendedor → USUARIO(_id)
FK: idComprador → USUARIO(_id)

MENSAJE (_id, idConversacion, idUsuario, descripcion)
FK: idConversacion → CONVERSACIÓN(_id)
FK: idUsuario → USUARIO(_id)

VALORACIÓN (_id, idGig, idUsuario, estrella, descripcion)
FK: idGig → GIG(_id)
FK: idUsuario → USUARIO(_id)

4.5 Diseño de las Interfaces de Usuario

Este apartado trata sobre cómo los usuarios interactuarán con la aplicación. Se define el diseño de las diferentes pantallas, cómo se navega entre ellas y qué acciones pueden realizar los usuarios. El objetivo es crear una experiencia que sea agradable e intuitiva.

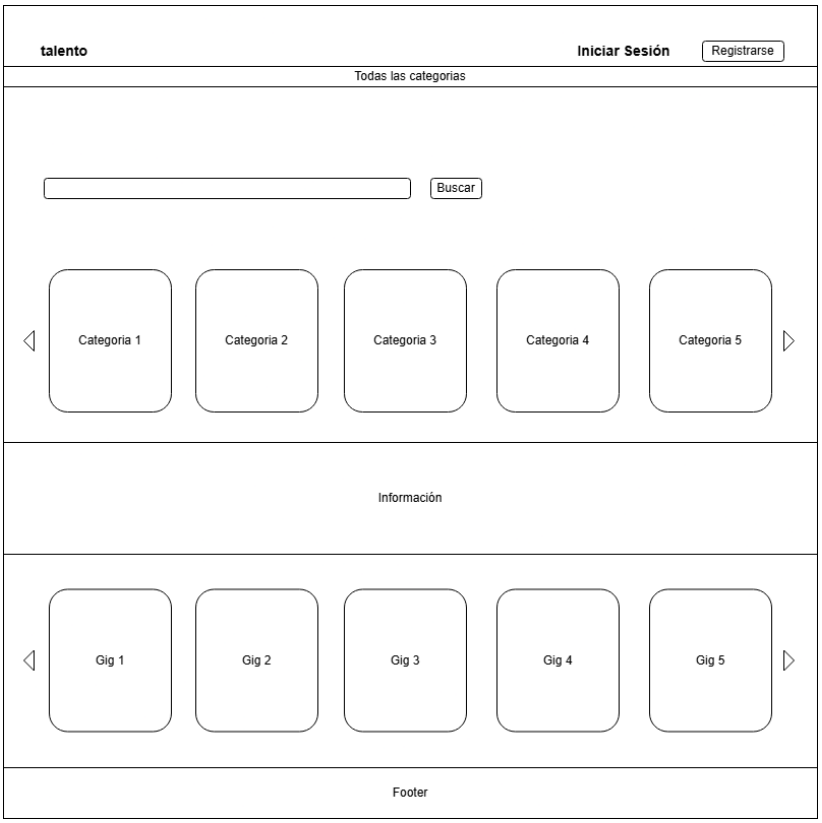
Página principal	
Descripción	Esta página es la primera que se muestra al ingresar a la web.
Activación	Se activa al acceder desde el navegador.
Boceto	
Eventos	1. Click en "Iniciar Sesión": Redirige a la página de inicio de sesión. 2. Click en "Registrarse": Redirige a la página de registro. 3. Click en “Buscar”: Redirige a la página de los gigs filtrados por la búsqueda realizada. 4. Click en “Categoría”: Redirige a la página de los gigs filtrados por la categoría seleccionada. 5. Click en “Gig”: Redirige a la página del gig concreto seleccionado.

Tabla 61. Diseño Interfaz Página principal

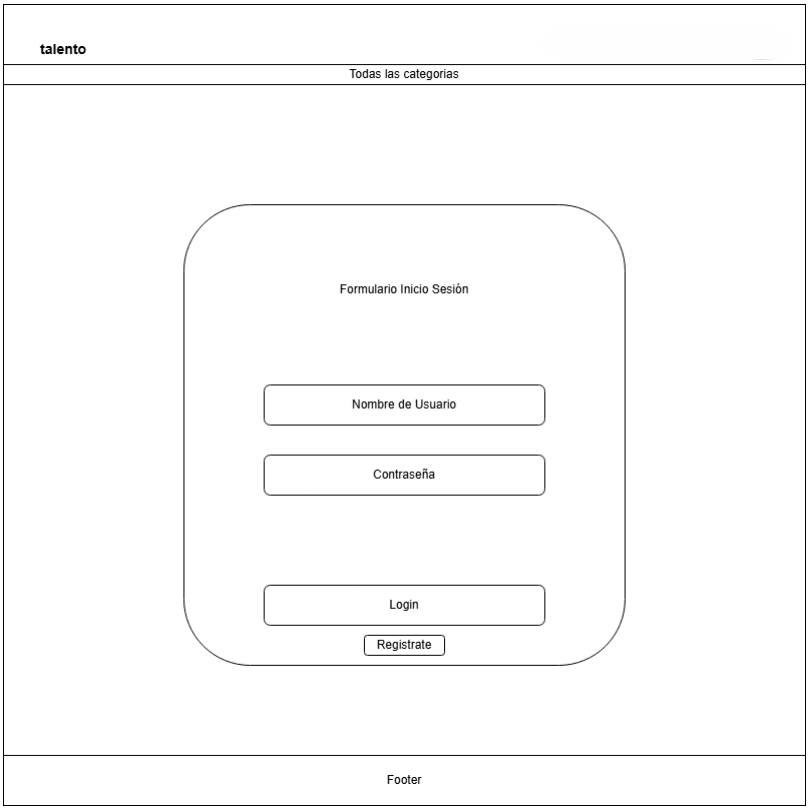
Página Inicio Sesión	
Descripción	Permite a los usuarios iniciar sesión en la plataforma.
Activación	Se activa al hacer clic en el botón "Iniciar Sesión" desde la página principal o desde un gig específico si se intenta realizar una compra sin estar registrado.
Boceto	
Eventos	1. Click en “Login”: Inicia la sesión y redirige a la página principal. 2. Click en “Registrate”: Redirige a la página de registro.

Tabla 62. Diseño Interfaz Página Incio Sesión

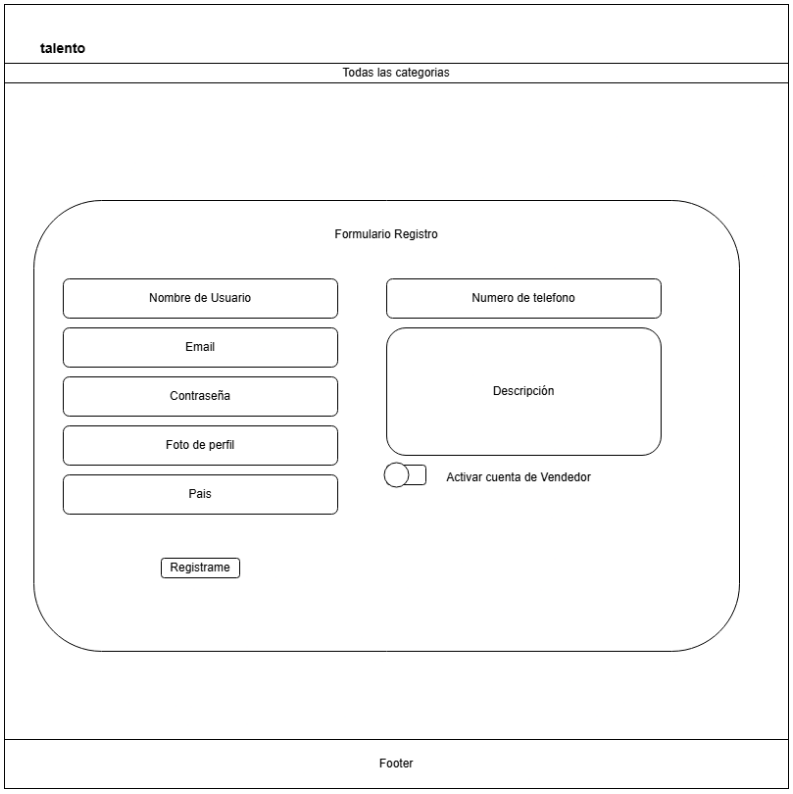
Página Registro	
Descripción	Permite a los usuarios registrarse en la plataforma.
Activación	Se accede al hacer clic en el botón "Registrarse" desde la página principal o desde el formulario de inicio de sesión.
Boceto	
Eventos	1. Click en “Registrarme”: Registra al usuario y redirige a la página principal. 2. Click en el toggle: Activa la cuenta de freelancer.

Tabla 63. Diseño Interfaz Página Registro

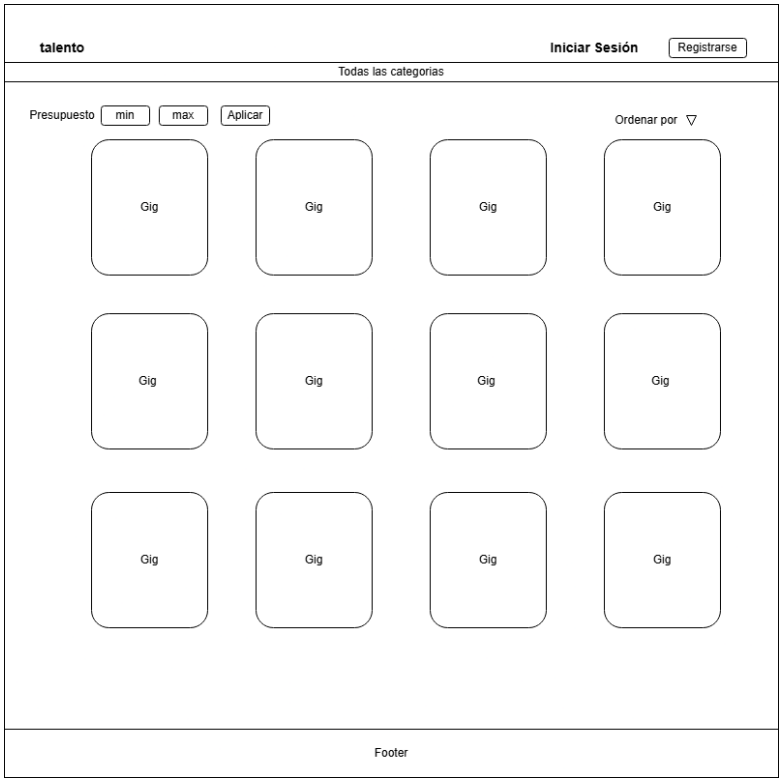
Página Gigs	
Descripción	Permite a los usuarios explorar gigs, ya sea filtrados por categoría o por búsqueda desde la página principal.
Activación	Se accede al realizar una búsqueda o al hacer clic en una categoría en la página principal.
Boceto	
Eventos	1. Click en "Iniciar Sesión": Redirige a la página de inicio de sesión. 2. Click en "Registrarse": Redirige a la página de registro. 3. Click en “Aplicar”: Filtra los gigs entre el rango de precio que le indiquemos. 4. Click en “Ordenar por”: Ordena los gigs por más vendidos o recientes. 5. Click en “Gig”: Redirige a la página del gig concreto seleccionado.

Tabla 64. Diseño Interfaz Página Gigs

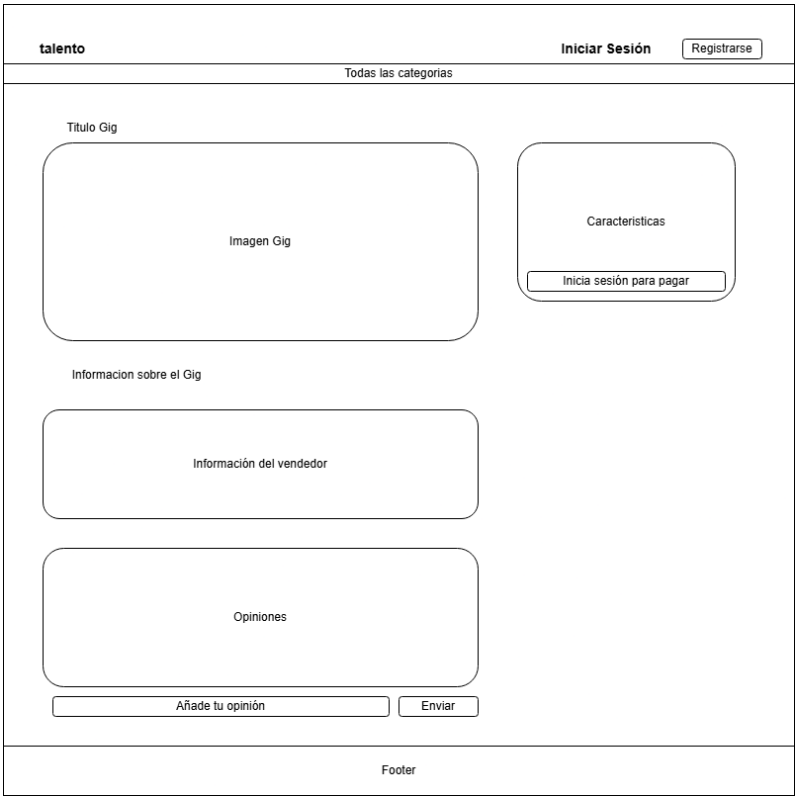
Página Gig detalle	
Descripción	Muestra información detallada de un gig específico.
Activación	Se accede al hacer clic en un gig desde la página de gigs o desde la página principal.
Boceto	
Eventos	1. Click en "Iniciar Sesión": Redirige a la página de inicio de sesión. 2. Click en "Registrarse": Redirige a la página de registro. 3. Click en “Continuar” o “Inicia sesión para pagar”: Si el usuario no está logueado, redirige a la página de inicio de sesión. Si está logueado, redirige al formulario de pago. 4. Click en “Enviar”: Permite publicar una opinión si el usuario ha contratado ese gig.

Tabla 65. Diseño Interfaz Página Gig detalle

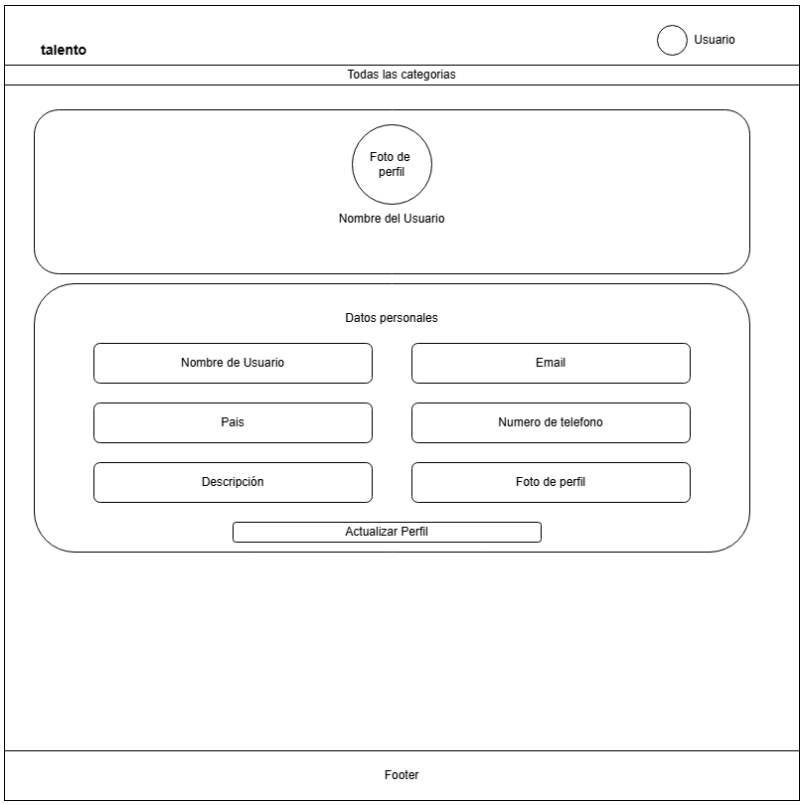
Página Mi perfil	
Descripción	Permite al usuario consultar y modificar su información personal.
Activación	Se accede desde el desplegable del usuario, haciendo clic en "Mi Perfil".
Boceto	
Eventos	1. Click en “Actualizar perfil”: Actualiza los datos modificados.

Tabla 66. Diseño Interfaz Página Mi perfil

Página Anuncios Activos	
Descripción	Permite a los vendedores ver, añadir o eliminar gigs que están ofreciendo.
Activación	Se accede desde el desplegable del usuario vendedor, haciendo clic en "Anuncios Activos".
Boceto	
Eventos	1. Click en “Añadir nuevo encargo”: Redirige a la página añadir encargo. 2. Click en el icono de la papelera: Elimina el gig seleccionado.

