UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

APLICACIÓN WEB: PORTAL DE CLIENTES SINCRONIZADO CON ERP I CRM

ALEX PERTUSA CUADRADO

Director/a
NIL SAMPER MAS (NBGROUP SOLUCIONES DE NEGOCIO SL)

Ponente: MARÍA JOSÉ CASAÑ GUERRERO (Departamento de Ingenieria de Servicios y Sistemas de Información)

Titulación
Grado en Ingeniería Informática (Ingeniería del Software)

Memoria del trabajo de fin de grado

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

20/01/2025

AGRADECIMIENTOS

Quiero agradecer a

- Mi tutor de empresa, Nil Samper Mas, por haberme ayudado, asesorado y guiado en todas las fases del proyecto.

- Mi ponente, Mª Jose Casany, por aceptar la propuesta de ser mi poniente en este proyecto y por aconsejar durante todo el transcurso del proyecto.

- Los dueños de *nbGroup Soluciones de Negocio S.L.*, Sergi Zaragoza Manrubia y Daniel Sanchez García, por la confianza depositada en mí y por permitirme desarrollar el proyecto a mi manera con total libertad desde el primer día.

- Compañeros de nbGroup, por haber participado en el apartado de testing y dar su opinión para mejorar ciertos aspectos de la aplicación.

- Mis padres, por ser un apoyo fundamental y haber estado a mi lado a lo largo del proyecto y de toda la carrera.

RESUMEN

Este proyecto nace tras completar las prácticas en la empresa, con la idea de optimizar los procesos internos para mejorar la eficiencia operativa y fortalecer la relación con el cliente. El objetivo es centralizar toda la información relevante al cliente en una única plataforma intuitiva y accesible, dando la capacidad de que el cliente se autogestione y no tenga la dependencia de interactuar directamente con el personal de la empresa.

Para lograr esto, se ha desarrollado una aplicación web que integra varios servicios de Microsoft, incluyendo Business Central y Customer Service, donde se almacena toda la información del cliente. Esta aplicación abarca la gestión administrativa, permitiendo que los clientes accedan a sus facturas, abonos, servicios, albaranes y contratos. Además, incluye una sección de gestión de soporte, donde pueden consultar el calendario de intervenciones de la empresa y el estado actual de sus casos.

RESUM

Aquest projecte neix després de completar les pràctiques a l'empresa, amb la idea d'optimitzar els processos interns per millorar l'eficiència operativa i enfortir la relació amb el client. L'objectiu és centralitzar tota la informació rellevant del client en una única plataforma intuïtiva i accessible, donant la capacitat per a que el client s'autogestioni i no tingui la dependència d'interactuar directament amb el personal de l'empresa.

Per aconseguir això, s'ha desenvolupat una aplicació web que integra diversos serveis de Microsoft, incloent Business Central i Customer Service, on es guarda tota la informació del client. Aquesta aplicació abasta la gestió administrativa, permetent als clients accedir a les seves factures, abonaments, serveis, albarans i contractes. A més, inclou una secció de gestió de suport, on poden consultar el calendari d'intervencions de l'empresa i l'estat actual dels seus casos.

ABSTRACT

This project born after completes the practices at the company, with the idea of optimize the internal processes to enhance operational efficiency and strengthen customer relationships. The purpose is to centralize all relevant customer information on a single intuitive and accessible platform, giving the client the ability to self-manage and not have the dependency of interacting directly with the company's staff.

To achieve this, a web application has been developed that integrates various Microsoft services, including Business Central and Customer Service, where all customer information is stored. This application covers administrative management, allowing customers to access their invoices, credits, services, delivery notes, and contracts. It also includes a support management section, where they can view the company's intervention calendar and the current status of their cases.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción y contextualización

El proyecto “Portal de clientes sincronizado con Dynamics 365 Customer Service y Business Central” se trata de un trabajo de final de Grado de Ingeniería Informática de la Facultad de Informática de Barcelona, Universidad Politécnica de Cataluña.

Este trabajo de final de grado se realiza en la modalidad B (proyecto realizado en una empresa) en la empresa *nbGroup Soluciones de Negocio S.L.* y pertenece a la especialidad de Ingeniera de Software.

1.1 Contexto

La empresa *nbGroup Soluciones de Negocio S.L.* se dedica al mundo de la informática, más en concreto, a la consultoría informática del ERP Business Central. No obstante, no es su único mercado actual, ellos disponen de tres principales ramas de negocio **[1]**.

La primera rama es la parte de *nbWeb*, aquí encontramos soluciones web para publicar blogs o páginas estáticas informativas a través de WordPress. También encontramos soluciones para Prestashop y WooCommerce.

La segunda línea de negocio es la de *nbSystems*, en esta encontramos todo tipo de soluciones para la parte hardware y sistemas en la nube.

La tercera rama es la de *nbDynamics*. Aquí se centra la mayor parte de personal y de recursos de la empresa. Dentro de esta encontramos soluciones de consultoría personalizada a cliente para Business Central, desarrollos dentro del ERP y alguna pequeña aplicación que conecta el ERP con tecnologías muy antiguas.

1.2 Definición del problema

Los principales desafíos que enfrentan los clientes de la empresa están profundamente arraigados en la dispersión y desorganización de la información entre distintas plataformas, como el ERP y el CRM. Esta fragmentación de los datos crea una serie de complicaciones para los empleados de las áreas de administración y comercial, que se ven obligados a navegar entre varios sistemas para obtener una visión clara y precisa de los productos y servicios que los clientes han contratado. Esta falta de integración no solo retrasa los tiempos de respuesta, sino que también incrementa la probabilidad de errores, dado que la información no siempre está actualizada o disponible en un solo lugar.

Uno de los problemas más evidentes es la imposibilidad de tener una visión unificada de cada cliente. Los empleados tienen que realizar consultas en varias plataformas para acceder a los datos necesarios, lo que no solo consume tiempo, sino que también crea una experiencia frustrante para el cliente, que espera respuestas rápidas y precisas. Por ejemplo, cuando un cliente llama para hacer una consulta sobre un producto o servicio, el equipo de soporte a menudo no puede acceder de inmediato a toda la información relevante, lo que provoca demoras en la resolución de dudas y genera una percepción negativa del servicio.

Otro desafío crítico es la desconexión entre las facturas de venta mensuales y los registros de servicios previamente proporcionados. En muchos casos, los registros de los servicios prestados se envían por correo electrónico. Esto significa que los clientes tienen dificultades para correlacionar los cargos que aparecen en sus facturas con los servicios específicos que recibieron. La falta de transparencia en esta área no solo aumenta el riesgo de disputas sobre facturación, sino que también alimenta la frustración del cliente, que espera una mayor claridad y facilidad para gestionar sus pagos.

Además, el enfoque en métodos de comunicación tradicionales, como llamadas telefónicas y correos electrónicos, ha quedado obsoleto para una empresa que se esfuerza por la innovación tecnológica. Estos métodos no solo son ineficientes, sino que no se alinean con las expectativas actuales de los clientes, quienes prefieren soluciones más ágiles y accesibles en tiempo real. En un entorno empresarial moderno, donde las interacciones con los clientes deben ser fluidas y rápidas, estos métodos tradicionales se perciben como lentos e ineficaces. Los clientes esperan poder gestionar sus consultas y solicitudes mediante plataformas en línea que les ofrezcan acceso inmediato a la información, sin necesidad de recurrir a múltiples canales de comunicación.

Como resultado, todos estos factores contribuyen a un aumento significativo en el tiempo de espera y respuesta a las solicitudes de los clientes. Esta combinación de problemas genera una experiencia de usuario negativa, en la que los clientes se sienten frustrados por la falta de eficiencia y transparencia en la comunicación. Si no se abordan adecuadamente, estos desafíos pueden tener un impacto directo en la retención de clientes y en la reputación de la empresa, ya que los usuarios actuales demandan un servicio rápido, integrado y proactivo. En un mundo donde la experiencia del cliente es crucial para el éxito, no resolver estos problemas supone un riesgo significativo para la competitividad de la empresa.

1.3 Conceptos previos

ERP: *Enterprise Resource Planning*, es una plataforma software que integra todos los procesos que se pueden encontrar en una empresa. Dentro de una sola aplicación se puede encontrar la gestión de inventario, compras, ventas, finanzas y recursos humanos que facilitan a tomar decisiones estratégicas y operativas de manera rápida y precisa. **[2]**

CRM: *Customer Relationship Management*, es una plataforma software que se encarga de gestionar la relación y comunicación con los clientes. Se utiliza para recopilar detalles sobre los clientes con el objetivo de optimizar las estrategias comerciales y mejorar la interacción con ellos. **[3]**

Dynamics 365 Business Central: Es el ERP creado por *Microsoft* para pequeñas y medianas empresas. **[4]**

Dynamics 365 Customer Service: Es el CRM creado por *Microsoft.* **[5]**

Modelo 347: Es una declaración informativa obligatoria que se presenta ante la Agencia Tributaria en España. Sirve para declarar las operaciones realizadas con terceros (clientes o proveedores) que superen los 3.005,06 euros en un año. **[6]**

1.4 Actores implicados

Los actores implicados o *stakeholders* son todas las personas o entidades que están interesadas, beneficiadas, implicadas o afectadas por el proyecto. Su participación en el proyecto es necesaria, ya que a partir de ellas podemos definir los objetivos y requisitos.

A continuación, definimos los actores implicados dentro del proyecto:

- **Director del Trabajo de Fin de Grado:** Nil Samper Más, director de operaciones de la empresa *nbGroup Soluciones de Negocio S.L.* y persona encargada en asesorar al estudiante, supervisar el trabajo realizado y brindar orientación para que el proyecto avance de manera correcta y se complete con éxito.