Tabla 67. Diseño Interfaz Página Anuncios Activos

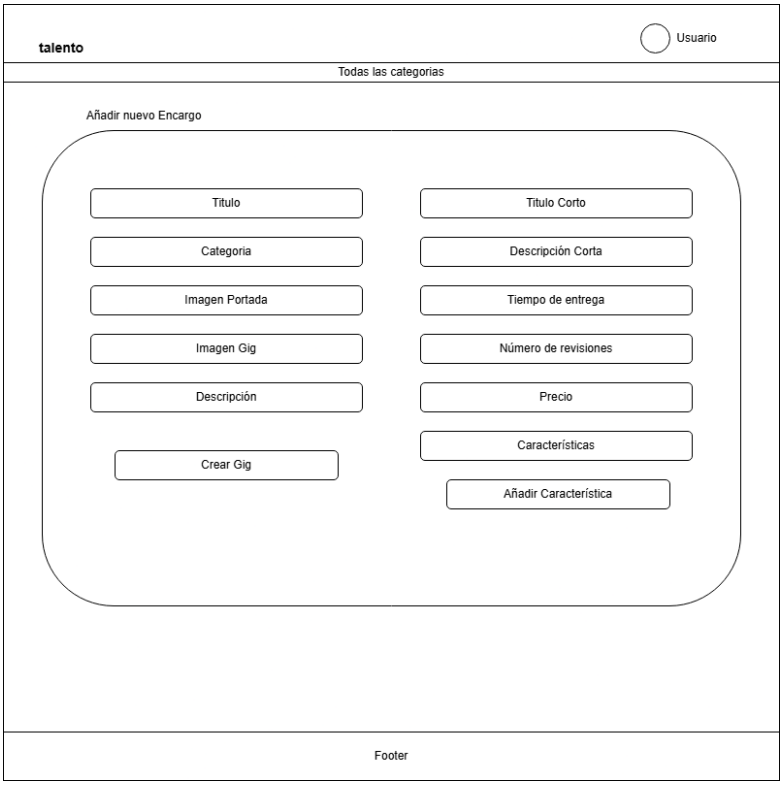
Página Añadir gig	
Descripción	Permite a los vendedores añadir nuevos gigs a sus anuncios activos.
Activación	Se accede al hacer clic en "Añadir Nuevo Encargo" desde la página de anuncios activos.
Boceto	
Eventos	1. Click en “Crear Gig”: Crea un nuevo gig y redirige a la página de anuncios activos. 2. Click en “Añadir Característica”: Añade un campo para incluir una nueva característica.

Tabla 68. Diseño Interfaz Página Añadir gig

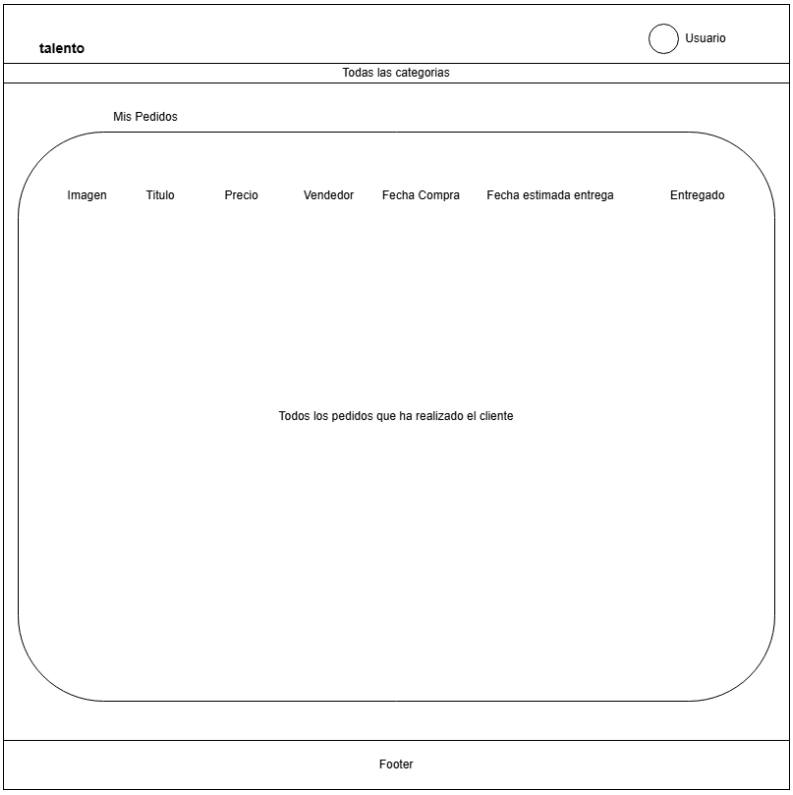
Página Mis pedidos	
Descripción	Permite al usuario visualizar los gigs que ha contratado y consultar información relacionada con ellos.
Activación	Se accede desde el desplegable del usuario, haciendo clic en "Mis Pedidos".
Boceto	
Eventos	No hay eventos para esta página.

Tabla 69. Diseño Interfaz Mis pedidos

Página Trabajos contratados	
Descripción	Permite al usuario vendedor gestionar y visualizar los pedidos que ha recibido.
Activación	Se accede desde el desplegable del usuario vendedor, haciendo clic en "Trabajos Contratados".
Boceto	
Eventos	1. Click en el icono de información: Redirige a la página con información del cliente que realizó el pedido. 2. Click en el icono de contacto: Redirige al chat específico con el cliente. 3. Click en el toggle: Marca el pedido como entregado.

Tabla 70. Diseño Interfaz Página Trabajos contratados

Página Perfil cliente	
Descripción	Permite al usuario vendedor consultar información básica del cliente que realizó un pedido.
Activación	Se accede al hacer clic en el icono de información en la página de "Trabajos Contratados".
Boceto	
Eventos	1. Click en el icono “X”: Redirige de nuevo a la página de "Trabajos Contratados".

Tabla 71. Diseño Interfaz Página Perfil cliente

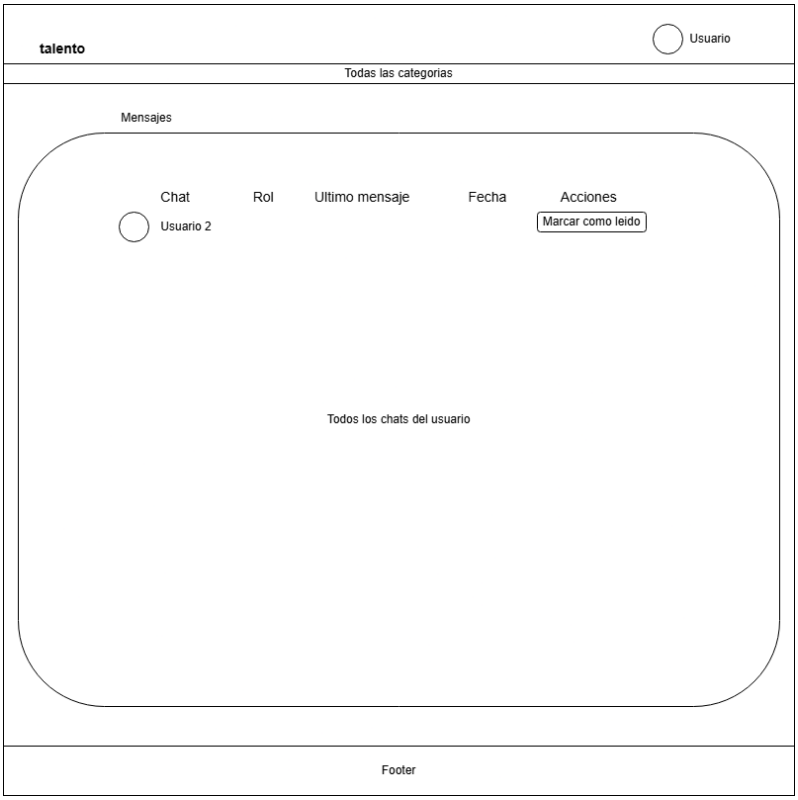
Página Conversaciones	
Descripción	Permite al usuario ver los chats disponibles e interactuar con ellos.
Activación	Se accede desde el desplegable del usuario, haciendo clic en "Mensajes".
Boceto	
Eventos	1. Click en “Marcar como leído”: Indica que el mensaje ha sido leído. 2. Click en la columna de “Ultimo mensaje”: Redirige al chat correspondiente.

Tabla 72. Diseño Interfaz Página Conversaciones

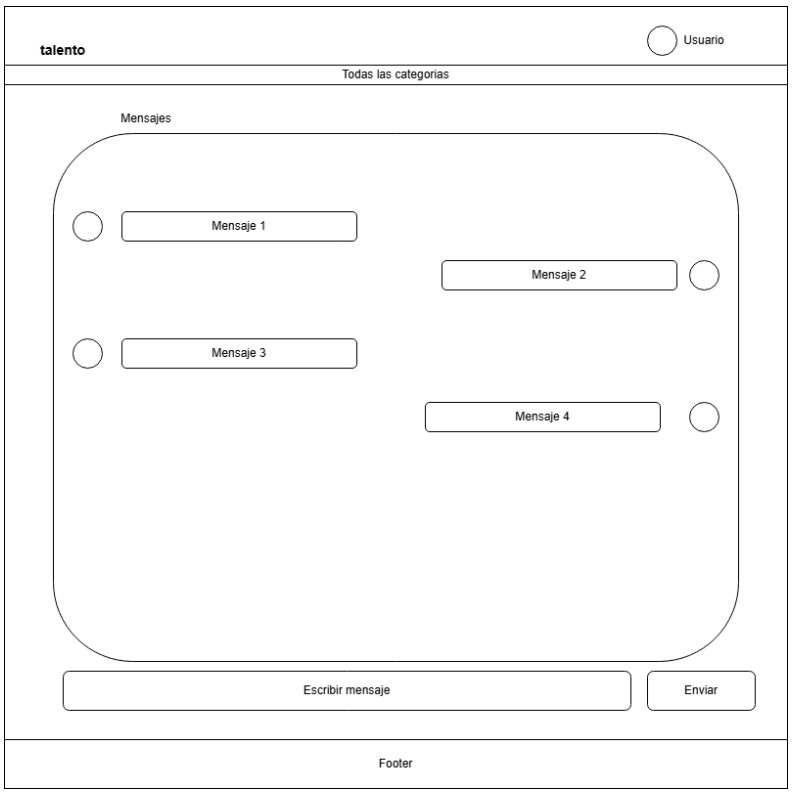
Página Chat concreto	
Descripción	Permite al usuario comunicarse con otros vendedores o clientes mediante el envío de mensajes.
Activación	Se accede desde la página de "Conversaciones" o, en el caso de vendedores, al hacer clic en el icono de contacto desde la página de "Trabajos Contratados".
Boceto	
Eventos	1. Click en “Enviar”: Envía el mensaje que ha escrito el usuario.

Tabla 73. Diseño Interfaz Página Chat concreto

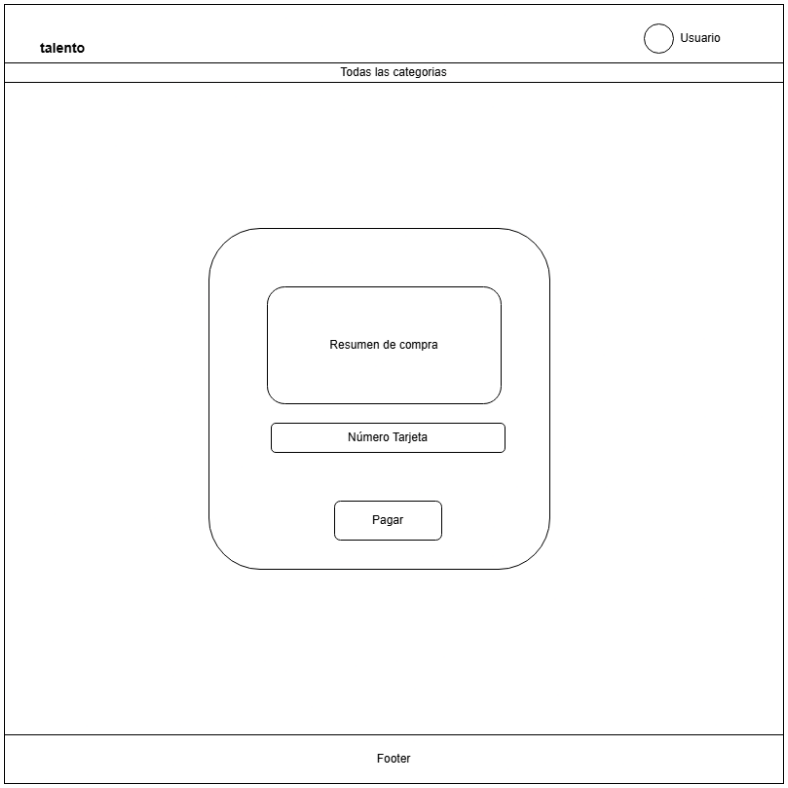
Página Formulario de pago	
Descripción	Permite al usuario realizar el pago de un gig seleccionado, mostrando un resumen del gig contratado.
Activación	Se accede desde la página de un gig concreto, haciendo clic en el botón "Continuar".
Boceto	
Eventos	1. Click en “Pagar”: Si el número de la tarjeta es válido, se realiza el pago y se redirige a la página de "Mis Pedidos".

Tabla 74. Diseño Interfaz Página Formulario de pago

Capítulo 5
Implementación

5.1. Estructura del proyecto

En este apartado, veremos la estructura general del proyecto, compuesta por dos elementos clave: el Backend y el Frontend, diseñados para operar de manera completamente autónoma. A lo largo de este apartado, analizaremos la configuración interna y las funciones específicas de cada uno de estos componentes, explicando su estructura y el rol que desempeña cada uno.

5.1.1 Backend

El backend del proyecto Talento constituye el núcleo de la lógica de negocio, autenticación y persistencia de datos. Diseñado bajo una arquitectura modular y estructurada, siguiendo el patrón MVC (Modelo - Vista - Controlador), donde se separan los modelos, rutas, controladores y middlewares, facilitando la escalabilidad y mantenibilidad del sistema. Este backend ha sido desarrollado utilizando Node.js con el framework Express.js, permitiendo una gestión robusta de rutas, middlewares y respuestas HTTP. La base de datos utilizada es MongoDB, accedida mediante la biblioteca Mongoose, que facilita la definición de esquemas y modelos de datos.

Cada carpeta del backend tiene un papel fundamental en el funcionamiento del sistema, ya sea gestionando el acceso a la base de datos, protegiendo rutas, definiendo lógicas específicas o estructurando los datos que fluyen entre el frontend y el backend. Esta es su estructura (Figura 56):

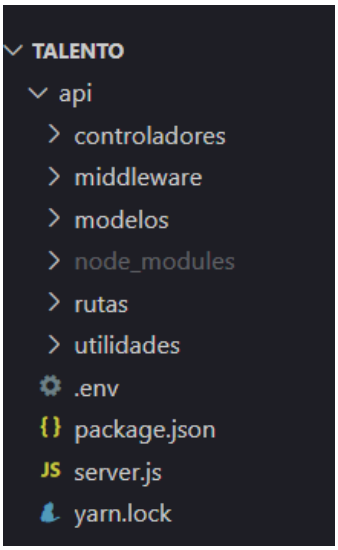


Figura 57. Estructura del Backend

5.1.1.1 Controladores

Esta carpeta contiene la lógica de negocio de cada uno de los recursos principales de la aplicación (Figura 58). Los controladores actúan como intermediarios entre las rutas y los modelos: reciben las peticiones, procesan los datos necesarios, aplican reglas de negocio, interactúan con la base de datos y devuelven respuestas adecuadas.

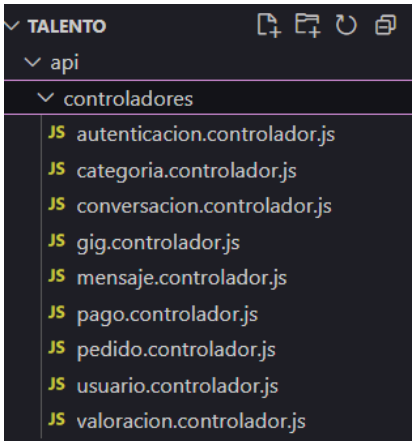


Figura 59. Contenido de la carpeta controladores

A continuación, se explicará la lógica que implementa cada controlador:

- **autenticacion.controlador.js:** Gestiona la autenticación de usuarios. Incorpora funcionalidades como el registro (creación de un nuevo usuario con cifrado de contraseña mediante bcrypt), login (verificación de credenciales y emisión de JWT), y logout (eliminación del token almacenado en cookies).
- **categoria.controlador.js:** Controlador simple pero esencial. Permite la creación y recuperación de categorías, fundamentales para clasificar los gigs y mejorar la navegabilidad del sistema.
- **conversacion.controlador.js:** Administra las conversaciones privadas entre usuarios. Asegura que no se dupliquen conversaciones entre los mismos usuarios siendo el ID de la conversación la concatenación de los IDs de los usuarios que participan en la conversación, permite su recuperación por ID, y permite marcar los mensajes como leídos dependiendo del rol del usuario (vendedor o comprador).
- **gig.controlador.js:** Contiene la lógica relacionada con los "gigs" o servicios ofrecidos por los usuarios. Permite su creación, eliminación (solo por parte del creador siempre y cuando no tenga pedidos pendientes de entrega), consulta individual o listado completo, y filtrado por categoría u otros criterios. Este módulo representa una parte central del proyecto.
- **mensaje.controlador.js:** Controla el sistema de mensajería. Permite enviar mensajes dentro de una conversación y recuperar el histórico. Es esencial para el contacto entre usuarios.
- **pago.controlador.js:** Simula la lógica de procesamiento de pagos. Aunque actualmente no está conectado a una pasarela de pago real, sienta las bases para su futura integración con plataformas como Stripe o PayPal.

- **pedido.controlador.js:** Gestiona los pedidos de gigs. Incluye funcionalidades para crear, eliminar, listar y actualizar pedidos. También permite marcarlos como entregados. Tiene un papel vital en el flujo de contratación de servicios.
- **usuario.controlador.js:** Permite visualizar, actualizar y eliminar perfiles de usuario. Controla el acceso a datos personales y perfiles públicos. Contiene validaciones a la hora de eliminar cuentas, en concreto la de los vendedores, no podrán eliminar su cuenta si tienen encargos pendientes de entrega.
- **valoracion.controlador.js:** Administra la funcionalidad de valoración de gigs. Solo los compradores pueden valorar, lo cual está validado por la lógica del controlador. Aporta confiabilidad al sistema y mejora la experiencia del usuario.

5.1.1.2 Middleware

Esta carpeta contiene funciones que se ejecutan antes de que las peticiones lleguen al controlador (Figura 60). En este caso, se ha implementado un middleware de protección de rutas basado en JWT.

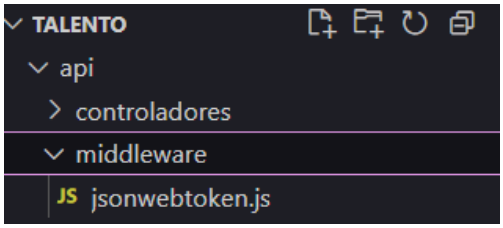


Figura 61. Contenido de la carpeta middleware

- **jsonwebtoken.js:** Este archivo contiene el middleware verificartoken, cuya misión es proteger rutas sensibles que solo deben ser accesibles por usuarios autenticados. Su funcionamiento consiste en:

    - Extraer el token JWT de las cookies de la petición entrante.
    - Verificar la validez del token mediante la clave secreta definida en .env.
    - Si el token es válido, extraer los datos del usuario (ID, rol, etc.) y almacenarlos en req.usuario.
    - Si el token no existe o es inválido, lanzar un error personalizado usando crearError, interrumpiendo el flujo hacia el controlador.

  Este middleware garantiza una autenticación persistente y segura entre cliente y servidor, protegiendo operaciones como la creación de gigs, visualización de pedidos o envío de mensajes.

5.1.1.3 Modelos

Esta carpeta define la estructura de los datos utilizando esquemas de Mongoose (Figura 62). Cada modelo representa una colección en MongoDB, con validaciones y relaciones entre entidades.

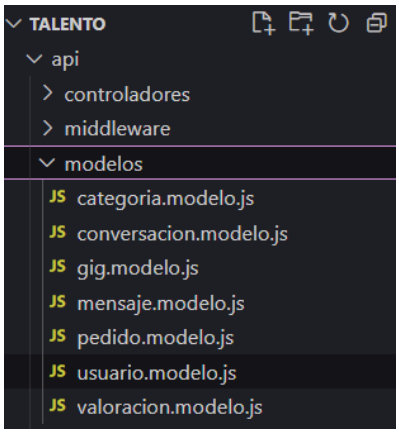


Figura 63. Contenido de la carpeta modelos

- **categoria.modelo.js:** Contiene el esquema para categorías de gigs. Permite agrupar los servicios y mejorar la exploración del usuario.
- **conversacion.modelo.js:** Define las conversaciones entre dos usuarios. Almacena IDs de ambos participantes y estados de lectura.
- **gig.modelo.js:** Representa los gigs o servicios publicados. Incluye referencias al usuario creador y a la categoría, así como información detallada del servicio.
- **mensaje.modelo.js:** Almacena cada mensaje dentro de una conversación. Cada entrada tiene un remitente, destinatario y texto.
- **pedido.modelo.js:** Define los pedidos realizados. Contiene referencias a los gigs, usuarios involucrados, estado del pedido y si ha sido completado o entregado.
- **usuario.modelo.js:** Representa a los usuarios del sistema. Almacena información pública (nombre, imagen) y privada (contraseña cifrada, rol, etc.).
- **valoracion.modelo.js:** Registra las valoraciones de gigs por parte de compradores. Permite dejar un comentario y una calificación numérica.

5.1.1.4 Rutas

Esta carpeta presenta las rutas de la API REST que exponen los diferentes recursos del sistema (Figura 64). Cada archivo agrupa rutas por entidad, y conecta cada una con su controlador correspondiente.

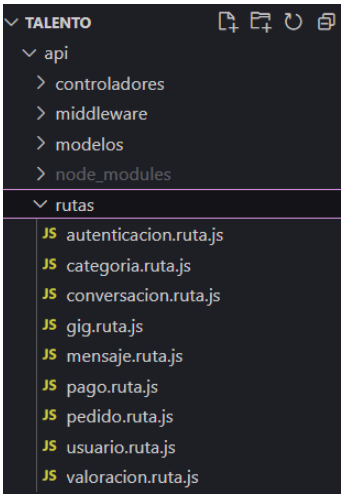


Figura 65. Contenido de la carpeta rutas

En la tabla que se muestra a continuación, se enumerarán todos los endpoints que contiene la aplicación, aportando información detallada de las rutas disponibles y sus funcionalidades:

Ruta	Endpoint	Método HTTP	Descripción
autenticacion.ruta.js	/api/autenticaciones/registro	POST	Registra un nuevo usuario. Cifra la contraseña con bcrypt y lo guarda en la base de datos.
	/api/autenticaciones/login	. POST	Inicia sesión de usuario. Verifica credenciales y devuelve un JWT como cookie
	/api/autenticaciones/logout	POST	Cierra la sesión. Elimina la cookie que contiene el token JWT.
categoria.ruta.js	/api/categoria/	POST	Crea una nueva categoría para los gigs.
	/api/categoria/	GET	Devuelve la lista de todas las categorías registradas en el sistema.
conversacion.ruta.js	/api/conversaciones/	GET	Obtiene todas las conversaciones en las que participa el usuario autenticado.
	/api/conversaciones/unica/:id	GET	Devuelve una conversación específica por su ID.
	/api/conversaciones/	POST	Crea una nueva conversación entre dos usuarios. Previene duplicados.
	/api/conversaciones/:id	PUT	Actualiza el estado de lectura de la conversación.

gig.ruta.js	/api/gigs/	POST	Crea un nuevo gig (servicio) con los datos enviados por el usuario autenticado.
	/api/gigs/:id	DELETE	Elimina un gig si el usuario autenticado es su creador.
	/api/gigs/unico/:id	GET	Devuelve un gig específico por su ID.
	/api/gigs/	GET	Lista todos los gigs disponibles. Puede incluir filtros como categoría o título.
mensaje.ruta.js	/api/mensajes/	POST	Envía un mensaje dentro de una conversación existente.
	/api/mensajes/:id	GET	Obtiene todos los mensajes asociados a una conversación específica.
pago.ruta.js	/api/pago/procesar	POST	Simula el procesamiento de un pago. En una implementación futura se conectaría a pasarelas reales como Stripe o PayPal.
pedido.ruta.js	/api/pedidos/:idGig	POST	Crea un pedido asociado a un gig específico.
	/api/pedidos/:id	DELETE	Elimina un pedido si corresponde al usuario autenticado.
	/api/pedidos/	GET	Devuelve todos los pedidos existentes.
	/api/pedidos/mis-pedidos	GET	Devuelve únicamente los pedidos del usuario autenticado (como comprador o vendedor).
	/api/pedidos/entregado/:id	PUT	Marca un pedido como entregado. Solo el vendedor puede ejecutar esta acción.
usuario.ruta.js	/api/usuarios/:id	DELETE	Elimina el perfil de usuario si coincide con el usuario autenticado.
	/api/usuarios/:id	GET	Devuelve el perfil público de un usuario según su ID.
	/api/usuarios/:id	PUT	Actualiza los datos del usuario autenticado.
valoracion.ruta.js	/api/valoraciones/	POST	Crea una nueva valoración sobre un gig. Solo permitido si el usuario ha contratado el servicio.
	/api/valoraciones/:idGig	GET	Obtiene todas las valoraciones asociadas a un gig específico.

Tabla 75. Rutas

5.1.1.5 Utilidades

Esta carpeta reúne funciones auxiliares para facilitar la gestión de errores (Figura 66).

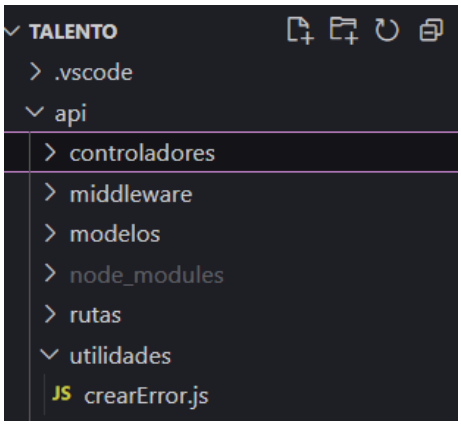


Figura 67. Contenido de la carpeta utilidades

- **crearError.js:** Crea objetos de error personalizados que incluyen códigos de estado HTTP y mensajes descriptivos. Mejora la claridad del manejo de errores en los controladores.

5.1.1.6 Archivos principales

Este apartado engloba archivos de gran importancia que cuelgan directamente de la carpeta /api:

- **server.js:** Este archivo es el punto de entrada del sistema, y orquesta toda la arquitectura. Es EL archivo principal de arranque del backend. Se encarga de:
  - Cargar configuraciones del entorno (dotenv).
  - Conectar a MongoDB.
  - Configurar middlewares globales (express.json, cookieParser, cors).
  - Registrar todas las rutas principales del sistema (/api/...).
  - Implementar un manejador de errores global.
  - Iniciar el servidor escuchando en el puerto 8800.

- **.env:** Contiene variables sensibles como la URL de conexión a la base de datos y la clave secreta para los JWT. Este archivo nunca debe subirse a repositorios.

- **package.json:** Define la configuración del proyecto y sus dependencias. Algunas dependencias destacadas son:
  - express: Framework de servidor.
  - mongoose: ODM para MongoDB.
  - jsonwebtoken, cookie-parser, bcrypt: Para autenticación y seguridad.
  - dotenv: Manejo de variables de entorno.
  - cors: Control de origen cruzado.

- **yarn.lock:** Este archivo es un archivo de bloqueo de dependencias automático generado por Yarn. Asegura que las versiones utilizadas sean consistentes entre diferentes entornos de desarrollo. No contiene lógica negocio.

5.1.2 Frontend

El frontend de Talento representa la parte visual y de interacción directa entre el usuario y la plataforma. Es, por tanto, uno de los componentes más críticos del sistema, ya que constituye el primer punto de contacto entre el usuario final y la lógica del backend. La experiencia de usuario, la presentación de la información, la navegabilidad y la gestión del estado de la interfaz, están orquestados en esta parte de la arquitectura.

La interfaz ha sido desarrollada empleando React, una de las bibliotecas más populares para construir interfaces de usuario dinámicas, junto con Vite, un empaquetador (bundler) que proporciona tiempos de arranque y recarga muy rápidos. Esta elección tecnológica responde a la necesidad de rendimiento, modularidad y escalabilidad que el proyecto demanda.

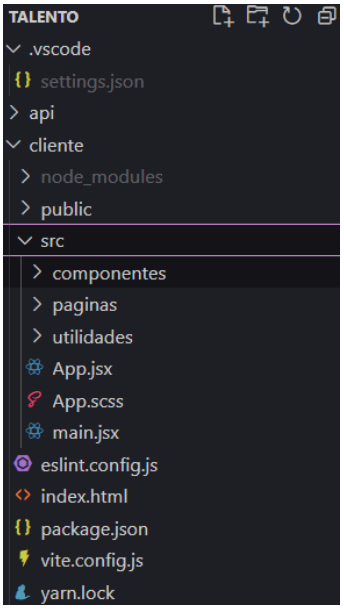


Figura 68. Estructura del Frontend

A continuación, se describirá en los siguientes apartados la estructura de carpetas, archivos clave y lógica funcional del frontend, viendo no solo su composición técnica sino también la relevancia de cada bloque dentro del ecosistema completo de la aplicación (Figura 69).

5.1.2.1 Componentes

Esta carpeta contiene todos los componentes visuales reutilizables de la interfaz (Figura 70). Su diseño modular permite reutilización, mantenimiento independiente y escalabilidad del diseño.

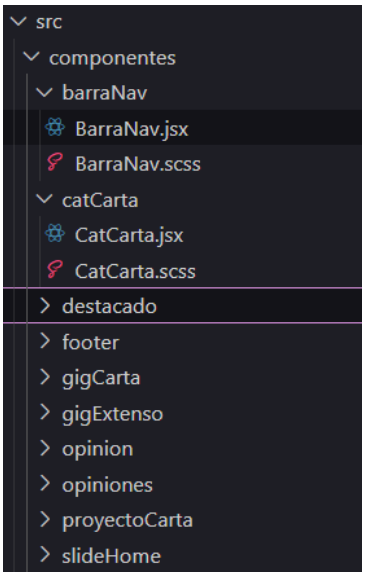


Figura 71. Contenido de la carpeta componentes

5.1.2.2 Páginas

Cada subcarpeta dentro de /paginas representa una vista o sección completa del sistema, asociada directamente a una ruta en el navegador (Figura 72).

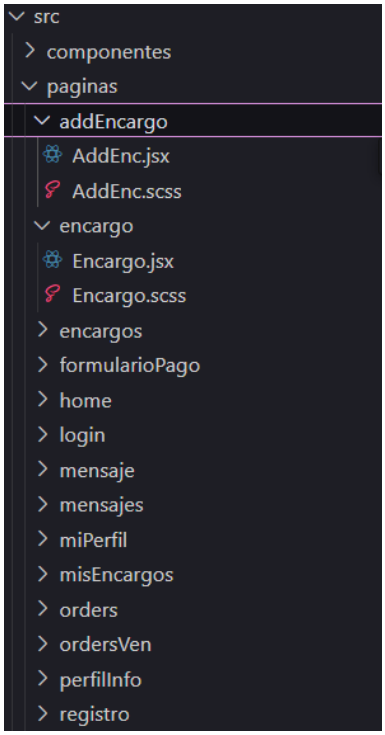


Figura 73. Contenido de la carpeta paginas

Algunas páginas clave son:

- **home/:** Página de inicio con un buscador, información, categorías, encargos destacados, etc.
- **registro/ y login/:** Formularios de autenticación del usuario. Incluyen validaciones tanto en cliente como en servidor.
- **encargos/ y encargo/:** Listado de todos los encargos disponibles y vista de detalle de cada uno, respectivamente.
- **addEncargo/:** Formulario protegido para que un usuario registrado y que sea vendedor pueda crear un nuevo encargo. Requiere validación exhaustiva y subida de imágenes.
- **orders/ y ordersVen/:** Listado y gestión de pedidos realizados o recibidos, dependiendo del rol del usuario.
- **misEncargos/:** Listado de encargos creados por el usuario.
- **formularioPago/:** Página encargada de la integración del sistema de pagos. Crucial para el modelo de negocio.
- **mensajes/ y mensaje/:** Sistema interno de mensajería entre compradores y vendedores.
- **miPerfil/ y perfilInfo/:** Visualización y edición de los datos del perfil propio o de otros usuarios.

Cada página contiene su propia lógica, estilo y subcomponentes, lo que permite independencia en el desarrollo y mejora el mantenimiento a largo plazo.

5.1.2.3 Utilidades

Esta carpeta incluye toda la lógica no visual pero esencial para el correcto funcionamiento del frontend (Figura 74). Destacan especialmente tres subgrupos:

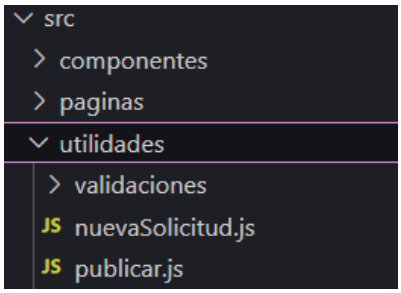


Figura 75. Contenido de la carpeta utilidades

- **Validaciones (/validaciones):** Contiene funciones reutilizables para verificar datos de formularios antes de enviarlos al backend. Esto no solo mejora la experiencia del usuario con retroalimentación inmediata, sino que reduce la carga del servidor.