- **Ponente:** María José Casany Guerrero, persona encargada en hacer de puente entre la empresa y la Facultad de Informática de Barcelona. También, en menor medida, se encarga de supervisar y guiar al alumno para realizar el proyecto de manera correcta.

- **Comité de dirección:** Grupo conformado por todos los directores de las distintas áreas de la empresa y dueños de la empresa. Es la parte interesada que define los requisitos funcionales de la aplicación.

- **Clientes:** Son la principal parte interesada e incluyen tanto a personas como a empresas. Son los principales usuarios que utilizarán la aplicación.

- **Desarrollador:** Persona encargada en desarrollar toda la aplicación. Desarrolla la parte visual (*Front-End) y toda la lógica de la aplicación y de negocio* (*Back-End)*. En este caso, solo hay un único desarrollador: Alex Pertusa Cuadrado.

- **Diseñador Gráfico:** Es la parte interesada responsable de diseñar todas las interfaces de usuario, asegurándose de que cumplan con los requisitos de la aplicación y mantengan coherencia con la línea de estilo de la empresa.

- **Testers:** Es la parte que se encarga de probar la aplicación y dar sus opiniones para ver en qué puntos es débil la aplicación.

2 Justificación

2.1 Soluciones existentes en el mercado

La empresa tecnológica ***Zendesk*** dispone de un conector que sincroniza el Dynamics 365 con su aplicación propia **[7]**. Dentro de su aplicación encontramos muchas funcionalidades que no son útiles para la empresa. Además, no es capaz de sincronizar los datos de Business Central, por lo tanto, se puede descartar como posible opción.

La empresa tecnológica **GRVPPE** ofrece el producto "*Portal Clientes para Microsoft Dynamics NAV, Business Central & AX*" **[8]**. Este dispone diferentes módulos dónde se pueden destacar el de albaranes y facturas. Aun así, se puede descartar este producto ya que no sincroniza con *Dynamics 365*.

2.2 Conclusión y justificación de la solución propia

Las soluciones disponibles en el mercado tienden a ser genéricas o requieren costosos procesos de personalización que no garantizan cubrir completamente las necesidades de la empresa. Ante la ausencia de estas posibles soluciones en el mercado que aborden de manera integral el problema específico de la empresa, se propone desarrollar una aplicación web a medida para la empresa.

La necesidad de desarrollar una solución de software que aborde los problemas actuales es fundamental para mejorar la eficiencia operativa, optimizar los tiempos de respuesta y, sobre todo, ofrecer una experiencia de cliente más satisfactoria. La fragmentación de la información entre diferentes sistemas, la desconexión entre las facturas y los servicios prestados, así como el uso de métodos de comunicación tradicionales, han generado ineficiencias que no solo afectan la operatividad interna de la empresa, sino que también impactan negativamente en la percepción de los clientes sobre la calidad del servicio prestado.

El desarrollo de un proyecto de software que integre y centralice la información de todos los sistemas implicados, brindando una visión unificada tanto a los empleados internos de la empresa como a los clientes, se justifica con los siguientes puntos:

- **Mejora en la eficiencia operativa:** Un sistema único que permitirá que los empleados accedan a toda la información relevante de cada cliente desde una sola plataforma, eliminando la necesidad de consultar múltiples fuentes y reduciendo el tiempo de respuesta del empleado en la resolución de dudas y consultas. Esto no solo mejorará la productividad del equipo administrativo y comercial, sino que también permitirá gestionar un mayor volumen de solicitudes de manera más ágil.

- **Reducción de errores y aumento de la precisión:** Al centralizar los datos, se minimizarán los errores que actualmente surgen debido a la falta de sincronización entre sistemas. Esto permitirá que tanto las facturas como los registros de los servicios prestados estén alineados, lo que facilitará que los clientes puedan verificar de manera rápida y precisa la relación entre los servicios recibidos y los cargos facturados. Una mejor claridad en este aspecto reduce la probabilidad de disputas y mejora la transparencia con los clientes.

- **Mejora en la experiencia del cliente:** Los clientes actuales demandan acceso rápido y en tiempo real a la información. Un sistema de software que ofrezca una plataforma en línea donde los usuarios puedan consultar sus servicios contratados, verificar facturas y resolver consultas sin necesidad de recurrir a llamadas o correos electrónicos mejorará significativamente su experiencia. Además, al ofrecer un canal de comunicación moderno y accesible, la empresa podrá alinear sus servicios con las expectativas tecnológicas de los clientes, fomentando una mayor satisfacción y fidelización.

- **Posicionamiento competitivo:** En un entorno empresarial cada vez más digitalizado, las empresas que no adopten tecnologías que mejoren su eficiencia y experiencia de cliente corren el riesgo de quedarse rezagadas. Implementar una solución de software que aborde los problemas actuales permitirá a la empresa mantenerse competitiva y diferenciarse de la competencia a través de un servicio más rápido, eficiente y orientado a las necesidades del cliente.

- **Escalabilidad y adaptabilidad futura:** El desarrollo de un sistema moderno y bien estructurado permitirá que la empresa sea capaz de escalar y adaptarse a futuras necesidades. La solución propuesta será flexible para que pueda integrar nuevas funcionalidades o áreas según las demandas de los clientes, permitiendo que la empresa continúe innovando y mejorando sus procesos a lo largo del tiempo.

- **Posibilidad de comercialización:** Dado que la empresa se especializa en consultoría y en la venta de productos que optimizan y facilitan el uso del ERP, esta solución podría evolucionar de una herramienta interna a un producto comercializable. Para ello, es esencial que la plataforma esté diseñada de manera flexible y estándar, permitiendo su implementación en otros clientes sin necesidad de personalizaciones adicionales. Esto abriría nuevas oportunidades de negocio, ampliando la oferta de productos y potenciando el crecimiento de la empresa en el mercado.

En conclusión, la justificación de este proyecto de software se basa en la necesidad urgente de solucionar los problemas actuales de desorganización de la información, así como en la oportunidad de mejorar la eficiencia operativa y la experiencia del cliente.

3 Alcance

Una vez descrita toda la problemática y la solución propuesta, es necesario definir el alcance del proyecto ya que disponemos de un tiempo acotado y limitado. Primero de todo se definen los objetivos que se quiere alcanzar en el proyecto. Posteriormente, se analizan todos los requisitos y se exponen los diferentes riesgos.

3.1 Objetivos

Tal y como se ha descrito en la justificación, el principal objetivo del proyecto es crear una plataforma web donde se centralice toda la información relacionada con el cliente. Para lograrlo, se han definido los siguientes subobjetivos clave para llevar a cabo el proyecto:

- Los usuarios podrán ver de manera resumida todas las facturas de venta, abonos de venta, contratos activos y finalizados, hojas de servicios y albaranes de venta.

- Los usuarios podrán descargar el pdf de un documento específico o el modelo 347 en csv.

- Los usuarios administradores del cliente podrán definir roles para sus usuarios, limitando el acceso a ciertas pantallas y vistas.

- Los usuarios podrán ver un calendario con todas las actuaciones de la empresa y solicitar un recurso en una fecha concreta.

- Los usuarios podrán ver las incidencias activas y las finalizadas.

- Los usuarios podrán abrir una incidencia, indicando el motivo correspondiente.

- La interfaz de usuario deberá replicar el diseño proporcionado por el diseñador web, asegurando que la aplicación sea intuitiva, fácil de usar y cumpla con los estándares de la empresa.

3.2 Requisitos

3.2.1 Requisitos funcionales

Desde el comité de dirección de la empresa se ha recalcado en mostrar dentro del proyecto cuatro grandes áreas: área administrativa, área de soporte, área de productos y área de infraestructura. Al tener un tiempo muy limitado solamente abordaremos el área administrativa y el área de soporte, el resto se abordarán en una segunda fase del proyecto. A continuación, se muestran las áreas con las funcionalidades a realizar dentro de cada una.

3.2.1.1 Área Administrativa

Dentro de esta área se debe mostrar toda la parte relacionada con la administración de la empresa.

Las funcionalidades para realizar son:

- Resumen de cliente: Poder ver de manera rápida el número de facturas, abonos, servicios y contratos activos.

- Gestión de facturas y abonos:
  - Poder ver y descargar en pdf las facturas y abonos de venta.
  - Poder ver el saldo pendiente.

- Gestión de contratos: Poder ver y descargar en pdf los contratos de servicio.

- Gestión de contactos: Poder ver, crear, editar y eliminar un nuevo contacto.

- Gestión de hojas de servicios: Poder ver y descargar en pdf las hojas de servicios realizadas.

- Descargar el modelo 347 en formato CSV.

- Solicitar una oferta: Poder crear una solicitud de oferta y que automáticamente cree una oportunidad de venta en el CRM para ese cliente.

3.2.1.2 Área de soporte

Es la sección dónde los clientes podrán ver todo lo relacionado con el ticketing y la gestión de calendario.

Las funcionalidades para realizar son:

- Gestión de ticketing:
  - Abrir un ticket
  - Cancelar un ticket
  - Consultar los tickets abiertos
  - Consultar el histórico de tickets cerrados
  - Integrar un KPI con datos relevantes

- Gestión de calendario:
  - Poder solicitar un técnico o consultor en una fecha
  - Poder ver el historial de intervenciones en una fecha

3.2.2 Requisitos no funcionales

Seguridad: Es fundamental garantizar la integridad y privacidad de los datos, ya que estamos utilizando información sensible.

Escalabilidad: La aplicación debe ser capaz de poder ser escalable en un futuro cuando se amplíen los módulos.

Usabilidad: La interfaz de usuario debe de ser simple e intuitiva para el usuario. Además, la imagen de la aplicación debe coincidir con los estilos de la empresa.

Rendimiento: La aplicación tiene que ser capaz de trabajar con un gran volumen de usuarios concurrentes al mismo tiempo.