    - **validarCampoVacio.js:** asegura que un campo no esté vacío.
    - **validarContraseña.js:** aplica reglas de seguridad (mínimo 8 caracteres, mayúsculas, números...).
    - **validarDescripcion.js:** limita la descripción a un máximo de 50 palabras.
    - **validarEmail.js:** comprueba que el formato del correo sea válido.
    - **validarNombreUsuario.js:** filtra caracteres especiales no permitidos.
    - **validarPais.js:** impide números y caracteres inválidos.
    - **validarTelefono.js:** exige formato nacional (9 dígitos).

  Estas validaciones están diseñadas para ser fácilmente integrables en cualquier formulario del sistema.

- **nuevaSolicitud.js:** Configura una instancia personalizada de axios con la base URL del backend (http://localhost:8800/api/) y habilita el envío de cookies. Este archivo centraliza todas las llamadas HTTP para mantener un estilo uniforme y facilitar la autenticación de sesión.

- **publicar.js:** Maneja la subida de archivos a Cloudinary, servicio de almacenamiento en la nube. Esta función encapsula el envío con FormData, optimiza la imagen automáticamente (w_800,q_auto,f_auto) y devuelve la URL accesible públicamente.

5.1.2.4 Archivos principales

- **App.jsx**: Este archivo constituye el corazón de la aplicación React. Es el punto de entrada lógico que define la arquitectura de rutas mediante react-router-dom. Aquí se establece un componente de Layout principal, que actúa como plantilla para todas las páginas de la aplicación. Este layout incluye tres partes fundamentales:

    - BarraNav: la barra de navegación superior, constante en todas las páginas.
    - Outlet: contenedor dinámico que carga el contenido de la ruta correspondiente.
    - Footer: pie de página común a toda la aplicación.

  Además, se configura el proveedor de @tanstack/react-query, una biblioteca para el manejo eficiente del estado del servidor, que permite una gestión óptima de caché, revalidaciones automáticas y sincronización de datos entre cliente y servidor.

- **main.jsx:** Este archivo es el punto de entrada principal de React. Utiliza ReactDOM.createRoot() para montar el componente App dentro del nodo #root

definido en el index.html. Se encapsula dentro de React.StrictMode para ayudar a detectar potenciales problemas en el desarrollo.

- **index.html:** Aunque es un archivo estático, cumple una función crucial: define la estructura básica del documento HTML, incluye fuentes personalizadas desde Google Fonts, y establece el favicon de la aplicación.

- **vite.config.js:** Archivo de configuración de Vite, donde se especifican los plugins utilizados, como @vitejs/plugin-react, que permite el soporte de JSX y otras capacidades propias de React. Esta configuración es fundamental para que la experiencia de desarrollo sea fluida y rápida.

- **package.json:** Contiene todas las dependencias del proyecto, tanto en desarrollo como en producción. Aquí se definen también los scripts básicos como dev, build y preview. Entre las bibliotecas más destacadas utilizadas están:
    - React y React DOM: base del frontend.
    - React Router DOM: enrutamiento.
    - @tanstack/react-query: gestión del estado del servidor.
    - Axios: comunicación con la API.
    - MUI: componentes de interfaz con diseño profesional.
    - SASS: preprocesador CSS.

Capítulo 6
Pruebas

Este capítulo recoge el conjunto de pruebas realizadas sobre Talento con el fin de verificar que el sistema funciona correctamente y responde de forma esperada ante las diferentes situaciones planteadas por los usuarios. Para asegurar la fiabilidad y estabilidad del software, se han realizado tanto pruebas de caja blanca como pruebas de caja negra.

6.1. Pruebas de caja blanca

Las pruebas de caja blanca se han enfocado en la verificación del funcionamiento interno del sistema, analizando directamente el código fuente tanto del backend como del frontend. Aunque en esta fase del proyecto no se ha incorporado un sistema automatizado de testing, como por ejemplo pruebas unitarias o de integración, sí se ha llevado a cabo una revisión del comportamiento del software a través de pruebas manuales orientadas a la lógica y estructura del código.

A continuación, se describen las comprobaciones más significativas realizadas durante este análisis:

- Análisis del flujo de control en los controladores: Se ha inspeccionado la lógica implementada en los controladores de cada uno de los módulos del backend (usuarios, gigs, pedidos, mensajes, etc.). Esta revisión ha permitido confirmar que se realizan las validaciones necesarias, que se manejan correctamente las excepciones y que las respuestas emitidas son coherentes con el estándar HTTP.
- Conectividad y persistencia de datos con MongoDB: Se ha validado que la comunicación entre la API y la base de datos se mantiene estable en todo momento. No se han detectado errores de conexión ni pérdidas de datos en escenarios normales.
- Operaciones CRUD desde el cliente: Se han probado manualmente las acciones de creación, lectura, modificación y eliminación de datos desde el frontend, observando cómo estas interacciones se reflejan correctamente en la base de datos, sin producir estados inconsistentes o inesperados.
- Validación del middleware de autenticación: Se ha comprobado que el sistema de protección de rutas mediante tokens JWT funciona adecuadamente. Las rutas privadas no pueden ser accedidas por usuarios no autenticados.
- Verificación de utilidades de validación en el frontend: En el cliente, se han testeado manualmente las funciones JavaScript que validan campos como el correo electrónico, la contraseña, el número de teléfono o las descripciones. Estas funciones devuelven los mensajes correctos cuando se introducen datos erróneos, cumpliendo su propósito preventivo.

- Simulación de errores y gestión de respuestas: Se han generado errores de manera controlada (parámetros incompletos, rutas no válidas, identificadores inexistentes) para evaluar la robustez de las respuestas. El sistema responde con mensajes estructurados en formato JSON y códigos HTTP adecuados como 400, 403, 404 o 500, según corresponda.
- Seguridad y privacidad de la información: Se ha verificado que no se exponen credenciales en el lado del cliente. Además, se ha comprobado que las contraseñas se encriptan correctamente usando bcrypt durante los procesos de registro e inicio de sesión, garantizando la confidencialidad de los datos sensibles.

6.2. Pruebas de caja negra

Las pruebas de caja negra se han centrado en evaluar la funcionalidad de la aplicación desde la perspectiva del usuario, sin tener en cuenta el código fuente. Se ha analizado si la aplicación responde correctamente a diversas entradas, acciones y situaciones, en cumplimiento de los requisitos funcionales definidos.

La siguiente tabla recoge una selección de pruebas clave realizadas sobre el sistema, cada una asociada con uno o varios requisitos funcionales (RF):

PCN-01 Registro de usuario	
Finalidad	Verificar que el sistema permite registrar nuevos usuarios correctamente.
Prerrequisitos	El usuario no debe estar registrado previamente.
Datos o acciones de entrada	Formulario de registro válido.
Resultado esperado según los RFs	El sistema crea una nueva cuenta y genera un token de acceso. (RF_01, RF_04)
Resultado obtenido	Correcto.

Tabla 76. PCN-01 Registro de usuario

PCN-02 Registro con correo ya existente	
Finalidad	Comprobar que el sistema detecta correos duplicados en el registro.
Prerrequisitos	El correo ya debe estar registrado por otro usuario.
Datos o acciones de entrada	Formulario de registro válido, pero con email ya registrado.
Resultado esperado según los RFs	Mensaje de error indicando que el correo ya está en uso. (RF_02)
Resultado obtenido	Correcto.

Tabla 77. PCN-02 Registro con correo ya existente

PCN-03 Inicio de sesión correcto	
Finalidad	Validar que el sistema permite iniciar sesión con credenciales válidas.
Prerrequisitos	Usuario registrado.
Datos o acciones de entrada	Email registrado y contraseña correcta.
Resultado esperado según los RFs	Inicio de sesión exitoso y generación de token. (RF_03, RF_04)
Resultado obtenido	Correcto.

Tabla 78. PCN-03 Inicio de sesión correcto

PCN-04 Inicio de sesión incorrecto	
Finalidad	Verificar el comportamiento ante credenciales incorrectas.
Prerrequisitos	Usuario registrado.
Datos o acciones de entrada	Email registrado y contraseña incorrecta.
Resultado esperado según los RFs	Mensaje de error indicando contraseña incorrecta. (RF_03)
Resultado obtenido	Correcto.

Tabla 79. PCN-04 Inicio de sesión incorrecto

PCN-05 Visualizar gigs sin registrarse	
Finalidad	Comprobar que los usuarios anónimos pueden ver gigs.
Prerrequisitos	No estar autenticado.
Datos o acciones de entrada	Acceder a la sección de gigs desde la Home.
Resultado esperado según los RFs	El sistema muestra gigs disponibles. (RF_05)
Resultado obtenido	Correcto.

Tabla 80. PCN-05 Visualizar gigs sin registrarse

PCN-06 Buscar gigs por palabra clave	
Finalidad	Verificar la funcionalidad de búsqueda de gigs.
Prerrequisitos	Gigs existentes con palabras clave reconocibles.
Datos o acciones de entrada	Palabra clave "desarrollo".
Resultado esperado según los RFs	Se muestran gigs relacionados. (RF_06)
Resultado obtenido	Correcto.

Tabla 81. PCN-06 Buscar gigs por palabra clave

PCN-07 Publicar gig como vendedor	
Finalidad	Asegurar que los vendedores autenticados pueden crear gigs.
Prerrequisitos	Usuario autenticado con rol de vendedor.
Datos o acciones de entrada	Formulario completo con título, descripción, precio, etc.
Resultado esperado según los RFs	El nuevo gig aparece en el listado. (RF_08)
Resultado obtenido	Correcto.

Tabla 82. PCN-07 Publicar gig como vendedor

PCN-08 Publicar gig sin autenticación	
Finalidad	Comprobar que los usuarios no registrados no pueden crear gigs.
Prerrequisitos	No estar autenticado.
Datos o acciones de entrada	Acceder a “Crear encargo” a través de la URL.
Resultado esperado según los RFs	Redirección a login. (RF_09)
Resultado obtenido	Correcto.

Tabla 83. PCN-08 Publicar gig sin autenticación

PCN-09 Comprar gig con resumen previo	
Finalidad	Comprobar que se muestra un resumen del gig antes del pago.
Prerrequisitos	Usuario autenticado, gig disponible.
Datos o acciones de entrada	Acceder al gig y hacer clic en “Contratar”.
Resultado esperado según los RFs	Se muestra resumen del pedido y formulario de pago. (RF_07, RF_33, RF_34)
Resultado obtenido	Correcto.

Tabla 84. PCN-09 Comprar gig con resumen previo

PCN-10 Comprar gig sin autenticarse	
Finalidad	Comprobar que los usuarios anónimos no pueden contratar gigs.
Prerrequisitos	No estar autenticado.
Datos o acciones de entrada	Clic en "Contratar" desde la vista de un gig.
Resultado esperado según los RFs	Redirección al login. (RF_10)
Resultado obtenido	Correcto.

Tabla 85. PCN-10 Comprar gig sin autenticarse

PCN-11 Enviar mensaje autenticado	
Finalidad	Validar que el sistema permite mensajes entre cliente y vendedor.
Prerrequisitos	Cliente y vendedor autenticados.
Datos o acciones de entrada	Texto del mensaje.
Resultado esperado según los RFs	El mensaje se guarda y aparece en la conversación. (RF_12, RF_26)
Resultado obtenido	Correcto.

Tabla 86. PCN-11 Enviar mensaje autenticado

PCN-12 Enviar mensaje sin estar autenticado	
Finalidad	Comprobar que no se permite chatear sin login.
Prerrequisitos	No estar autenticado.
Datos o acciones de entrada	Intento de abrir una conversación a través de la URL.
Resultado esperado según los RFs	Redirección a login. (RF_13)
Resultado obtenido	Correcto.

Tabla 87. PCN-12 Enviar mensaje sin estar autenticado

PCN-13 Pago con tarjeta válida	
Finalidad	Verificar que se puede realizar un pago con tarjeta válida.
Prerrequisitos	Usuario autenticado, tarjeta válida.
Datos o acciones de entrada	Tarjeta valida "1111 1111 1111 1111".
Resultado esperado según los RFs	Confirmación de pago exitoso. (RF_14, RF_15)
Resultado obtenido	Correcto.

Tabla 88. PCN-13 Pago con tarjeta válida

PCN-14 Pago con tarjeta inválida	
Finalidad	Verificar que no se puede realizar un pago con tarjeta inválida.
Prerrequisitos	Usuario autenticado, tarjeta válida.
Datos o acciones de entrada	Tarjeta invalida, cualquiera distinta de "1111 1111 1111 1111".
Resultado esperado según los RFs	Confirmación de pago erróneo. (RF_14, RF_15)
Resultado obtenido	Correcto.

Tabla 89. PCN-14 Pago con tarjeta inválida

PCN-15 Ver pedidos (cliente)	
Finalidad	Validar que los clientes pueden ver su historial de pedidos.
Prerrequisitos	Cliente autenticado con pedidos realizados.
Datos o acciones de entrada	Acceder a "Mis pedidos".
Resultado esperado según los RFs	Aparece el listado de gigs contratados. (RF_16, RF_32)
Resultado obtenido	Correcto.

Tabla 90. PCN-15 Ver pedidos (cliente)

PCN-16 Ver trabajos contratados (vendedor)	
Finalidad	Verificar que los vendedores ven información de sus clientes.
Prerrequisitos	Vendedor autenticado con gigs contratados.
Datos o acciones de entrada	Acceder a "Trabajos contratados".
Resultado esperado según los RFs	Se muestra la información de los pedidos que le han contratado. (RF_17)
Resultado obtenido	Correcto.

Tabla 91. PCN-16 Ver trabajos contratados (vendedor)

PCN-17 Actualizar perfil	
Finalidad	Verificar que se puede actualizar el nombre u otros datos.
Prerrequisitos	Usuario autenticado.
Datos o acciones de entrada	Nuevo nombre, guardar cambios.
Resultado esperado según los RFs	Perfil actualizado con mensaje de éxito. (RF_18, RF_36)
Resultado obtenido	Correcto.

Tabla 92. PCN-17 Actualizar perfil

PCN-18 Eliminar cuenta desde el perfil	
Finalidad	Verificar el proceso de eliminación de cuenta.
Prerrequisitos	Usuario autenticado.
Datos o acciones de entrada	Clic en "Eliminar cuenta", confirmación.
Resultado esperado según los RFs	Cuenta eliminada y datos borrados. (RF_19, RF_20, RF_21)
Resultado obtenido	Correcto.

Tabla 93. PCN-18 Eliminar cuenta desde el perfil

PCN-19 Eliminar cuenta de vendedor con pedidos pendientes de entrega	
Finalidad	Verificar que el vendedor no puede eliminar la cuenta si tiene pedidos pendientes de entrega.
Prerrequisitos	Usuario vendedor autenticado con pedidos por entregar.
Datos o acciones de entrada	Clic en "Eliminar cuenta", confirmación.
Resultado esperado según los RFs	Cuenta no eliminada y mensaje de error. (RF_21)
Resultado obtenido	Correcto.

Tabla 94. PCN-19 Eliminar cuenta de vendedor con pedidos pendientes de entrega

PCN-20 Dejar opinión tras contratación	
Finalidad	Validar que un cliente puede dejar una reseña.
Prerrequisitos	Usuario autenticado con gig finalizado.
Datos o acciones de entrada	Texto y puntuación.
Resultado esperado según los RFs	Opinión publicada. (RF_22)
Resultado obtenido	Correcto.

Tabla 95. PCN-20 Dejar opinión tras contratación

PCN-21 Intentar opinar sin contratar	
Finalidad	Comprobar que no se puede opinar sin haber contratado.
Prerrequisitos	Usuario autenticado, sin relación con el gig.
Datos o acciones de entrada	Intento de dejar opinión.
Resultado esperado según los RFs	Acción denegada y mensaje de error. (RF_23)
Resultado obtenido	Correcto.

Tabla 96. PCN-21 Intentar opinar sin contratar

PCN-22 Marcar encargo como entregado	
Finalidad	Validar que el vendedor puede marcar como entregado.
Prerrequisitos	Pedido activo sin entregar.
Datos o acciones de entrada	Clic en "Marcar como entregado".
Resultado esperado según los RFs	Estado del pedido actualizado. (RF_25)
Resultado obtenido	Correcto.

Tabla 97. PCN-22 Marcar encargo como entregado

PCN-23 Diferenciar mensajes por usuario	
Finalidad	Verificar que se distinguen los mensajes por emisor.
Prerrequisitos	Conversación activa con mensajes.
Datos o acciones de entrada	Visualización del chat.
Resultado esperado según los RFs	Visual diferencia clara entre remitentes. (RF_28)
Resultado obtenido	Correcto.

Tabla 98. PCN-23 Diferenciar mensajes por usuario

PCN-24 Mostrar mensajes ordenados	
Finalidad	Validar que los mensajes se ordenan cronológicamente.
Prerrequisitos	Conversación activa con mensajes.
Datos o acciones de entrada	Visualización del chat.
Resultado esperado según los RFs	Mensajes ordenados por fecha. (RF_27)
Resultado obtenido	Correcto.

Tabla 99. PCN-24 Mostrar mensajes ordenados

PCN-25 Filtrar gigs	
Finalidad	Verificar que los filtros por precio, ventas o fecha funcionan.
Prerrequisitos	Varios gigs publicados.
Datos o acciones de entrada	Aplicar filtro desde la interfaz.
Resultado esperado según los RFs	Lista actualizada correctamente. (RF_30)
Resultado obtenido	Correcto.

Tabla 100. PCN-25 Filtrar gigs

PCN-26 Vista previa de gig antes del pago	
Finalidad	Comprobar que se muestra una vista previa completa.
Prerrequisitos	Gig activo.
Datos de o acciones entradas	Acceder a la página del gig.
Resultado esperado según los RFs	Se muestra toda la información del gig antes de contratar. (RF_31)
Resultado obtenido	Correcto.

Tabla 101. PCN-26 Vista previa de gig antes del pago

PCN-27 Mensajes de error y éxito	
Finalidad	Validar la retroalimentación al usuario.
Prerrequisitos	Realizar acciones válidas y erróneas en formularios de forma general.
Datos o acciones de entrada	Formulario mal completado y luego correcto.
Resultado esperado según los RFs	Mensajes específicos de éxito o error según resultado. (RF_35, RF_36)
Resultado obtenido	Correcto.

Tabla 102. PCN-27 Mensajes de error y éxito

PCN-28 Cerrar sesión	
Finalidad	Verificar que se puede cerrar sesión correctamente.
Prerrequisitos	Usuario autenticado.
Datos o acciones de entrada	Clic en “Cerrar sesión” desde el menú.
Resultado esperado según los RFs	Sesión finalizada, redirección a la página Home. (RF_37)
Resultado obtenido	Correcto.

Tabla 103. PCN-28 Cerrar sesión

Capítulo 7
Manuales

En este capítulo se recogen los manuales para Talento. Está dividido en dos apartados: Manual de despliegue y Manual de Usuario.

7.1. Manual de despliegue

En este primer apartado se detallan los pasos necesarios para realizar una correcta instalación y ejecución local de Talento. El propósito de este manual es facilitar la posibilidad de instalar, configurar y ejecutar el sistema en un entorno propio, utilizando herramientas y tecnologías estándar. A continuación, se describen paso a paso los requisitos previos, el proceso de clonación del repositorio, la instalación de dependencias, la configuración del entorno y la ejecución de la aplicación tanto en su parte cliente como servidor.

- **Requisitos previos:** Antes de proceder con la instalación, es necesario asegurarse de tener el siguiente software instalado en el equipo:

  - Node.js
  - Yarn (gestor de paquetes alternativo a npm)
  - Git
  - Visual Studio Code u otro editor de código compatible

  Todas estas herramientas pueden descargarse desde sus sitios web oficiales.

- **Clonación del repositorio:** En primer lugar, se debe crear una carpeta de trabajo en el equipo local y clonar el repositorio del proyecto desde GitHub. Para ello, abrir la terminal o consola de comandos y ejecutar:

  ```
  $ cd ruta/deseada
  $ git clone https://github.com/jesussanfrutos/Talento.git
  $ cd Talento
  ```

- **Instalación de dependencias:** Una vez clonado el proyecto, es necesario instalar las dependencias tanto del backend como del frontend.

  ```
  $ cd api
  $ yarn install
  $ cd cliente
  $ yarn install
  ```

- Ejecución de la aplicación: Una vez instaladas las dependencias y configurado el entorno, se puede proceder al despliegue local de la aplicación:

    - Iniciar el backend: Desde la carpeta api, ejecutar el siguiente comando:

      ```
      $ yarn start
      ```

    - Iniciar el frontend: En una nueva terminal, acceder a la carpeta cliente y ejecutar:

      ```
      $ yarn run dev
      ```

    - Por último, se deberá introducir la siguiente URL en el navegador para acceder a la aplicación: `http://localhost:5173`

7.2. Manual de Usuario

Este segundo apartado tiene como objetivo ofrecer una guía completa y estructurada sobre el uso de la plataforma Talento.

La plataforma contempla distintos perfiles de usuario, y en función del estado y rol de cada uno, se habilitan diferentes funcionalidades. A lo largo del presente manual, se detallará el flujo de uso y las opciones disponibles según los tres casos posibles:

- **Usuario no registrado o no autenticado:** Puede navegar por la plataforma, visualizar gigs y explorar categorías, pero no podrá contratar servicios ni interactuar con vendedores hasta iniciar sesión.

- **Usuario registrado con rol de cliente:** Puede contratar gigs, modificar sus propios datos, comunicarse con vendedores, visualizar sus pedidos y dejar valoraciones.

- **Usuario registrado con rol de vendedor (freelancer):** Además de todas las funcionalidades de un cliente, puede crear y gestionar sus propios gigs, visualizar los encargos recibidos y realizar acciones sobre ellos.

A lo largo de este documento se explicarán paso a paso las distintas pantallas y acciones disponibles para cada tipo de usuario, incluyendo capturas ilustrativas y detalles sobre el comportamiento de la plataforma según distintos escenarios.

Pantalla de inicio:

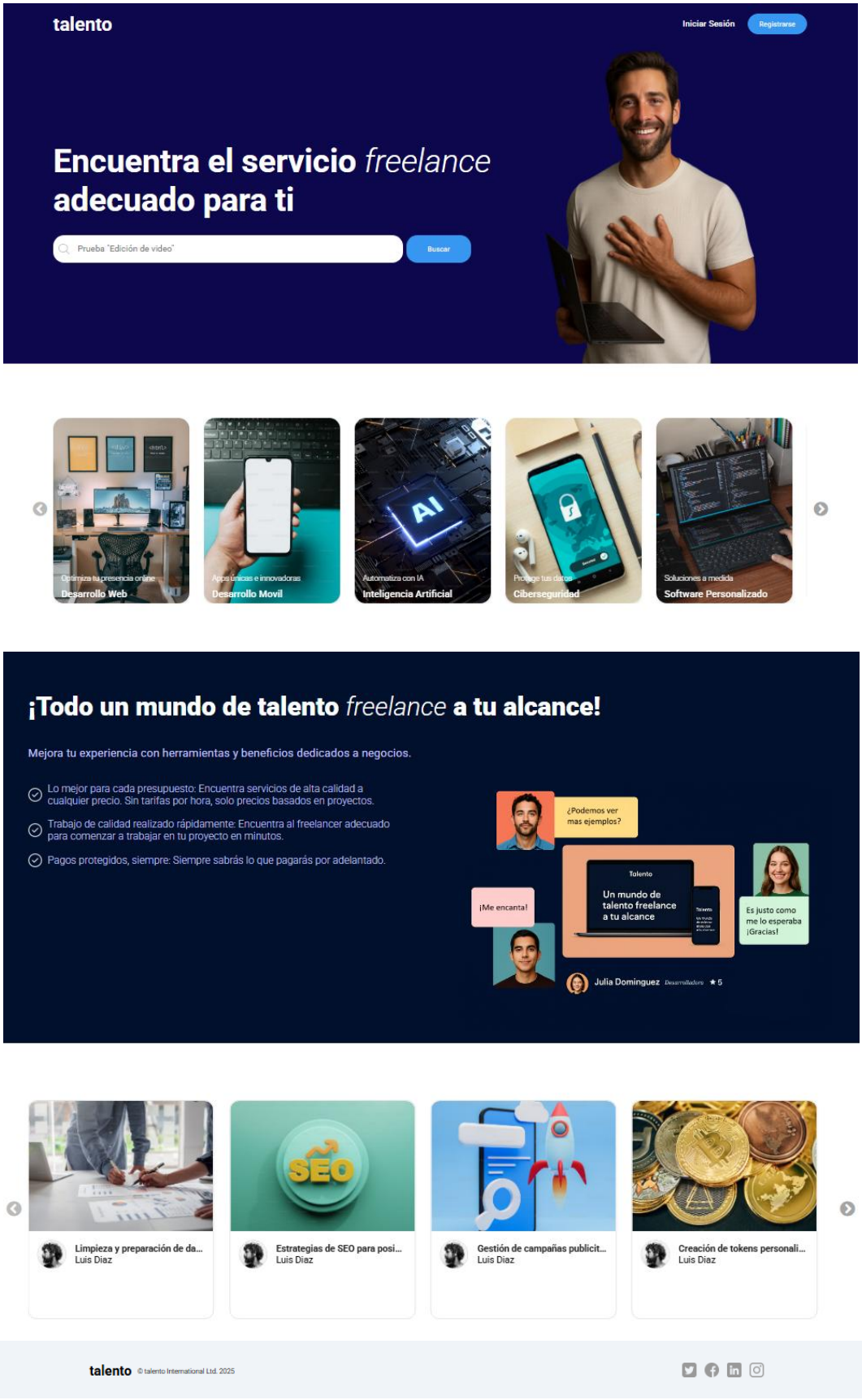


Figura 76. Manual de Usuario página Home

La primera vista al ingresar a la plataforma es la pantalla de inicio (Figura 77), común para cualquier visitante, ya sea que tenga o no cuenta en el sistema. Esta pantalla está diseñada para ofrecer una navegación clara y visual, facilitando el descubrimiento de servicios desde el primer momento.

En la parte superior se encuentran dos botones destacados: uno para iniciar sesión y otro para registrarse. Justo debajo, se despliega un menú de navegación fijo que acompañará al usuario a lo largo de toda la experiencia, permitiéndole acceder rápidamente a las diferentes categorías disponibles.

La pantalla incluye también un buscador donde se pueden introducir palabras clave para filtrar gigs relacionados. A continuación, se presenta un carrusel visual con las distintas categorías disponibles, una sección donde veremos información de la plataforma, y finalmente un segundo carrusel donde se destacan algunos gigs.

Navegación de un usuario no registrado:

Si el usuario accede a la plataforma sin haber iniciado sesión, podrá explorar libremente las categorías de gigs a través del menú principal, del carrusel de categorías o bien realizando una búsqueda específica mediante el buscador. Al hacer clic en cualquiera de estas opciones, se mostrará una vista con todos los gigs disponibles que coinciden con la categoría o los términos de búsqueda seleccionados (Figura 78).

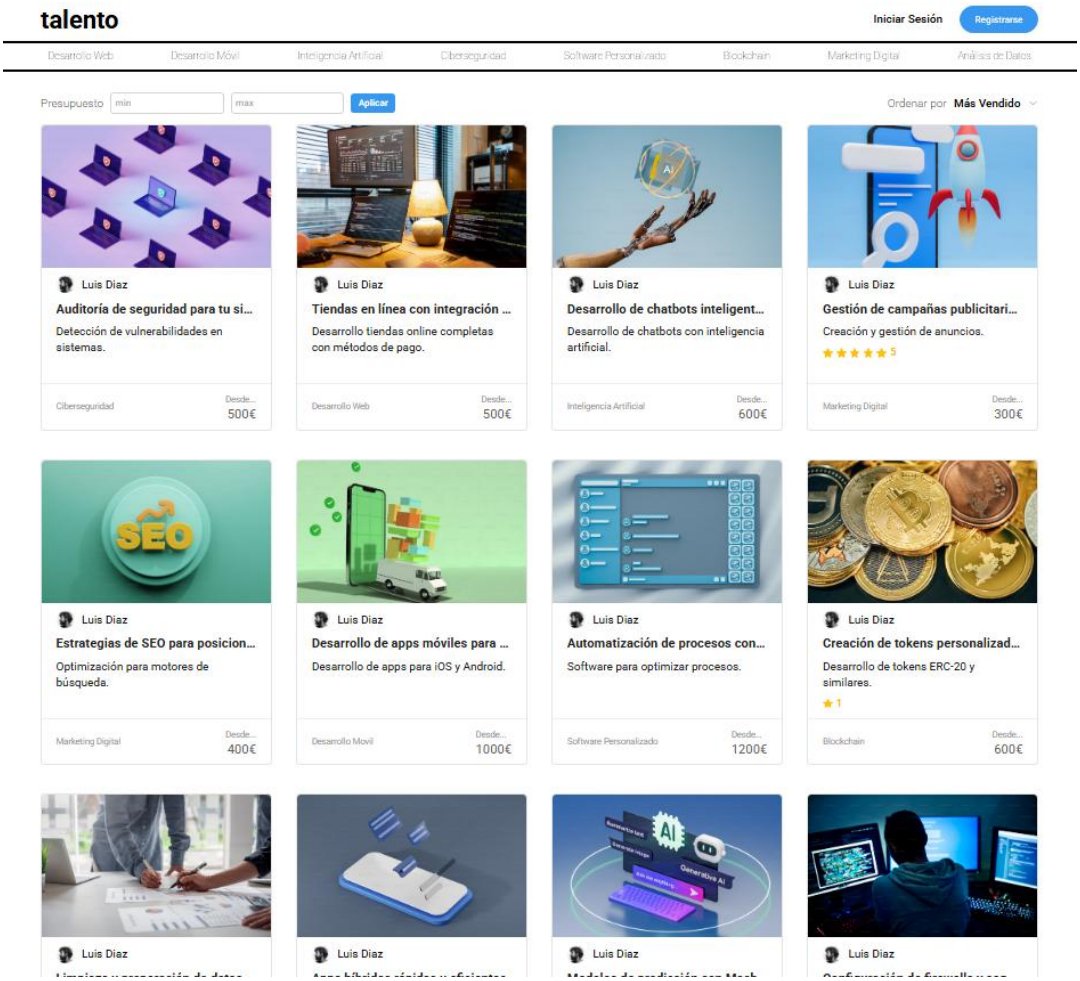


Figura 79. Manual de Usuario página Gigs

Desde esta vista, el usuario no registrado puede volver a filtrar los gigs por presupuesto introduciendo un presupuesto mínimo y uno máximo o también puede ordenar estos gigs por más vendido o reciente. Este usuario también puede pulsar sobre cualquier gig para ver más detalles. Esto lo llevará a una pantalla de detalle del gig, donde podrá visualizar toda la información relevante: título, descripción, características ofrecidas, nombre del vendedor, precio, valoraciones, entre otros datos importantes.

Sin embargo, al no tener una sesión iniciada, el usuario no podrá contratar el servicio ni interactuar con el vendedor. En lugar de ello, verá un botón con el texto "Inicia sesión para continuar" (Figura 80). Si intenta realizar alguna acción como opinar sobre el gig, también se le mostrará un mensaje informativo indicando que debe iniciar sesión para poder continuar.

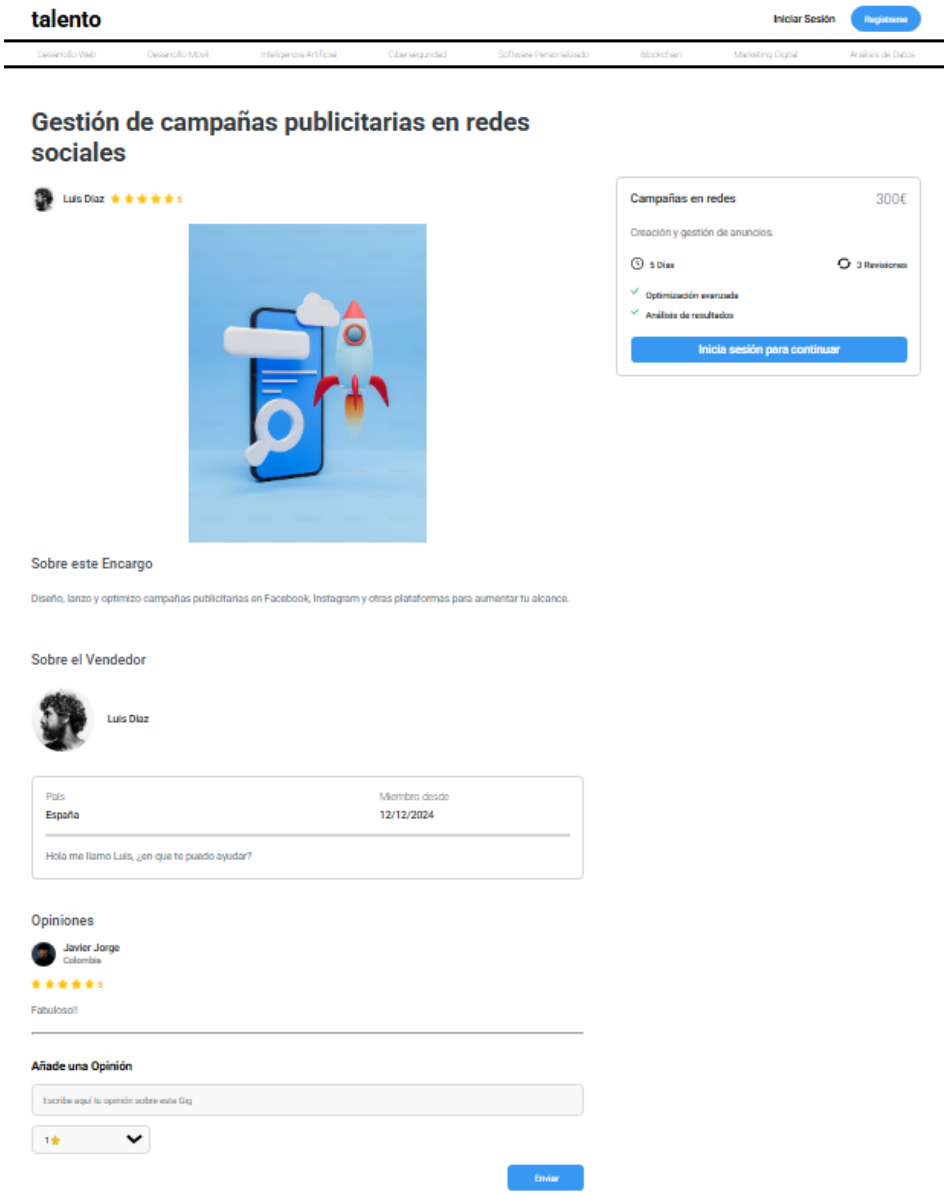


Figura 81. Manual de Usuario página Gig detalle – Usuario no registrado

De este modo, queda claro que un usuario no registrado puede visualizar el contenido de la plataforma, pero no interactuar con él. Para continuar, será necesario crear una cuenta o iniciar sesión si ya se dispone de una.