3.2.3 Riesgos

En cualquier proyecto hay que tener en cuenta diferentes escenarios negativos y riesgos que pueden ir apareciendo a lo largo del proceso.

Actualizaciones de la API: Al tener que recoger información y datos de dos softwares distintos a través de API, hay que ser consciente que los proveedores de dichas APIs pueden lanzar nuevas versiones de los *endpoints*[1] o sustituirlos por otros.

Inexperto en las tecnologías utilizadas: Al utilizar tecnologías que el desarrollador no ha tenido experiencia con ellas puede provocar que el inicio del proyecto sea algo más lento ya que previamente ha tenido que adquirir conocimiento sobre ellas.

Bugs: Dentro del mundo de la programación siempre hay que tener en cuenta que pueden aparecer bugs. Estos pueden surgir debido a equivocaciones durante el proceso de desarrollo o como resultado de problemas inesperados al instalar y compilar diferentes tecnologías.

[1] Endpoint: Una URL específica que define un punto de acceso a través del cual se puede interactuar con un sistema para enviar o recibir datos.

4 Metodología

4.1 Metodología de trabajo

En este proyecto, se implementará la metodología Agile mediante la técnica de Scrum. Se ha seleccionado Scrum debido a su capacidad para adaptarse a cambios en las funcionalidades del proyecto, permitiendo una respuesta eficaz a cualquier ajuste necesario.

Scrum **[9]** se centra en el trabajo colaborativo y en la capacidad de adaptarse a cualquier cambio durante el proyecto. Se utilizan *sprints* que son períodos de trabajo de unas dos a cuatro semanas donde el equipo se enfoca en completar ciertas tareas. Al comenzar un sprint, el equipo escoge tareas de una lista principal llamada *Product Backlog*, que son las cosas que necesitan hacer. Estas tareas seleccionadas forman el *Sprint Backlog*.

Cada día, el equipo tiene una breve reunión llamada *Daily Scrum* para hablar sobre el progreso y resolver problemas. Esto ayuda a todos a estar al día y ajustar lo que están haciendo si es necesario.

Al final del sprint, el equipo muestra lo que ha hecho en una reunión de revisión del sprint. Aquí, pueden recibir comentarios para saber si van por buen camino. Luego, tienen una reunión llamada retrospectiva del sprint para pensar en cómo pueden trabajar mejor en el próximo sprint. Esta forma de trabajar en ciclos cortos ayuda al equipo a mejorar constantemente y a adaptarse rápidamente a cualquier cambio.

La estructura básica incluye tres roles: Scrum Master, Product Owner y el equipo de desarrollo.

El Scrum Master es el líder y mentor del equipo de trabajo. Este se asegura de que el equipo siga todos los principios y prácticas de Scrum. Su objetivo es optimizar la productividad y el rendimiento del equipo.

El Product Owner es la persona encargada de hablar con el cliente y de representar la voz del cliente a nivel interno. Además, es el rol encargado de gestionar el *Product BackLog,* que es una lista priorizada con los requisitos del proyecto.

El equipo de desarrollo es el grupo de personas o la persona que se encarga de desarrollar el producto con los requisitos impuestos.

4.2 Gestión del repositorio

Para la gestión del código de este proyecto utilizaremos *Git*. Este nos facilita la colaboración entre desarrolladores permitiéndoles trabajar juntos en el proyecto, registrar las modificaciones en el código y restaurar versiones anteriores a través de las ramas.

En relación con las metodologías de trabajo utilizando *Git*, implementaremos *Gitflow* **[10]**, un modelo de ramificación bien estructurado que define un entorno claro para el desarrollo de software. Este enfoque especifica varias ramas y establece directrices claras para su manejo, lo que mejora el proceso de trabajo en proyectos de software. A continuación, describimos las principales ramas utilizadas en *Gitflow*:

Rama *master*: Esta rama alberga la versión estable del proyecto. Esta es la rama que se implementa en producción.

Rama *develop*: Esta rama actúa como la rama principal de desarrollo, donde se integran y prueban todas las nuevas características desarrolladas.

Ramas de funcionalidades: Se crea una nueva rama separada para cada nueva funcionalidad o característica que se desarrolla, partiendo de la rama *develop*. Una vez que la característica está completa, se reintegra a la rama *develop*.

Ramas de *hotfix*: Estas ramas están destinadas a corregir errores que se encuentran en producción y hay que realizar un cambio en producción de manera urgente. Estas ramas se crean directamente desde la rama master y una vez se ha arreglado el error se fusionan de nuevo tanto con la rama master como con la rama *develop* para tener los el *fix* en desarrollo.

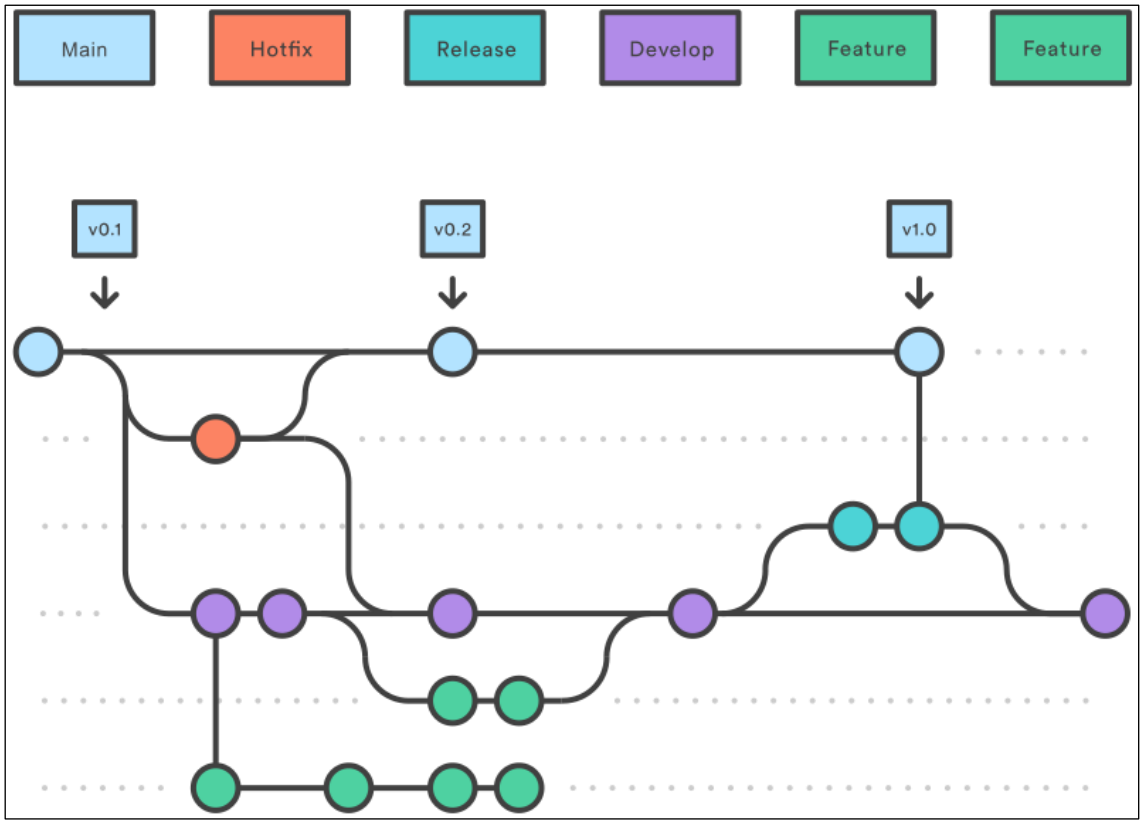


Figura 1: Diagrama de ramas de GitFlow. Fuente: Atlassian

4.3 Riesgos

Al utilizar la metodología Scrum, ya se contempla un proceso específico para gestionar bugs o errores. Dentro del *Product Backlog*, es posible marcar una tarea como error, lo que permite identificarla claramente y discutirla en las reuniones del equipo. A partir de ahí, se tomará una decisión sobre cómo abordarla, lo que puede implicar ajustes en las prioridades del sprint actual o futuros. Este enfoque asegura que los errores se gestionen de manera estructurada, sin interrumpir el flujo de trabajo.

4.4 Herramientas de trabajo

4.4.1 Azure Devops

Esta herramienta de *Microsoft* permite trabajar en un proyecto utilizando la técnica de Scrum e incluye la gestión del repositorio en la misma plataforma. De hecho, se puede enlazar una rama o un *commit* a una cierta tarea para lograr una mayor trazabilidad. Desde esta plataforma se puede validar el estado de todas las tareas y visualizar el progreso del proyecto, asegurando que avanza de manera correcta. Una vez acabado el proyecto, se podrá analizar si se han cumplido todos los objetivos o queda alguno pendiente por desarrollar ya que la información estará guardada en la base de datos.