Navegación de un usuario registrado – Clientes y vendedores:

Una vez que el usuario decide formar parte activa de la plataforma, debe pasar por el proceso de registro. Este proceso está diseñado para ser intuitivo y seguro. El formulario de registro incluye campos básicos como nombre, correo electrónico y contraseña, cada uno con sus correspondientes validaciones para asegurar que la información introducida sea válida. Por ejemplo, se comprueba que el correo tenga un formato correcto, que la contraseña cumpla con los requisitos de seguridad y que todos los campos obligatorios estén cumplimentados.

Un detalle clave del formulario es la presencia de un botón tipo switch, que permite al usuario elegir si desea registrarse como cliente o como vendedor (Figura 82). Esta selección inicial determina los permisos y funcionalidades a las que tendrá acceso una vez dentro de la plataforma.

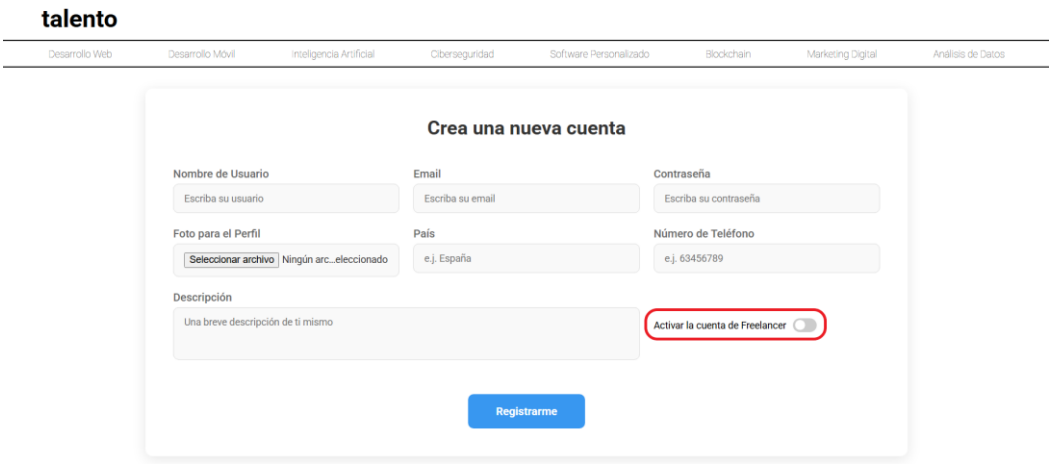


Figura 83. Manual de Usuario página Registro

Tras completar el registro, el sistema redirige automáticamente al usuario a la pantalla de inicio de sesión (Figura 84). Aquí deberá introducir el correo electrónico con el que se registró y la contraseña correspondiente. Al igual que en el registro, existen validaciones que garantizan el correcto uso de esta funcionalidad, como detectar si el correo no existe en la base de datos o si la contraseña introducida es incorrecta. Una vez superado este paso, el usuario accede nuevamente a la pantalla de inicio, pero ya con su sesión activa.

Figura 85. Manual de Usuario página Login

Interacción con gigs:

Con la sesión iniciada, el comportamiento de la plataforma cambia. Ahora, al navegar desde la pantalla de inicio hasta una categoría específica o tras realizar una búsqueda, el usuario puede ver los gigs disponibles y acceder a sus detalles tal y como lo haría un usuario no registrado. Sin embargo, al entrar al detalle de un gig, el botón que antes decía "Inicia sesión para continuar" ha sido sustituido por "Continuar" (Figura 86). Al hacer clic en este botón, el sistema redirige al usuario directamente al formulario de pago del que luego hablaremos, habilitando así la contratación del servicio.

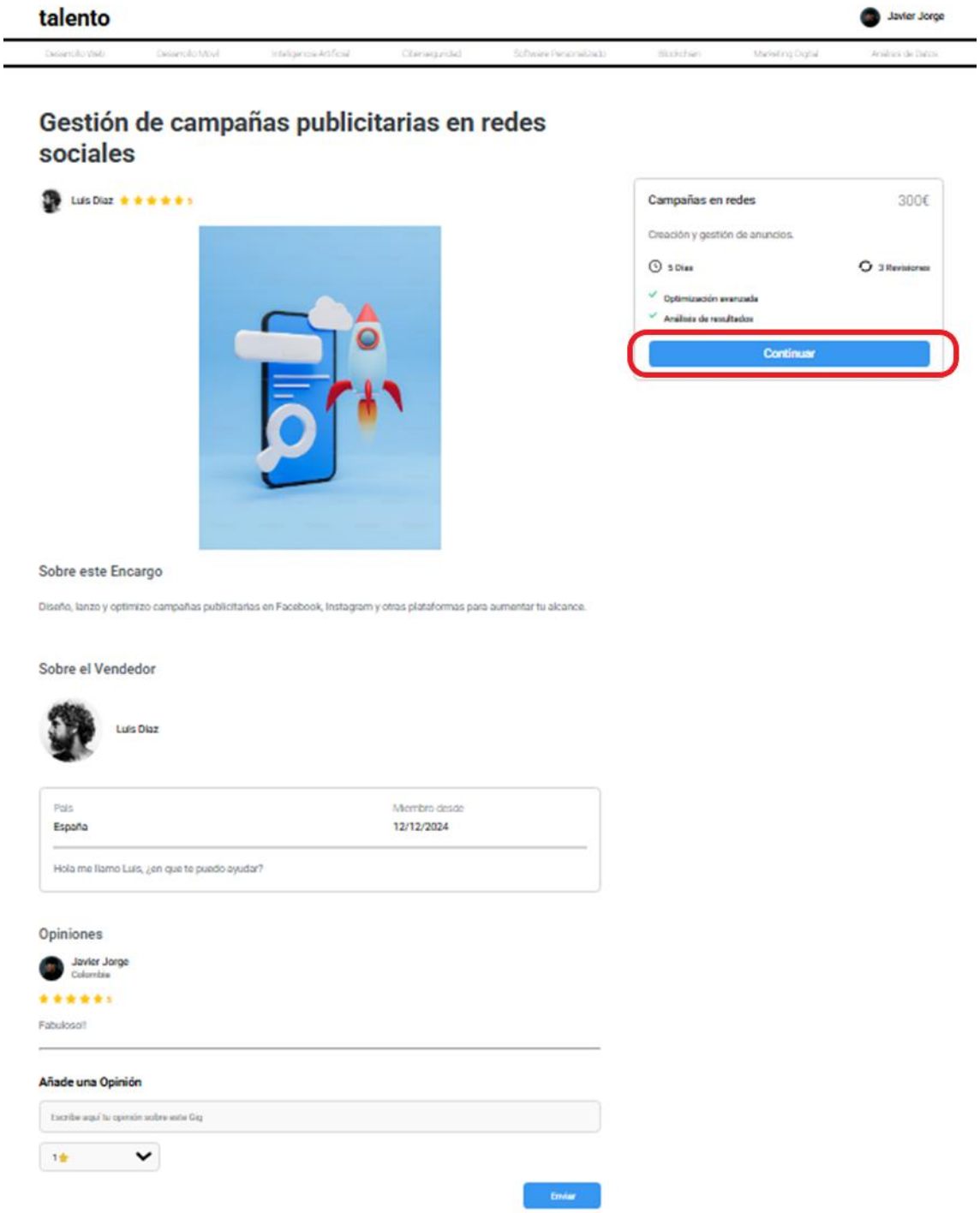


Figura 87. Manual de Usuario página Gig detalle – Usuario registrado

Un aspecto interesante de esta pantalla es que si el usuario que está visualizando el gig es también su propietario (es decir, si el vendedor intenta contratar su propio servicio), el botón cambiará automáticamente por un mensaje que indica: “No puedes contratar tu propio gig”, impidiendo así la operación (Figura 88).

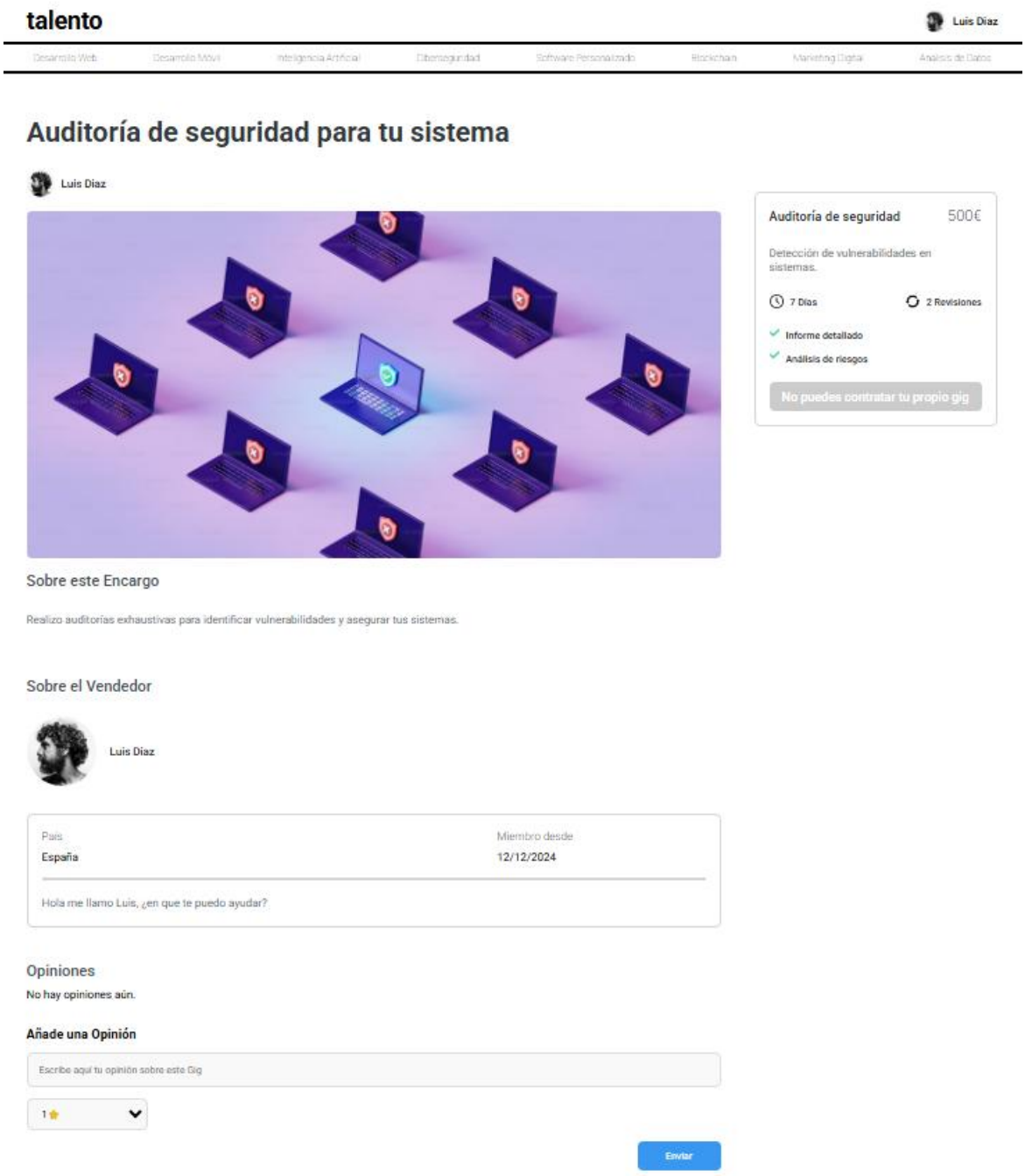


Figura 89. Manual de Usuario página Gig detalle – Usuario es también propietario del Gig

En cuanto a las valoraciones, se establece una condición clara, solo podrán opinar aquellos usuarios que hayan contratado ese gig. Además, cada usuario únicamente podrá dejar una opinión por gig. Si se intenta opinar sin haber contratado el servicio, aparecerá un mensaje informativo indicando que primero debe realizarse una contratación. Y si el usuario ya dejó su opinión anteriormente, se mostrará una notificación indicando que ya ha sido registrada una valoración.

Formulario de pago:

Al hacer clic en “Continuar” desde la pantalla de detalle de un gig, el usuario es redirigido a la pantalla de pago (Figura 90). Esta contiene un resumen conciso del gig que se está contratando y un formulario en el que se debe introducir la información de la tarjeta de crédito.

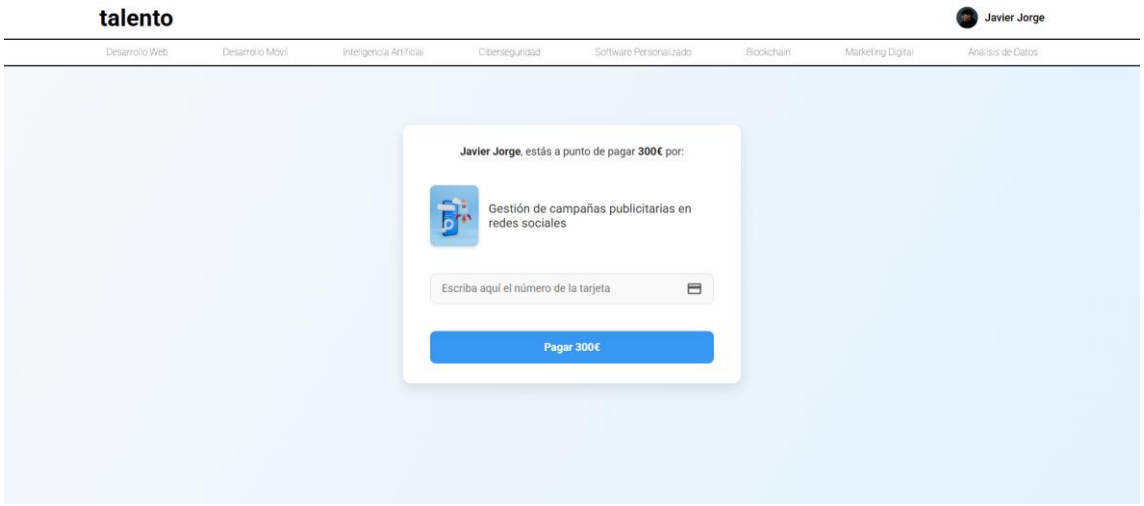


Figura 91. Manual de Usuario página Formulario de pago

Ya que se trata de un formulario de pago ficticio, solo se acepta una tarjeta ficticia válida con el número “1111 1111 1111 1111”. Si se introduce un número diferente, el sistema mostrará un mensaje de error y no permitirá completar el pedido.

En caso de que el pago sea exitoso, se genera un pedido y el usuario es redirigido automáticamente a la pantalla de Mis pedidos.

Funcionalidades comunes para clientes y vendedores:

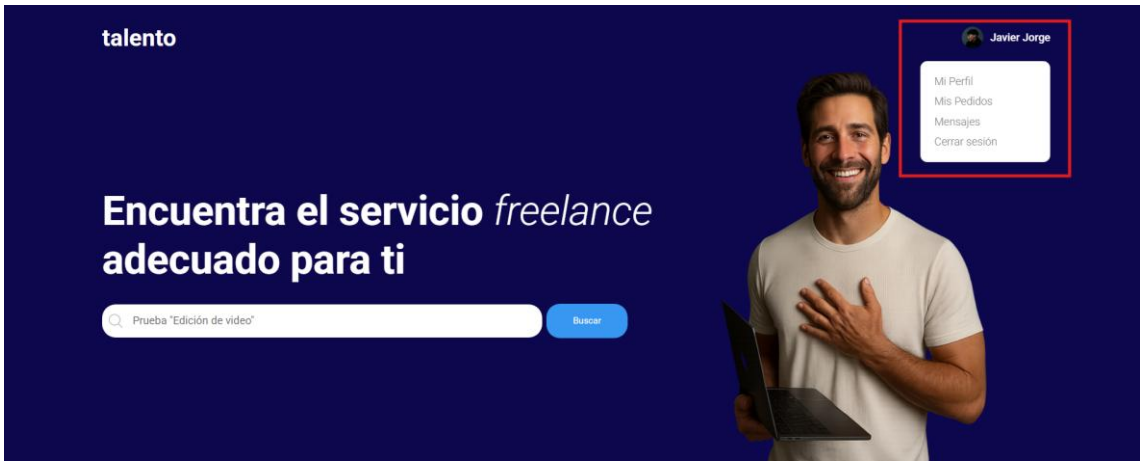


Figura 92. Manual de Usuario funcionalidades comunes para clientes y vendedores

Una vez autenticado, si el usuario pulsa en la parte superior en su nombre, dispone de un desplegable que le permite acceder a secciones adicionales de la plataforma (Figura 93), entre ellas: Mi perfil, Mis pedidos, Mensajes y Cerrar sesión.

Mi perfil:

En la sección Mi perfil (Figura 94), los usuarios pueden visualizar y modificar su información personal, como nombre, correo electrónico o contraseña.

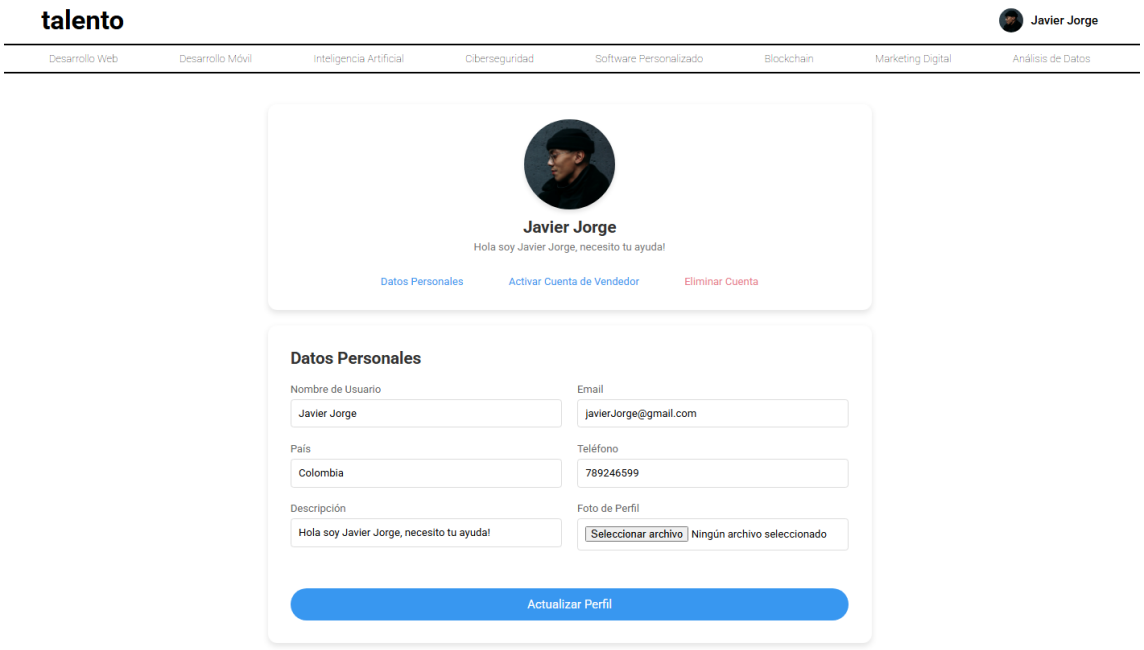


Figura 95. Manual de Usuario página Mi perfil

Desde esta misma pantalla también es posible activar la cuenta de vendedor, si inicialmente se registraron como clientes. Esta activación se realiza mediante un botón tipo switch (Figura 96), igual al del formulario de registro, lo que transforma la cuenta y habilita las funcionalidades exclusivas para los vendedores.

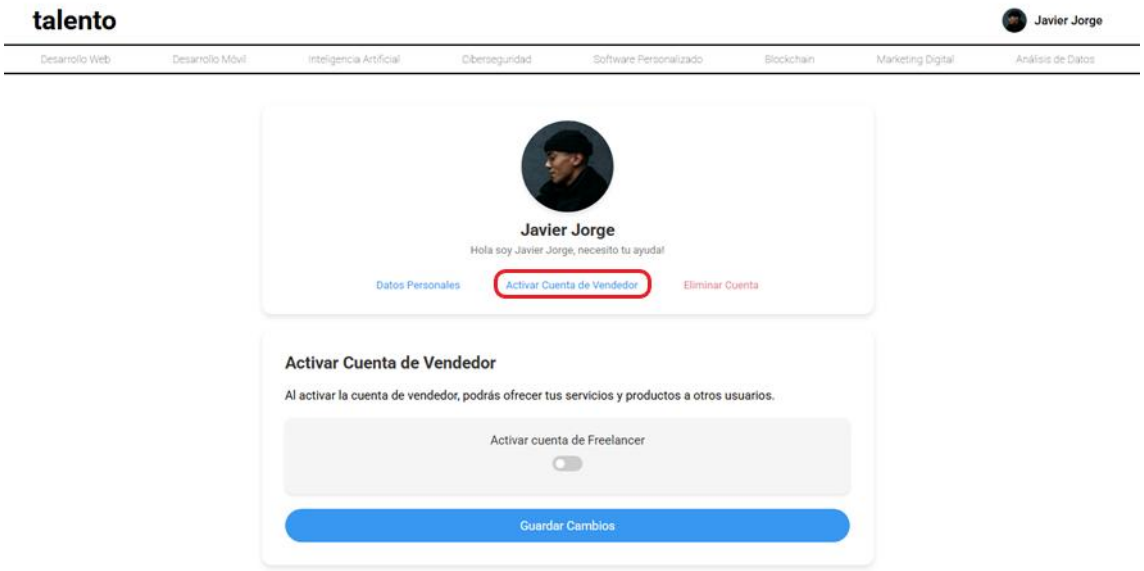


Figura 97. Manual de Usuario página Mi perfil - Activar cuenta de vendedor

Asimismo, desde esta sección se podrá eliminar la cuenta de usuario siempre que sea posible. Si el usuario es vendedor y tiene pedidos pendientes de entregar no se le permitirá eliminar su cuenta, si todo esta entregado se eliminara su cuenta y toda la información asociada a ese vendedor. En el caso de los usuarios clientes si tienen algún pedido pendiente de entrega se les advertirá que perderán su dinero invertido en el pedido si borran su cuenta, si aceptan esa advertencia su cuente será eliminada.

Mis pedidos:

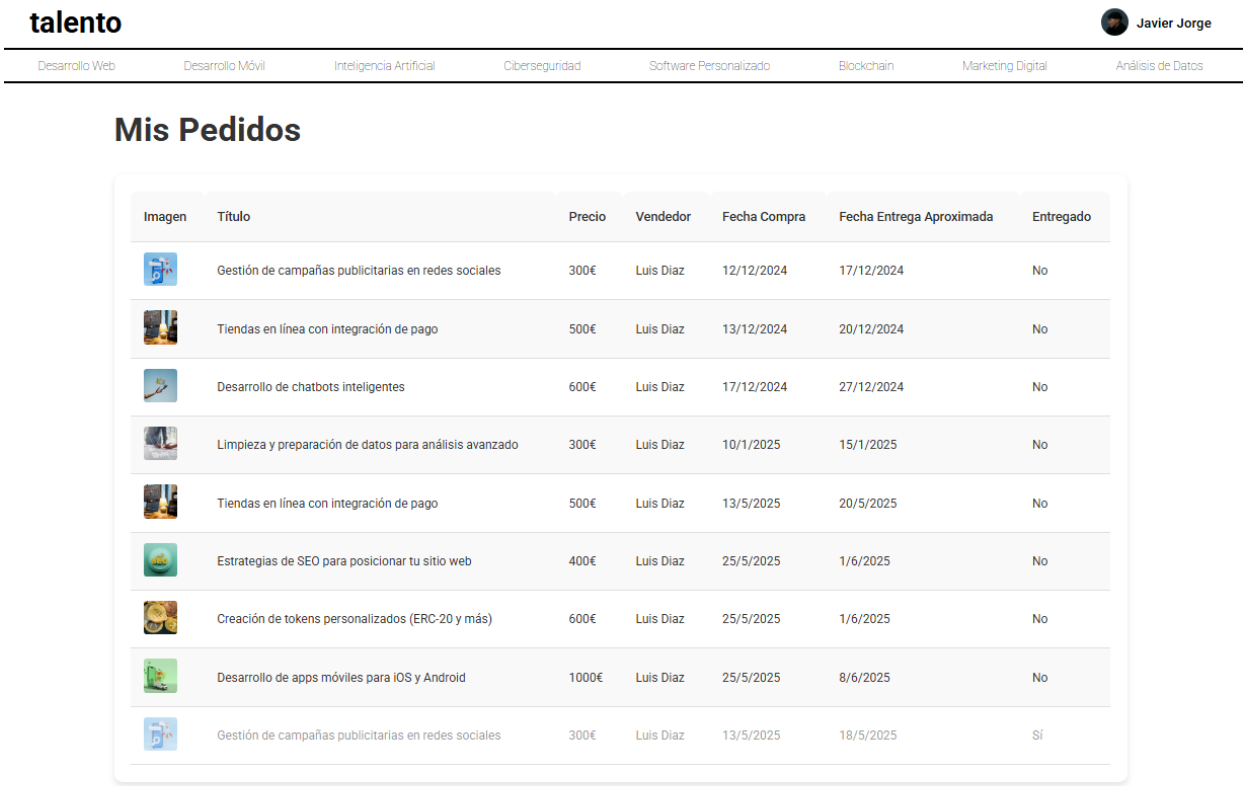


Figura 98. Manual de Usuario página Mis pedidos

Esta pantalla muestra el historial de contrataciones realizadas por el usuario (Figura 99). Cada pedido incluye información relevante como el título del gig, el vendedor, la fecha de contratación, el estado del pedido y el botón para acceder al detalle.

Los pedidos que ya han sido entregados se colocan al final de la lista con un aspecto visual atenuado, indicando que su ciclo ha finalizado. Como se mencionó antes, tras completar una contratación, el sistema redirige directamente a esta pantalla para facilitar el seguimiento.

Mensajes:

Los mensajes permiten la comunicación entre usuarios y vendedores. Es importante destacar que solo los vendedores pueden iniciar una conversación, aunque una vez creada, ambas partes pueden participar libremente.

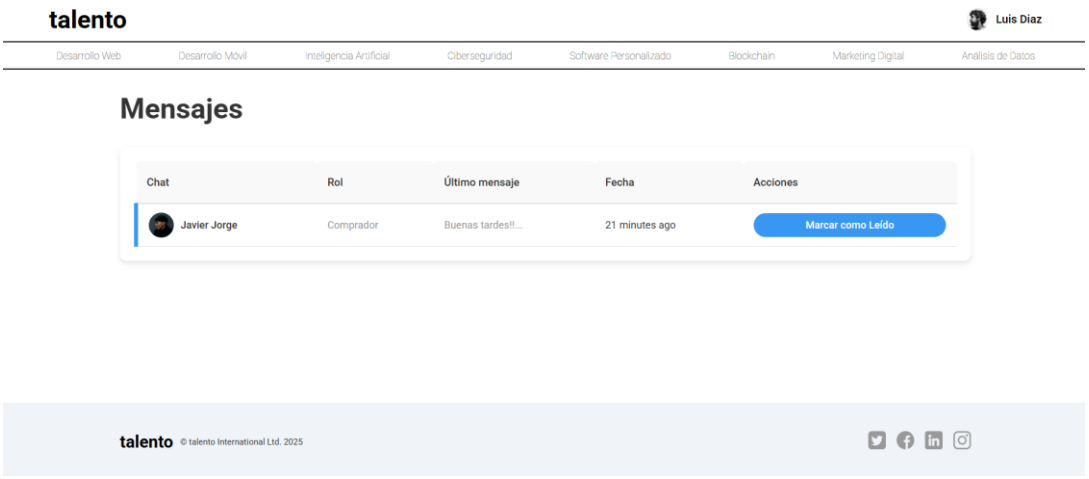


Figura 100. Manual de Usuario página Conversaciones

Desde la pantalla de conversaciones se puede marcar un mensaje como leído (Figura 101) y acceder al hilo completo haciendo clic en el último mensaje recibido (Figura 102).

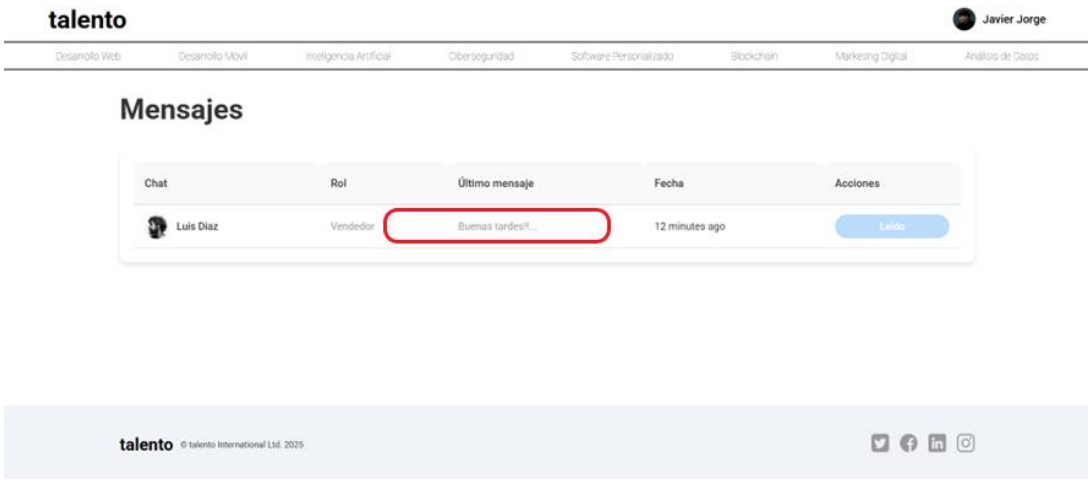


Figura 103. Manual de Usuario página Conversaciones - Ultimo mensaje

Una vez dentro de una conversación, los mensajes se presentan ordenados cronológicamente, diferenciando claramente al emisor y al receptor. Hay un campo para escribir nuevos mensajes y un botón para enviarlos (Figura 104).

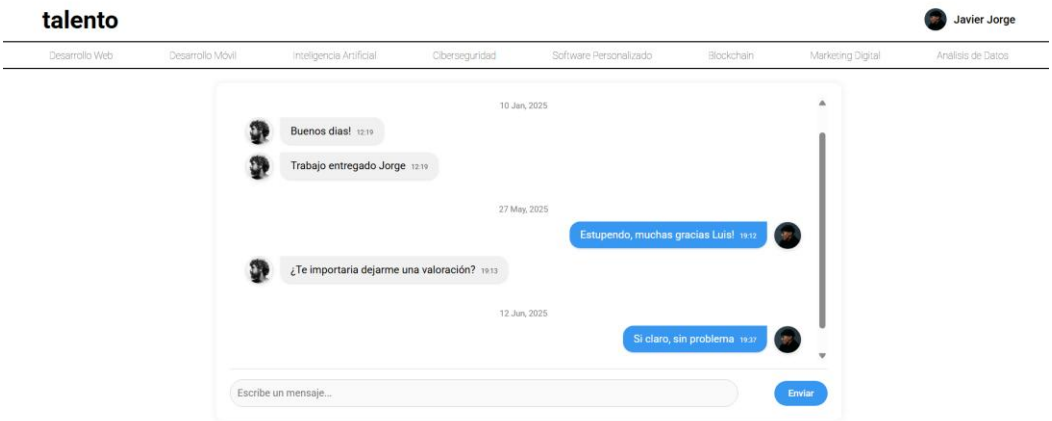


Figura 105. Manual de Usuario página Chat concreto

Funcionalidades exclusivas para vendedores:

Además de las opciones mencionadas anteriormente, los vendedores tienen acceso a secciones específicas para gestionar sus servicios y pedidos recibidos (Figura 106).