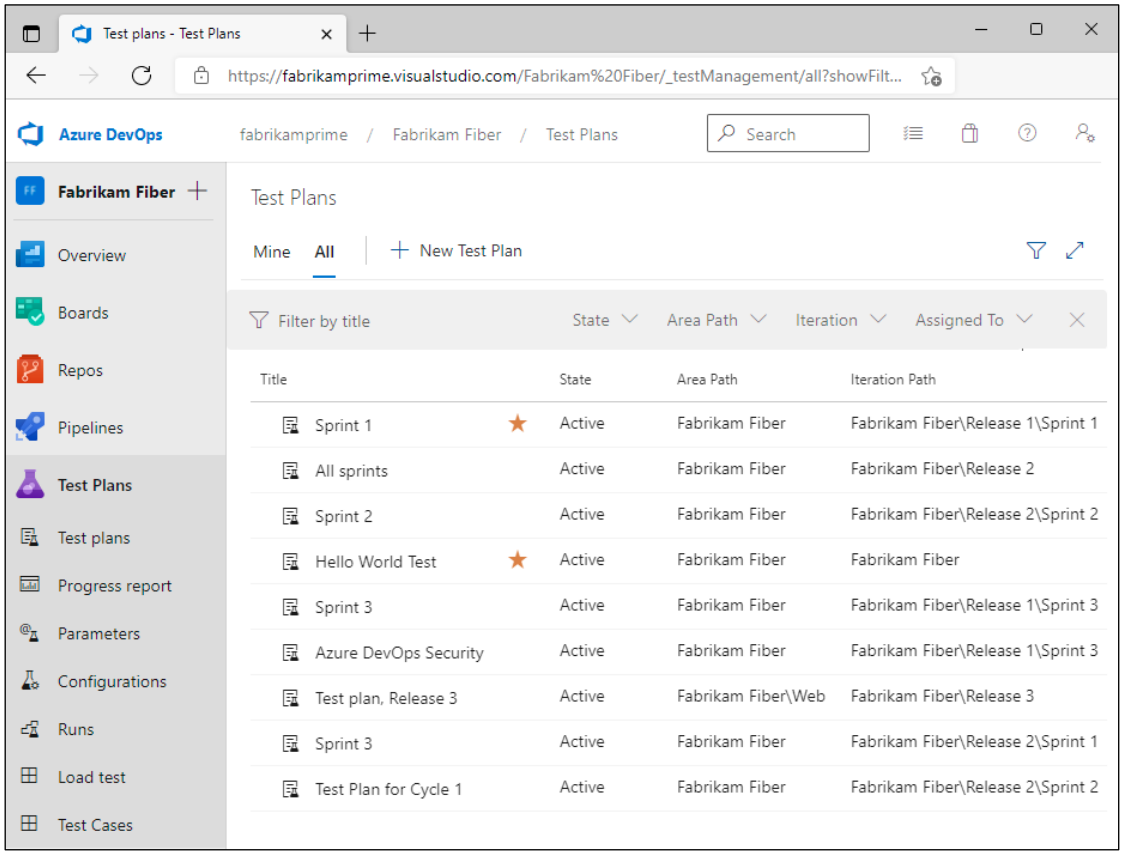


Figura 2: Captura de Pantalla de Azure Devops. Fuente: Microsoft Learn

4.4.2 Microsoft Teams

Esta herramienta de Microsoft permite la comunicación con los miembros del equipo y con el comité de dirección de la empresa.

Figura 3: Logo de Microsoft Teams. Fuente: Microsoft

5 Planificación temporal

La planificación del proyecto es una de las claves para que este tenga éxito. Dentro de la planificación se definen las distintas tareas a realizar, se analizan los distintos recursos disponibles para abordar el proyecto y se generan estimaciones para predecir la duración del proyecto y sus etapas. Además, en la planificación también se requiere la anticipación a posibles riesgos y la preparación de estrategias o alternativas para poder mitigarlos o solucionarlos.

5.1 Roles

Antes de describir las tareas, es esencial definir los roles que tendrá el equipo de trabajo. Aunque en este caso, el autor será el que desempeñe todos los roles.

- **Gestor de Proyecto:** Responsable de planificar, supervisar y garantizar que el proyecto se realiza y avanza con éxito. Se asegura de que se respeten los plazos de entrega y el presupuesto establecido. También se encarga de coordinar al equipo y gestionar los recursos.

- **Arquitecto de Software:** Persona encargada de diseñar la base de datos y la arquitectura física de la aplicación, asegurando que la solución propuesta sea técnicamente viable y cumpla con los requisitos del proyecto.

- **Desarrollador *Frontend:*** Responsable de la implementación de la interfaz visual de la aplicación, siguiendo los diseños proporcionados por el proveedor.

- **Desarrollador Backend:** Persona encargada de desarrollar la lógica de negocio de la aplicación, implementando los procesos y las funcionalidades que se ejecutan en el servidor. Debe asegurar que todo funcione según los criterios establecidos por el arquitecto de software y el gestor de proyectos.

- **Testers:** Persona o equipo encargado de realizar pruebas exhaustivas de la aplicación. Su misión es detectar errores, identificar casos extremos no contemplados y proporcionar *feedback* sobre posibles mejoras.

5.2 Descripción de tareas

La descripción de las tareas es esencial en un proyecto ya que son los elementos básicos del proyecto. Cada tarea representa una actividad específica que debe llevarse a cabo para alcanzar los objetivos del proyecto. Dividir el proyecto en tareas facilita la planificación y permite gestionar de manera más eficiente el seguimiento del progreso.

5.2.1 Tareas de gestión del proyecto (GP)

Dentro de la gestión del proyecto, se encuentran todas las tareas relacionadas con la planificación y seguimiento del proyecto. Estas tareas son necesarias para tener controlado en todo momento el seguimiento del proyecto, entre ellas se encuentran las tareas relacionadas con la documentación, la comunicación con el equipo y la preparación de la defensa del proyecto.

- **GP1 - Contextualización y alcance**
  - Primera fase de la documentación, en esta se redacta un documento dónde se especifica el contexto del proyecto, el alcance, los objetivos a lograr, la metodología a seguir, la justificación de la solución propuesta, los requisitos, y los posibles riesgos.
  - Estimación: **30 horas**
  - Dependencias: **-**
  - Roles que participan: Gestor de proyectos y Arquitecto de Software

- **GP2 - Planificación temporal**
  - Segunda fase de la documentación, es esta se amplía el documento inicial especificando las tareas a realizar durante el proyecto, la estimación de horas, las dependencias entre tareas, un diagrama de Gantt con la planificación temporal, la gestión de recursos y la manera de gestionar los riesgos.
  - Estimación: **25 horas**
  - Dependencias: **GP1**
  - Roles que participan: Gestor de proyectos

- **GP3 - Presupuesto y sostenibilidad**
  - Tercera fase de la documentación de la asignatura de GEP. Se amplía el documento especificando una estimación económica del proyecto y un análisis de la sostenibilidad.
  - Estimación: **20 horas**
  - Dependencias: **GP2**
  - Roles que participan: Gestor de proyectos

- **GP4 - Final de la documentación**
  - Última fase de documentación, en esta se corrigen todos los errores y mejoras aconsejadas por el profesor y se entrega el documento.
  - Estimación: **15 horas**
  - Dependencias: **GP3**
  - Roles que participan: Gestor de proyectos

- **GP5 - Reuniones semanales**
  - Reuniones semanales con el tutor de empresa o el comité de dirección de la empresa con el objetivo de enseñar el progreso de la aplicación. En estas reuniones, también se pueden especificar nuevos requisitos funcionales.
  - Estimación: **20 horas**
  - Dependencias: **-**
  - Roles que participan: Gestor de proyectos

- **GP6 - Documentación de la memoria**
  - Se redacta el documento final de la memoria del proyecto.
  - Estimación: **100 horas**
  - Dependencias: **GP4**
  - Roles que participan: Gestor de proyectos

5.2.2 Tareas de desarrollo (D)

Al tener definidos los requisitos funcionales estructurados por áreas, se han estructurado las tareas de desarrollo por áreas.

5.2.2.1 Incepción

- **I1 - Especificación del sistema**
  - Definir los casos de uso e historias de usuario dentro del *Azure Devops.*
  - Estimación: **16 horas**
  - Dependencias: **GP1**
  - Roles que participan: Gestor de proyectos y Arquitecto de Software

- **I2 - Diseño de la base de datos**
  - Definir la estructura y el tipo de base de datos a utilizar.
  - Estimación: **10 horas**
  - Dependencias: **I1**
  - Roles que participan: Gestor de proyectos y Arquitecto de Software

- **I3 - Formación de la tecnología utilizada**
  - Al utilizar una tecnología que no es habitual para el desarrollador, hay que realizar una pequeña formación para entender y aprender las diferentes peculiaridades propias de la tecnología.
  - Estimación: **8 horas**
  - Dependencias: **GP1**
  - Roles que participan: Desarrollador *Frontend* y Desarrollador *Backend*

- **I4 - Creación del proyecto**
  - Crear el repositorio dentro del *Azure Devops* para alojar el código y conectar el *backend* con el *frontend*.
  - Estimación: **8 horas**
  - Dependencias: **I3**
  - Roles que participan: Desarrollador *Frontend* y Desarrollador *Backend*

5.2.2.2 Gestión de usuarios

- **GU1 - Autenticación**
  - Desarrollar las funcionalidades relacionadas con la autenticación de usuarios.
  - Estimación: **16 horas**
  - Dependencias: **I4**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GU2 - Roles y permisos**
  - Desarrollar las funcionalidades relacionadas con la gestión de permisos y roles de la aplicación.
  - Estimación: **16 horas**
  - Dependencias: **GU1**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

5.2.2.3 Gestión del área de administración

- **GAD1 - Gestión de facturas de venta**
  - Desarollar la funcionalidad de poder ver las facturas de venta en la aplicación y poder descargar el pdf de la factura.
  - Estimación: **14 horas**
  - Dependencias: **GU1**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GAD2 - Gestión de abonos de venta**
  - Implementar la funcionalidad de poder ver los abonos de venta en la aplicación y poder descargar el pdf del abono.
  - Estimación: **14 horas**
  - Dependencias: **GAD1**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GAD3 - Gestión de contactos**
  - Desarrollar la funcionalidad de crear, editar y eliminar un contacto de la aplicación.
  - Estimación: **14 horas**
  - Dependencias: **GAD2**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GAD4 - Gestión de contratos**
  - Implementar la funcionalidad de poder ver los contratos activos y vencidos en la aplicación y poder descargar el pdf del contrato.
  - Estimación: **14 horas**
  - Dependencias: **GAD3**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GAD5 - Gestión de hojas de servicios**
  - Implementar la funcionalidad de poder ver los contratos activos y vencidos en la aplicación y poder descargar el pdf del contrato.
  - Estimación: **12 horas**
  - Dependencias: **GAD4**
  - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GAD6 – Modelo 347**
  - Desarrollar la funcionalidad de descargar el Modelo 347 en CSV
  - Estimación: **20 horas**
  - Dependencias: **GAD5**
  - Roles que participan: Desarrollador *Frontend* y Desarrollador *Backend*

- **GAD7 - Solicitud de oferta**
    - Desarrollar la funcionalidad de poder solicitar una oferta con un formulario para expresar los requerimientos necesarios.
    - Estimación: **12 horas**
    - Dependencias: **GAD6**
    - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

5.2.2.4 Gestión del área de soporte

- **GAS1 - Abrir un ticket**
    - Desarrollar la funcionalidad de abrir un ticket para reportar un problema o incidencia.
    - Estimación: **12 horas**
    - Dependencias: **GAD**
    - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GAS2 - Cerrar un ticket**
    - Desarrollar la funcionalidad de cerrar un ticket abierto que ya ha sido solucionado por el propio cliente sin la necesidad de intervenir la sección de soporte.
    - Estimación: **12 horas**
    - Dependencias: **GAS1**
    - Roles que participan: Desarrollador *Frontend* y Desarrollador *Backend*

- **GAS3 - Consultar tickets**
    - Desarrollar la funcionalidad de ver los tickets abiertos o el histórico de tickets cerrados.
    - Estimación: **10 horas**
    - Dependencias: **GAS2**
    - Roles que participan: Desarrollador *Frontend* y Desarrollador *Backend*

- **GAS4 - Consultar estadísticas**
    - Integrar un informe de *Power Bi* para que el cliente pueda consultar las estadísticas relacionadas con los tickets.
    - Estimación: **12 horas**
    - Dependencias: **GAS3**
    - Roles que participan: Desarrollador *Frontend* y Desarrollador *Backend*

5.2.2.5 Gestión del calendario

- **GC1 - Historial de actuaciones**
    - Desarrollar la funcionalidad de poder ver el historial de actuaciones de un consultor o técnico en un calendario.
    - Estimación: **30 horas**
    - Dependencias: **GAS**
    - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **GC2 - Solicitar un recurso**
    - Desarrollar la funcionalidad de poder solicitar un recurso (técnico o consultor) en una fecha en concreto.
    - Estimación: **20 horas**
    - Dependencias: **GC1**
    - Roles que participan: Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

5.2.3 Tareas de cierre de proyecto (CP)

La finalización o cierre de un proyecto es una fase crucial dentro de cualquier proyecto. Durante esta etapa, se llevan a cabo diversas tareas que garantizan que el proyecto se finalice de manera organizada y conforme a los objetivos planteados.

- **CP1 - Testeo**
    - Tarea en la que se testea la aplicación y se intentan encontrar bugs y fallos para arreglarlos.
    - Estimación: **20 horas**
    - Dependencias: **D y GP**
    - Roles que participan: Principalmente el tester, aunque en menor medida participarán todos los roles.

- **CP2 - Despliegue**
    - Desplegar y poner en marcha la aplicación en producción
    - Estimación: **20 horas**
    - Dependencias: **CP1**
    - Roles que participan: Gestor de Proyectos, Arquitecto de Software, Desarrollador *Frontend* y Desarrollador *Backend*

- **CP3 - Defensa del proyecto**
    - Preparación del material visual para defender el proyecto de manera oral delante del tribunal.
    - Estimación: **20 horas**
    - Dependencias**: CP2**
    - Roles que participan: Gestor de proyectos

5.3 Recursos

Para completar las tareas descritas anteriormente, los recursos son fundamentales, ya que proporcionan el soporte necesario para llevar a cabo cada actividad del proyecto de manera efectiva. A continuación, se mencionan los recursos necesarios para que el proyecto pueda realizarse de manera correcta.

5.3.1 Recursos humanos

- **Alex Pertusa Cuadrado [P1]:** Alumno que realiza el proyecto de final de grado. Actúa como gestor de proyecto, arquitecto de software, desarrollador *front-end*, desarrollador *back-end*.

- **Nil Samper Más [P2]:** Persona encargada en asesorar al estudiante, supervisar el trabajo realizado y brindar orientación para que el proyecto avance de manera correcta y se complete con éxito.

- **Empresa proveedora de diseños web [PV1]:** Empresa que suministra los diseños web del proyecto, teniendo en cuenta los requisitos especificados.

- **Comité de dirección [DIR]:** Grupo conformado por todos los directores de las distintas áreas de la empresa y dueños de la empresa.

5.3.2 Recursos materiales

- **Ordenador portátil [M1]:** Ordenador portátil con el que se realiza tanto la documentación como el desarrollo del proyecto. Dispone de un sistema operativo Windows 11 y cuenta con cámara, micrófono, conexión a Internet.

  Si el trabajo se realiza desde la oficina, se dispone también de dos pantallas, ratón y teclado para mejorar la comodidad y productividad.

5.3.3 Recursos de software

- **Microsoft Office 2024 [S1]:** Se aprovecharán diferentes aplicaciones dentro del catálogo de *Microsoft* para la realización del proyecto. Para redactar el documento se utilizará *Word*. Para preparar el contenido audiovisual en la defensa del proyecto se utilizará *PowerPoint*. Para guardar el proyecto en la nube y poder editarlo desde cualquier dispositivo se utilizará *OneDrive*. Para la comunicación se utilizará *Teams* y *Outlook*. **[11]**

- **GanttProject [S2]:** Herramienta que permite crear diagramas de Gantt. **[12]**

- **Draw.io [S3]:** Herramienta que permite crear casos de uso y diagramas de secuencia. **[13]**

- **DBeaver [S4]:** Herramienta que permite interactuar con una base de datos SQL. **[14]**

- **Azure Devops [S5]:** Aplicación software que permite gestionar el proyecto siguiendo los métodos Scrum. Además, el código se alojará en el repositorio del propio proyecto dentro de la herramienta, permitiendo la gestión de versiones. **[15]**

- **Clouding.io [S6]:** Plataforma que ofrece servidores en la nube dónde alojaremos la aplicación. **[16]**

- **Visual Studio [S7]:** IDE para desarrollar tanto el frontend como el backend. **[17]**

- **Postman [S8]:** Herramienta para poder crear y testear llamadas a las APIs. **[18]**

5.4 Estimaciones

En la Tabla 1, se observa de forma resumida el desglose de horas y recursos de cada tarea. A nivel de recursos, ya que no aparece explícitamente en la tabla, recalcar que todas las tareas utilizarán el recurso **[P1]** y **[M1]**.

Código	Tarea	Horas	Dependencias	Recursos
	Gestión del Proyecto	**Total: 210**		
GP1	Contextualización y alcance	30	-	S1
GP2	Planificación temporal	25	GP1	S1,S2
GP3	Presupuesto y sostenibilidad	20	GP2	S1
GP4	Final de la documentación	15	GP3	S1
GP5	Reuniones semanales	20	-	S1, P2, DIR
GP6	Documentación de la memoria	100	GP4	S1,S2,S3
D	**Desarrollo**	**Total: 270**		
I	**Incepción**	**Total: 42**		
I1	Especificación del sistema	16	GP1	S4,S5, PV1
I2	Diseño de la base de datos	10	I1	S4
I3	Formación de la tecnología utilizada	8	I2	S7
I4	Creación del proyecto	8	I3	S4,S5,S7,S8
GU	**Gestión de Usuarios**	**Total: 32**		
GU1	Autenticación	16	I4	S4,S5,S7,S8
GU2	Roles y permisos	16	GU1	S4,S5,S7,S8
GAD	**Gestión del área de administración**	**Total: 100**		
GAD1	Gestión de facturas de venta	14	GU	S4,S5,S7,S8
GAD2	Gestión de abonos de venta	14	GAD1	S4,S5,S7,S8
GAD3	Gestión de contactos	14	GAD2	S4,S5,S7,S8
GAD4	Gestión de contratos	14	GAD3	S4,S5,S7,S8
GAD5	Gestión de hojas de servicios	12	GAD4	S4,S5,S7,S8
GAD6	Modelo 347	20	GAD5	S4,S5,S7,S8
GAD7	Solicitud de oferta	12	GAD6	S4,S5,S7,S8
GAS	**Gestión del área de soporte**	**Total: 46**		
GAS1	Abrir un ticket	12	GAD	S4,S5,S7,S8
GAS2	Cerrar un ticket	12	GAS1	S4,S5,S7,S8
GAS3	Consultar tickets	10	GAD	S4,S5,S7,S8
GAS4	Consultar estadísticas	12	GAD	S4,S5,S7,S8
GC	**Gestión del calendario**	**Total: 50**		
GC1	Historial de actuaciones	30	GAS	S4,S5,S7,S8
GC2	Solicitar un recurso	20	GC1	S4,S5,S7,S8
CP	**Cierre de Proyecto**	**Total: 40**		
CP1	Testeo	20	D	P2,DIR
CP2	Despliegue	20	CP1	S6
CP3	Defensa del proyecto	20	CP2 y GP6	S1
	Resumen Proyecto	**Total: 520**		