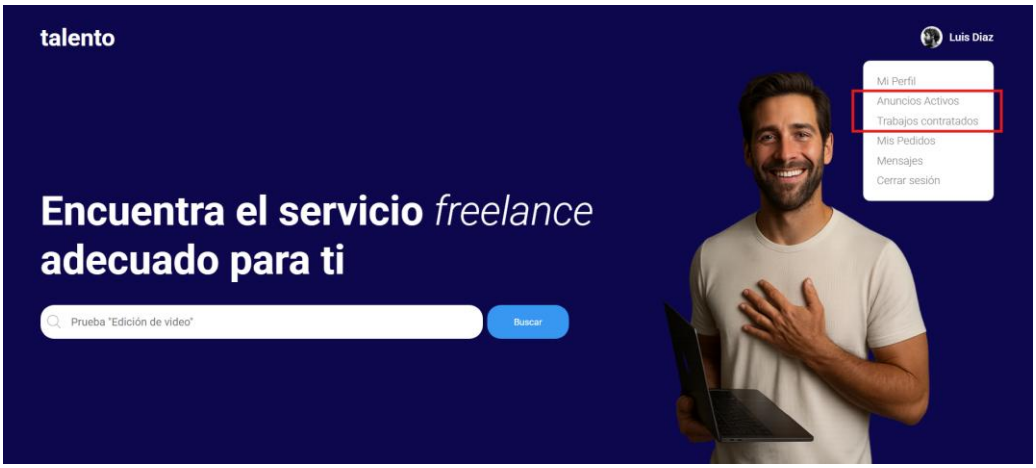


Figura 107. Manual de Usuario funcionalidades exclusivas para vendedores

Anuncios activos:

En la pantalla Mis anuncios activos (Figura 108), el vendedor puede ver todos los gigs que tiene publicados. Cada uno muestra detalles como el título, el precio, número de ventas, y estado. Desde aquí también se puede eliminar un gig, siempre y cuando no tenga pedidos pendientes de entrega. Si se intenta eliminar un gig con pedidos activos, el sistema mostrará un mensaje advirtiendo que esta acción no está permitida.

En la parte superior de esta pantalla se encuentra un botón para Añadir un nuevo encargo.

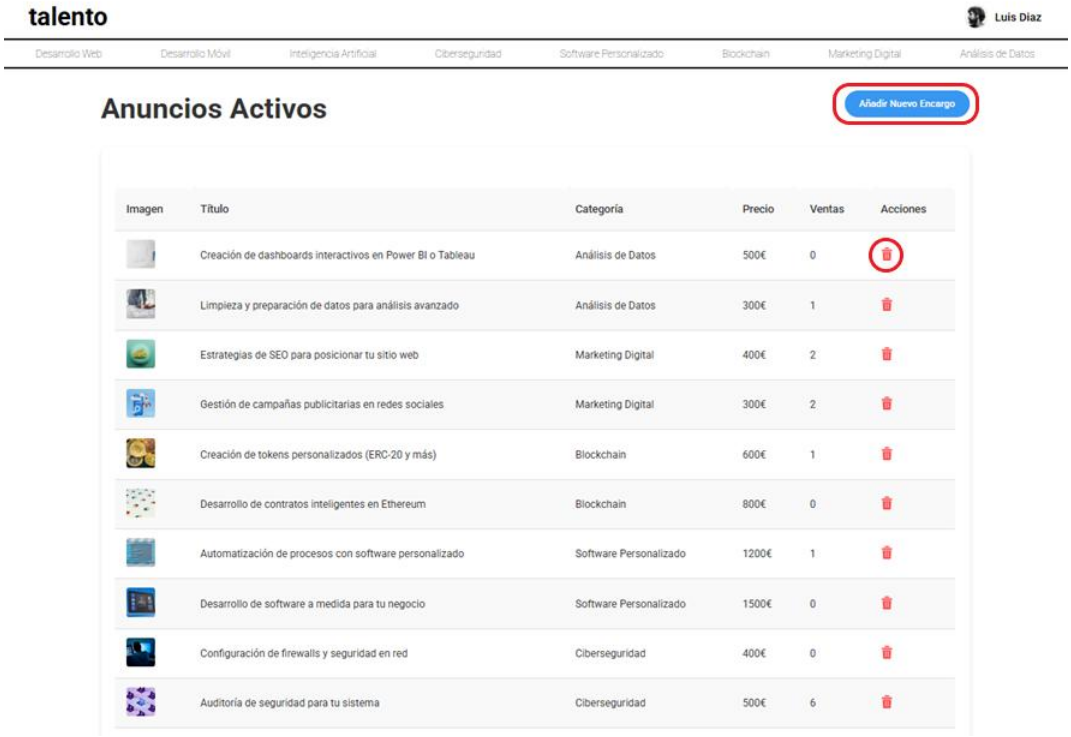


Figura 109. Manual de Usuario página Anuncios activos

Al hacer clic en este botón, se accede a un formulario en el que se puede registrar un nuevo gig (Figura 110), introduciendo título, descripción, precio y características. Existe también un botón para "Añadir característica", que permite agregar tantos campos como se deseen, de forma dinámica.

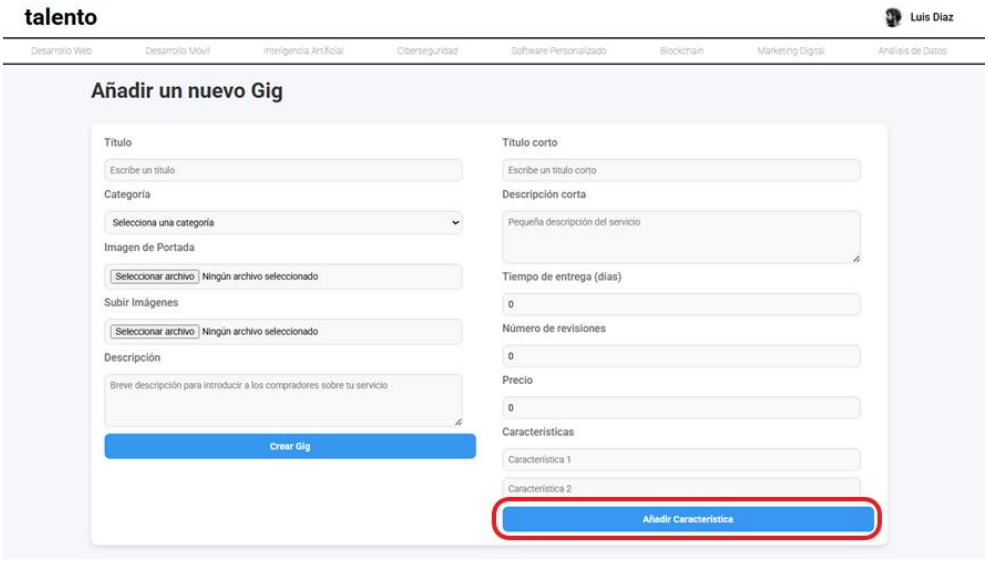


Figura 111. Manual de Usuario página Añadir gig

Trabajos contratados:

Esta sección muestra un historial de los pedidos que los usuarios han realizado a un vendedor (Figura 112). Cada entrada incluye información como el nombre del cliente, la fecha de contratación y el estado del pedido.

Figura 113. Manual de Usuario página Trabajos contratados

Desde aquí, el vendedor puede realizar tres acciones:

- Contactar con el cliente mediante un icono que abre el chat correspondiente.
- Marcar el pedido como entregado a través de un botón tipo switch. Una vez activado, esta acción es irreversible y el pedido aparecerá con una apariencia atenuada al final de la lista.
- Consultar información del comprador mediante un icono de información, que despliega una ficha con los datos básicos del usuario que contrató el servicio (Figura 114).

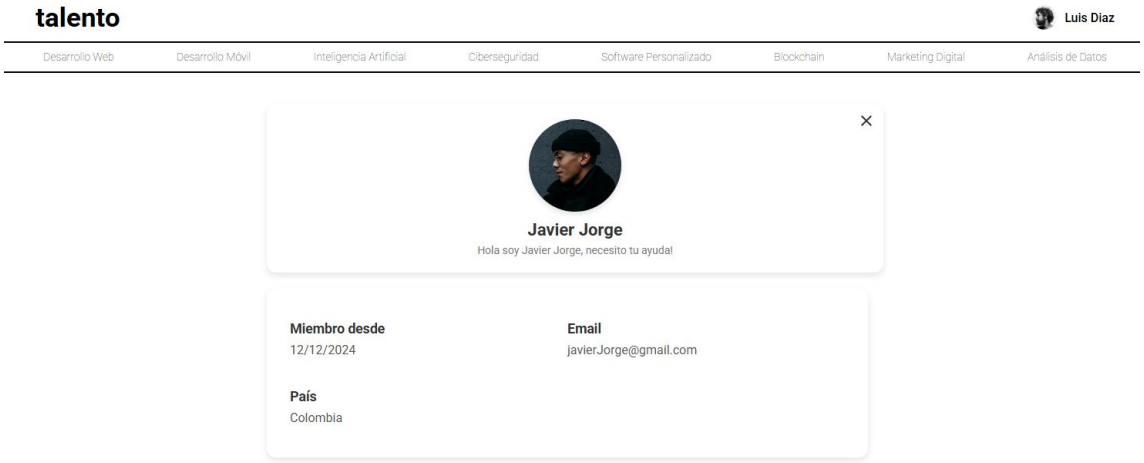


Figura 115. Manual de Usuario página Perfil del cliente que nos contrató

Capítulo 8
Conclusiones y trabajo futuro

Este capítulo ofrece una reflexión final sobre el trabajo realizado, destacando las principales conclusiones obtenidas a lo largo del desarrollo del proyecto. También, se presentan diversas líneas de mejora y posibles ampliaciones que podrían integrarse en la plataforma en futuras iteraciones.

8.1. Conclusiones

Conclusiones del proyecto

El Trabajo de Fin de Grado ha cumplido con los objetivos técnicos y funcionales definidos en su fase de planificación. Se ha diseñado y desarrollado una plataforma web funcional de tipo marketplace, que permite la contratación de servicios digitales entre usuarios a través de una arquitectura basada en tecnologías modernas.

El sistema incluye funcionalidades clave como el registro y autenticación de usuarios, la creación y gestión de gigs, la mensajería interna, y una estructura de datos coherente y escalable sustentada en MongoDB. Además, se ha trabajado con principios de diseño centrado en el usuario, ofreciendo una interfaz clara y moderna.

A lo largo del desarrollo, se ha aplicado una metodología de trabajo en espiral, combinando fases de diseño, análisis, implementación y pruebas. Los diagramas de casos de uso, secuencia y entidad-relación, así como la planificación temporal, han servido como guías estructurales del proceso.

El backend desarrollado con Node.js y Express, junto con un frontend en React, ha demostrado ser una arquitectura eficiente y extensible, adecuada para futuros crecimientos. Las pruebas funcionales han verificado que el sistema cumple los requisitos definidos, tanto funcionales como no funcionales (usabilidad, seguridad, portabilidad...).

En resumen, el TFG ha dado lugar a un producto robusto, bien documentado y con una base tecnológica sólida, que puede ser mantenido y ampliado fácilmente.

Conclusiones personales

Este proyecto ha sido una experiencia muy enriquecedora y emocionante desde el primer momento. Más allá del desarrollo técnico, ha supuesto también un reto personal en cuanto a organización, planificación y superación de obstáculos. Ver cómo una idea inicial se ha transformado en una plataforma funcional es, sin duda, profundamente gratificante.

También ha sido una oportunidad para aplicar y consolidar conocimientos en desarrollo web full stack, concretamente con el stack MERN (MongoDB, Express, React y Node.js). Afrontar su uso sin apenas experiencia previa ha supuesto un auténtico desafío, pero también una valiosa fuente de aprendizaje que me ha permitido profundizar en conceptos

clave como la creación de APIs, autenticación segura o la gestión de datos en bases de datos NoSQL.

Es cierto que el tiempo de entrega ha sido algo mayor del que me hubiera gustado. Compatibilizar el desarrollo con responsabilidades laborales ha sido complicado por momentos, y ha requerido una buena dosis de organización, constancia y compromiso. Aun así, estoy satisfecho con el resultado final, ya que el sistema cumple con los objetivos propuestos, funciona de forma estable y ofrece una base sólida sobre la que seguir mejorando.

Este proyecto me ha servido también para reafirmar el valor del enfoque práctico y realista en la resolución de problemas, así como la importancia de diseñar pensando en la experiencia del usuario y en aspectos fundamentales como la seguridad, la escalabilidad o la accesibilidad.

En definitiva, el proyecto ha sido una excelente oportunidad para consolidar habilidades técnicas, desarrollar autonomía en la toma de decisiones y profundizar en el desarrollo de aplicaciones web completas. Me voy con una sensación muy positiva y con una base sólida para seguir creciendo profesional y personalmente.

8.2. Líneas de trabajo futuras

El sistema desarrollado cumple con los objetivos propuestos y constituye una plataforma funcional, sólida y escalable. No obstante, en un entorno tecnológico en constante evolución, existen oportunidades interesantes para enriquecer y perfeccionar la solución actual.

A continuación, se recogen posibles líneas de trabajo que permitirían evolucionar el proyecto hacia un producto aún más completo, competitivo y preparado para el mercado real:

Incorporación progresiva de inteligencia artificial:

La integración de tecnologías de IA abriría la puerta a funcionalidades más avanzadas, tales como:

- Recomendaciones personalizadas de gigs, basadas en los hábitos de navegación de los usuarios.
- Chatbots de asistencia inteligente para atención 24 horas.
- Análisis automatizado de valoraciones mediante técnicas de procesamiento de lenguaje natural (NLP).
- Moderación automática de contenido inapropiado, garantizando la seguridad y calidad de la plataforma.

Desarrollo de un panel de administración:

Aunque la plataforma es autogestionable por usuarios, una herramienta de gestión centralizada permitiría mayor control del sistema:

- Perfil de administrador con acceso a gestión de usuarios, gigs, opiniones y pagos.

- Dashboard con estadísticas en tiempo real sobre actividad y rendimiento.
- Sistema de moderación y reportes con intervención manual o automática.

Ampliación de funcionalidades para vendedores y compradores:

Para potenciar el valor de la plataforma para ambos perfiles:

- Sistema de niveles para vendedores, con rangos o insignias según desempeño.
- Ofertas de paquetes de gigs (básico, estándar, premium).
- Calendario de disponibilidad y entregas para una mejor gestión del tiempo.

Evolución del sistema de mensajería y notificaciones:

Con el fin de optimizar la comunicación dentro de la plataforma:

- Mensajería en tiempo real, mediante WebSockets o tecnologías similares.
- Notificaciones push y por correo electrónico.

Integración de pasarelas de pago reales y facturación:

De cara a un entorno productivo:

- Conexión con plataformas como Stripe o PayPal.
- Facturación automática y soporte para múltiples monedas.

Exploración de modelos de monetización sostenibles:

En fases futuras, podrían estudiarse opciones como:

- Comisiones por cada gig contratado.
- Membresías premium para vendedores.
- Publicidad destacada respetando la experiencia de usuario.
- Servicios de valor añadido, como análisis o asesoramiento personalizado.

Webgrafía

[1] bcryptjs. (s.f.). bcryptjs. npm. https://www.npmjs.com/package/bcryptjs

[2] Boehm, B. W. (1988). A spiral model of software development and enhancement. ACM SIGSOFT. https://doi.org/10.1145/12944.12948

[3] Dotenv. (s.f.). dotenv – Loads environment variables from .env files. npm. https://www.npmjs.com/package/dotenv

[4] Express.js. (s.f.). Express – Fast, unopinionated, minimalist web framework for Node.js. Express. https://expressjs.com/

[5] Git. (s.f.). Git – Distributed version control system. https://git-scm.com/

[6] GitHub Docs. (s.f.). Documentación oficial de GitHub. https://docs.github.com/

[7] Glassdoor. (s.f.). Sueldo de desarrollador de software en España. https://www.glassdoor.es/Sueldos/desarrollador-de-software-sueldo-SRCH_KO0,25.htm

[8] JSON Web Token (JWT). (s.f.). Introduction to JSON Web Tokens. JWT.io. https://jwt.io/introduction

[9] Material-UI. (s.f.). Material UI – React components for faster web development. https://mui.com/

[10] MongoDB. (s.f.). MongoDB Developer Documentation. https://www.mongodb.com/docs/

[11] Mongoose. (s.f.). Mongoose – Elegant MongoDB object modeling for Node.js. https://mongoosejs.com/

[12] Node.js. (s.f.). Node.js v20.x Documentation (API). https://nodejs.org/docs/latest-v20.x/api/index.html

[13] React. (s.f.). React – A JavaScript library for building user interfaces. https://react.dev/

[14] React Router. (s.f.). React Router – Declarative routing for React apps. https://reactrouter.com/

[15] UML Diagrams.org. (s.f.). How to draw a feature tree. https://www.umldiagrams.org/feature-tree.html

[16] Visual Paradigm. (s.f.). Use Case Diagram tutorial. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

[17] Visual Paradigm. (s.f.). Sequence Diagram vs. Activity Diagram. https://www.visual-paradigm.com/guide/uml-unified-modeling-language/sequence-diagram-vs-activity-diagram/