Tabla 1: Estimación de horas y recursos por tarea. Fuente: Propia

5.5 Diagrama de Gantt

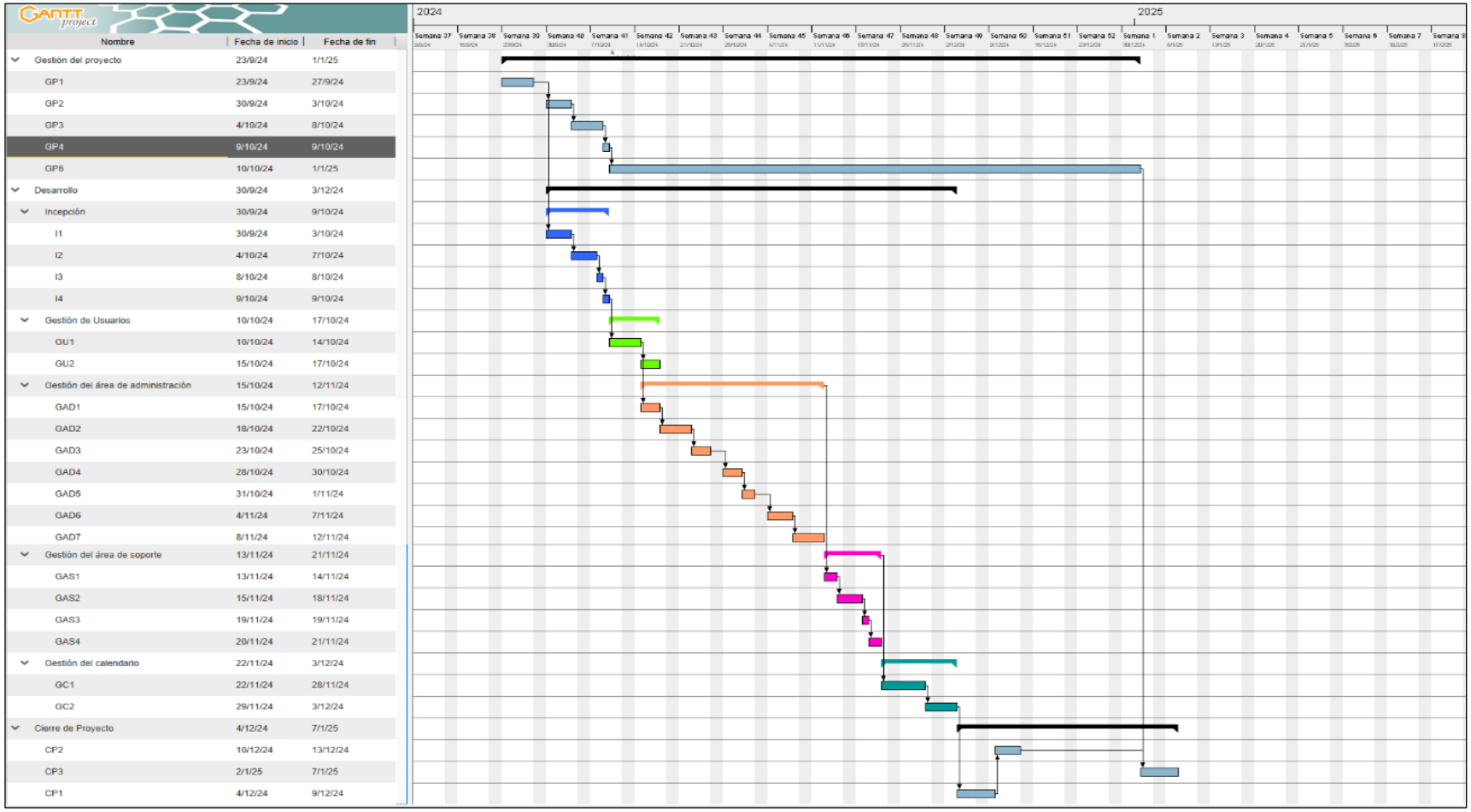


Figura 4: Diagrama de Gantt. Fuente: Propia

5.6 Gestión del Riesgo

En el apartado 3.2.3, se han definido una serie de posibles riesgos dentro del proyecto. Como en cualquier ámbito de la vida, en la programación también hay que estar preparado para actuar por si algunos de estos errores o riesgos aparecen. Antes de pensar en el plan alternativo, se ha analizado, en la Tabla 2, el impacto que tiene ese riesgo en el proyecto y la probabilidad que hay de que aparezca.

Riesgo	Impacto	Probabilidad
Actualizaciones de la API	Muy Alto	10%
Inexperto en las tecnologías utilizadas	Moderado	20%
Bugs	Moderado	40%

Tabla 2: Análisis del impacto y la probabilidad de los riesgos. Fuente: Propia

Al ver que el impacto puede afectar negativamente al desarrollo y finalización del proyecto se han propuesto diferentes soluciones y planes alternativos para poder abordarlos de manera eficiente.

5.6.1 Actualizaciones de la API

Para abordar este riesgo, dentro de la estimación de las tareas de desarrollo ya se ha dado más margen por si *Microsoft* decide cambiar las *APIs.*

Existe una posibilidad remota de que *Microsoft* pueda eliminar la API, este hecho provocaría un cambio radical en la aplicación o incluso la cancelación del proyecto.

En caso de que la API pasará a ser de pago, la empresa está dispuesta a asumir los costes del consumo de la API.

5.6.2 Inexperto en las tecnologías utilizadas

Para solucionar este riesgo se ha creado la tarea I3, esta tarea consiste en realizar una formación de las diversas tecnologías que se emplearán en el proyecto. Adicionalmente, se han ajustado los tiempos de algunas tareas de desarrollo, sobrestimándolos ligeramente para permitir formación adicional durante la fase de desarrollo si fuera necesario.

5.6.3 Bugs

Al tener experiencia en diferentes proyectos de Software y poder esperar que salgan errores o bugs, en las tareas de desarrollo se ha dado un pequeño margen para poder abordar los bugs. Además, también se ha añadido tiempo en la fase de testeo de la aplicación para poder corregir bugs que se detectan al final. En caso, de que el bug fuera muy grande y no se pudiera resolver de manera rápida, se creará una tarea específica para solucionar el error. Este hecho no afectará a la planificación.

6 Gestión económica

Una vez realizada toda la planificación temporal y la gestión de riesgos, en un proyecto de software es crucial analizar la parte económica. Esto implica evaluar los costos asociados con el desarrollo, implementación y mantenimiento del software, considerando tanto los recursos humanos como tecnológicos. Se deben calcular los gastos relacionados con la adquisición de licencias, herramientas de desarrollo, infraestructuras de hardware y servicios en la nube, así como los salarios del equipo de trabajo. Además, es esencial prever un margen para imprevistos y posibles ajustes durante el ciclo de desarrollo. Una buena gestión económica garantizará que el proyecto avance dentro del presupuesto y que los fondos se utilicen de manera eficiente.

6.1 Identificación de los costes

6.1.1 Costes de personal por actividad

Dentro de un proyecto software se encuentran diferentes roles los cuales tienen distinto coste. En este caso, tal y como se ha definido en la sección 5.3.1, el autor del proyecto actúa como gestor de proyecto [GP], arquitecto de software [AS], desarrollador *back-end* [DB], desarrollador *front-end* [DF] y *testers* [T]. Al tratarse de un desarrollador junior que asume todos estos roles, el coste para la empresa es considerablemente menor en comparación con la contratación de múltiples especialistas para cada uno de los puestos. Sin embargo, para reflejar una simulación más realista del coste del proyecto, la empresa ha proporcionado los datos salariales correspondientes a cada uno de estos roles. De esta manera, se puede estimar de manera más precisa cuál sería el coste del proyecto si se utilizaran recursos especializados en cada área.

En la Tabla 3 se puede observar el desglose de costes por rol y el coste del autor del proyecto.

Rol	Salario bruto / hora	Salario bruto / hora + Seguridad Social
Gestor de proyecto	24,04 €	31,25 €
Arquitecto software	21,63 €	28,12 €
Desarrollador *front-end* senior	14,42 €	18,75 €
Desarrollador *back-end* senior	16,83 €	21,88 €
Testers	12,02 €	15,63 €

Tabla 3. Costes por rol. Fuente: Propia

Código	Tarea	Horas	Horas de cada rol					Coste (€)
			GP	AS	DF	DB	T	
GP1	Contextualización y alcance	30	25	5				921,90 €
GP2	Planificación temporal	25	25					781,30 €
GP3	Presupuesto y sostenibilidad	20	20					625,04 €
GP4	Final de la documentación	15	15					468,78 €
GP5	Reuniones semanales	20	20					625,04 €
GP6	Documentación de la memoria	100	100					3.125,20 €
I1	Especificación del sistema	16	4	12				462,44 €
I2	Diseño de la base de datos	10	2	8				287,46 €
I3	Formación de la tecnología utilizada	8			4	4		162,50 €
I4	Creación del proyecto	8			4	4		162,50 €
GU1	Autenticación	16		2	4	10		350,01 €
GU2	Roles y permisos	16		4	4	8		362,49 €
GAD1	Gestión de facturas de venta	14		2	4	8		306,25 €
GAD2	Gestión de abonos de venta	14		2	4	8		306,25 €
GAD3	Gestión de contactos	14		4	4	6		318,73 €
GAD4	Gestión de contratos	14		2	4	8		306,25 €
GAD5	Gestión de hojas de servicios	12		2	4	6		262,50 €
GAD6	Modelo 347	20			4	16		425,05 €
GAD7	Solicitud de oferta	12		2	5	5		259,36 €
GAS1	Abrir un ticket	12		2	4	6		262,50 €
GAS2	Cerrar un ticket	12			4	8		250,02 €
GAS3	Consultar tickets	10			7	3		196,86 €
GAS4	Consultar estadísticas	12			8	4		237,48 €
GC1	Historial de actuaciones	30		4	16	10		631,20 €
GC2	Solicitar un recurso	20		2	6	12		431,26 €
CP1	Testeo	20	2	2	2	2	12	387,50 €
CP2	Despliegue	20	14	2	2	2		575,02 €
CP3	Defensa del proyecto	20	20					625,04 €
Total		**540**	**247**	**57**	**94**	**130**	**12**	**14.115,93 €**

Tabla 4. Costes por actividad y rol. Fuente: Propia

No obstante, al tener un proveedor que es especialista en la parte de diseño de UIX, hay que sumar el coste del servicio en la tarea I1. En este caso, se han presupuestado 2000€ por el proyecto. Por lo tanto, hay que sumar ese coste al total de la Tabla 4.

Coste total por actividad = 14.115,93€ + 2000€ = 16.115,93€

6.1.2 Costes genéricos

A nivel de costes, también se tienen que incluir los costes fijos relacionados con el proyecto. Dentro de estos se incluyen gastos como el hardware necesario para la realización del proyecto, las licencias de los distintos softwares utilizados y el coste del espacio de trabajo.

6.1.2.1 Hardware

Para poder desarrollar el proyecto, se necesita de material hardware.

Al ser recursos de Hardware, Hacienda nos indica un periodo de 4 años para su amortización. Todos los recursos se han comprado nuevos, es decir, que todos los recursos pueden ser amortizados. Para la amortización, se ha utilizado el siguiente cálculo:

$$\textbf{Amortización}\ (€) = \frac{Coste\ hardware\ (€)\ *\ Dedicación\ al\ proyecto\ (Horas)}{Vida\ Util\ (Años) *\ Días\ laborables\ (Días) *\ Dedicación\ diaria\ (Horas)}$$

La vida útil, como se ha comentado anteriormente, es de 4 años. Los días laborables, en 2024 en Cataluña, son 251 **[19]** y la dedicación diaria es de 6 horas. La dedicación al proyecto es de 540 horas. En la Tabla 5 se observa el resultado del coste total y de la amortización.

Recurso	Coste	Amortización
MSI Prestige 16 (Ordenador portátil)	1.671,52 €	149,84 €
LG 24BK55YP-B 23.8" (Pantalla 1)	93,22 €	8,36 €
LG 24BK55YP-B 23.8" (Pantalla 2)	93,22 €	8,36 €
Logitech MK120 (Teclado y Ratón)	15,81 €	1,42 €
Total	**1.873,77 €**	**191,64 €**

Tabla 5. Coste del Hardware. Fuente: Propia

6.1.2.2 Software

A nivel de software la mayoría de las herramientas y programas para desarrollar el proyecto son gratis, excepto *Microsoft Office 2024* y *Clouding.io.*

El coste de *Microsoft Office 2024* es ínfimo ya que la empresa es *Partner* oficial de *Microsoft* y dispone de muchas licencias a un coste muy bajo. Por lo tanto, no se considerará un coste para el proyecto.

Por otro lado, la empresa ya tiene varias aplicaciones internas o de clientes alojadas en la plataforma *Clouding.io*, por lo tanto, el coste de desplegar la aplicación va a ser muy pequeño porque va a compartir espacio con otras aplicaciones. Por ese hecho, tampoco se considerará un coste para el proyecto.

6.1.2.3 Espacio de trabajo

Al llevar a cabo el proyecto en las instalaciones de la empresa, es importante considerar el coste asociado. Aunque la empresa cuenta con tres salas disponibles, solo se considerará el coste de una, ya que el autor del proyecto utiliza únicamente una de ellas para su trabajo. Para calcular el coste de la oficina se ha utilizado la siguiente formula:

$$\text{Coste oficina} = \text{Coste por mes} * \text{Número de meses}$$

El coste por mes es de 497€, y el proyecto se desarrolla durante un periodo de 4 meses. Por lo tanto, el coste de la oficina es de **1988€**.

6.1.3 Contingencias

En proyectos de desarrollo de software, se estima una contingencia del 10%. Por lo tanto, es necesario recalcular tanto los costes de personal como los costes generales, aplicando este porcentaje adicional. Se puede observar el resultado del coste total en la Tabla 6.

Tipo de Coste	Coste	% Contingencia	Contingencia	Coste total
Personal	16.115,93 €	10	1.611,59 €	17.727,52 €
Genéricos	2.179,64 €		217,96 €	2.397,60 €
Total	**18.295,57 €**		**1.829,56 €**	**20.125,13 €**

Tabla 6. Costes con contingencias. Fuente: Propia

6.1.4 Imprevistos

Dentro del proyecto, pueden surgir imprevistos que afecten al coste total. Entre estos imprevistos se encuentran los riesgos previamente definidos, los costes relacionados con los recursos personales, de material y de oficina.

Para los riesgos ya se ha añadido tiempo extra a las tareas críticas, por lo que esos costes extras ya están incluidos en el coste de personal.

En relación a los costes de los recursos materiales, se debe contemplar la posibilidad de tener en cuenta avería del portátil, que requeriría una reparación, o en las pantallas, en cuyo caso sería necesario reemplazarlas por una nueva.

Respecto al coste de oficina, si surgiera algún problema, la empresa dispone de salas de *Coworking* [2] gratuitas. En caso de un problema muy grave, el proyecto se podría seguir desarrollando desde casa sin inconvenientes.

En la Tabla 7, se puede observar el coste total de los imprevistos.

Elemento	Coste imprevisto	% Posibilidad	Coste
Portátil	200 €	5	10,00 €
Pantalla	93,22 €	5	4,66 €
Total	**293 €**		**14,66 €**

Tabla 7. Costes por Imprevistos. Fuente: Propia

[2] Coworking: Espacio de trabajo compartido para autónomos, PYMES, startups.

6.1.5 Presupuesto

El presupuesto del proyecto es la suma de los costes previamente mencionados. En la Tabla 8 se detallan los costes desglosados por secciones, y el coste total del proyecto que asciende a los **20.139,79€**.

Tipo de Coste	Coste
Personal por Actividad	16.115,93 €
Contingencias	1.829,56 €
Genéricos	2.179,64 €
Imprevistos	14,66 €
Total	**20.139,79 €**

Tabla 8. Coste total del proyecto. Fuente: Propia

6.2 Control de gestión

La gestión de desviaciones es un aspecto clave, ya que permite evaluar continuamente su progreso y detectar cualquier desviación respecto al plan original. Este control es fundamental para asegurar que el proyecto se mantenga dentro de los parámetros establecidos de tiempo y presupuesto.

Para gestionar estas desviaciones de manera efectiva, se utilizarán las siguientes métricas:

- **Desviación de horas consumidas por tarea**
  (Horas estimadas – Horas Reales) * Coste por hora de la tarea

- **Desviación de costes según las horas consumidas por tarea**
  (Horas estimadas – Horas Reales) * Coste real de la tarea

- **Porcentaje de tareas completadas**
  (Tareas completadas / Tareas pendientes de completar) * 100

- **Desviación total de los costes**
  Horas estimadas – Horas Reales

- **Desviación total de los costes**
  Coste total estimado – Coste total real

7 Especificación de requisitos

En esta sección se describe la especificación de requisitos de la aplicación. Es una de las partes más importantes del proyecto, ya que, si los requisitos no están bien especificados, posteriormente la implementación y el resultado de la aplicación no será el esperado por la empresa.

7.1 Funcionales

Para los requisitos funcionales se ha dividido la aplicación en áreas, dónde cada una pertenece a un grupo de acciones realizadas por el usuario.

- **Área de gestión de usuarios:** En esta área se encuentran todas las funcionalidades relacionadas con la gestión de usuario y sus diferentes personalizaciones y configuraciones que puede tener.
    - Iniciar sesión con email básico
    - Iniciar sesión con Microsoft
    - Cerrar sesión
    - Restablecer contraseña
    - Añadir usuario
    - Editar perfil
    - Editar permisos
    - Dar de alta una empresa
    - Editar empresa

- **Área administración:** En esta área se encuentran las funcionalidades relacionadas con la parte administrativa de la empresa.
    - Gestión de facturas
    - Gestión de servicios
    - Gestión de albaranes
    - Gestión de contratos
    - Gestión de contactos
    - Gestión de abonos
    - Modelo 347
    - Resumen de cliente

- **Área de soporte:** En esta área se encuentran las funcionalidades relacionadas con el soporte de atención al cliente de la empresa.
    - Gestión del ticketing
    - Ver actuaciones en el calendario
    - Resumen del ticketing

7.2 Roles

Dentro de la aplicación, existen diferentes tipos de usuarios con roles y permisos específicos. Aunque todas las personas que acceden se consideran usuarios, los roles definen qué acciones y áreas pueden gestionar según sus permisos. Los roles disponibles son:

- **Administrador global:** Este usuario tiene acceso completo a todas las funcionalidades y áreas de la aplicación. Puede gestionar usuarios, visualizar toda la información disponible y añadir nuevos clientes. Es el rol con el conjunto más amplio de permisos.

- **Cliente administrador:** Corresponde a un usuario asignado al cliente que puede gestionar su empresa dentro de la aplicación. Tiene acceso a todas las áreas, excepto aquellas relacionadas con la configuración de nuevos clientes. Además, puede añadir usuarios asociados a su empresa y gestionar sus permisos.

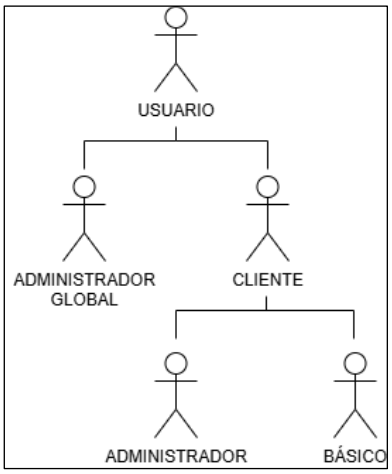


Figura 5. Diagrama de usuarios. Fuente: Propia

- **Cliente básico:** Es un usuario con permisos limitados, configurados por el cliente administrador. Solo puede acceder a las áreas y funcionalidades que le han sido asignadas explícitamente por el administrador de su empresa.

7.3 Diagrama de casos de uso

Área de gestión de usuarios

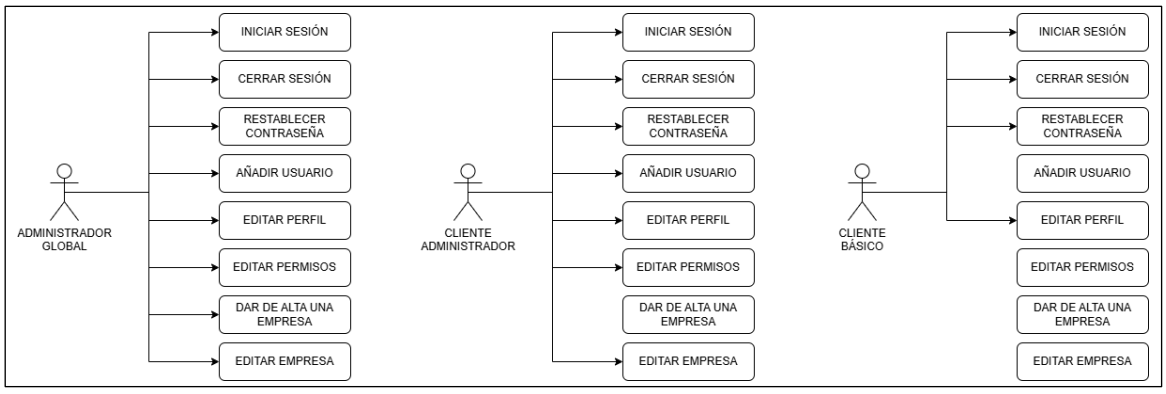


Figura 6. Diagrama de casos de uso del área de gestión de usuarios. Fuente: Propia

Área de administración

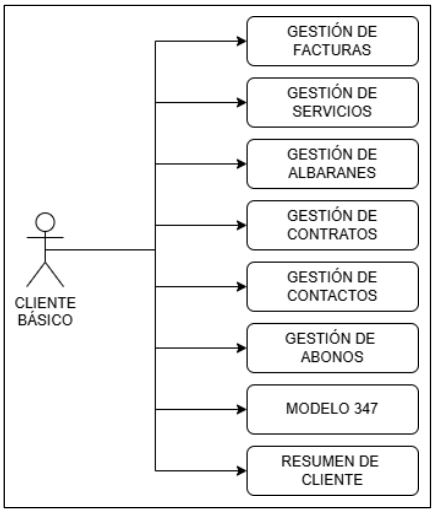


Figura 7. Diagrama de casos de uso del área de administración. Fuente: Propia

Área de soporte

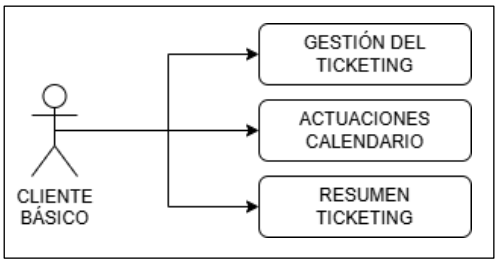


Figura 8. Diagrama de casos de uso del área de soporte. Fuente: Propia

7.4 Historias de usuario

En esta sección se pueden observar todas las historias de usuario recopiladas para el proyecto:

Historia de usuario 1: Iniciar sesión con email básico	
Descripción	Como usuario, quiero poder iniciar sesión utilizando mi dirección de correo electrónico y contraseña para acceder a la aplicación.
Criterios de aceptación	• El usuario tiene que estar dado de alta en la aplicación • El usuario tiene configurado el método de autenticación en básico • El usuario ha introducido la contraseña adecuada

Tabla 9. HU1: Iniciar sesión con email básico. Fuente: Propia

Historia de usuario 2: Iniciar sesión con Microsoft	
Descripción	Como usuario, quiero iniciar sesión usando mi cuenta de Microsoft para agilizar el acceso sin necesidad de ingresar manualmente mis credenciales.
Criterios de aceptación	• El usuario tiene que estar dado de alta en la aplicación • El usuario tiene configurado el método de autenticación en Microsoft • El usuario debe tener una cuenta válida en Microsoft

Tabla 10. HU2: Iniciar sesión con Microsoft. Fuente: Propia

Historia de usuario 3: Cerrar sesión	
Descripción	Como usuario, quiero cerrar mi sesión para salir de la aplicación web.
Criterios de aceptación	• El usuario tiene que estar autenticado

Tabla 11. HU3: Cerrar sesión. Fuente: Propia

Historia de usuario 4: Restablecer contraseña	
Descripción	Como usuario, quiero tener la opción de restablecer mi contraseña para asegurar que siempre tenga acceso controlado y seguro a mi cuenta.
Criterios de aceptación	• El usuario tiene que estar dado de alta en la aplicación • El usuario tiene el método de autenticación en básico • El usuario no ha intentado restablecer la contraseña más de 3 veces en un día

Tabla 12. HU4: Restablecer contraseña. Fuente: Propia

Historia de usuario 5: Añadir usuario	
Descripción	Como administrador global o cliente administrador, quiero añadir nuevos usuarios de la empresa a la plataforma para dar acceso a nuevos miembros del equipo.
Criterios de aceptación	• El usuario tiene que tener permisos de administrador global o cliente administrador • El usuario no puede añadir un usuario que ya existe

Tabla 13. HU5: Añadir usuario. Fuente: Propia

Historia de usuario 6: Editar Perfil	
Descripción	Como usuario, quiero editar mi perfil para actualizar mi información personal o imagen.
Criterios de aceptación	• El usuario con permisos básicos solo puede editar su propio perfil • El usuario con permisos de cliente administrador puede editar su propio perfil y el perfil de todos los usuarios de la empresa en la que está configurado • El usuario con permisos de administrador global puede editar el perfil de cualquier usuario de la aplicación

Tabla 14. HU6: Editar perfil. Fuente: Propia

Historia de usuario 7: Editar Permisos	
Descripción	Como administrador global o cliente administrador, necesito editar los permisos de los usuarios para asegurar el acceso adecuado a las funcionalidades de la plataforma.
Criterios de aceptación	• El usuario con permisos de cliente administrador puede editar los permisos de los usuarios de la empresa que está configurado • El usuario con permisos de administrador global puede editar los permisos de todos los usuarios de la aplicación

Tabla 15. HU7: Editar permisos. Fuente: Propia

Historia de usuario 8: Dar de alta una empresa	
Descripción	Como administrador global, quiero registrar nuevas empresas en la plataforma para que puedan acceder a la plataforma.
Criterios de aceptación	• El usuario tiene que tener los permisos de administrador global • La empresa no puede estar ya dada de alta en la aplicación

Tabla 16. HU8: Dar de alta una empresa. Fuente: Propia

Historia de usuario 9: Editar empresa	
Descripción	Como administrador global o cliente administrador, necesito editar los detalles de una empresa registrada para mantener la información actualizada.
Criterios de aceptación	• El usuario con permisos de administrador global puede editar los detalles de cualquier empresa de la aplicación • El usuario con permisos de cliente administrador solo puede editar los detalles de su empresa

Tabla 17. HU9: Editar empresa. Fuente: Propia

Historia de usuario 10: Gestión de facturas	
Descripción	Como usuario, quiero poder ver, filtrar, descargar en PDF y consultar el saldo pendiente de las facturas de venta para gestionar adecuadamente las cuentas por cobrar.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 18. HU10: Gestión de facturas. Fuente: Propia

Historia de usuario 11: Gestión de abonos	
Descripción	Como usuario, quiero poder ver, filtrar y descargar en PDF los abonos de venta para gestionar adecuadamente las cuentas por abonar.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 19. HU11: Gestión de abonos. Fuente: Propia

Historia de usuario 12: Gestión de contratos	
Descripción	Como usuario, quiero poder ver, filtrar y descargar en PDF los contratos de servicio para mantener un control adecuado sobre los acuerdos en curso y finalizados.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 20. HU12: Gestión de contratos. Fuente: Propia

Historia de usuario 13: Gestión de contactos	
Descripción	Como usuario, quiero poder ver, filtrar, crear, editar y eliminar contactos para mantener actualizados los contactos de la empresa.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 21. HU13: Gestión de contactos. Fuente: Propia

Historia de usuario 14: Gestión de albaranes	
Descripción	Como usuario, quiero poder ver, filtrar y descargar en PDF los albaranes para mantener un control adecuado sobre la mercancía recibida.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 22. HU14: Gestión de albaranes. Fuente: Propia

Historia de usuario 15: Gestión de hojas de servicios	
Descripción	Como usuario, quiero poder ver, filtrar y descargar en PDF las hojas de servicios realizados para llevar un registro detallado de las actividades que se han realizado.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 23. HU15: Gestión de hojas de servicios. Fuente: Propia

Historia de usuario 16: Modelo 347	
Descripción	Como usuario, quiero poder descargar el modelo 347 en formato CSV para cumplir con las obligaciones fiscales de la empresa y comparar que el resultado de los ficheros concuerde
Criterios de aceptación	• El usuario debe tener permisos para ver el área de administración

Tabla 24. HU16: Modelo 347. Fuente: Propia

Historia de usuario 17: Gestión del ticketing	
Descripción	Como usuario, quiero gestionar poder ver, abrir o cerrar un ticket para poder gestionar las incidencias y consultas en tiempo real.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de soporte

Tabla 25. HU17: Gestión del ticketing. Fuente: Propia

Historia de usuario 18: Resumen del ticketing	
Descripción	Como usuario, quiero ver un resumen rápido de los tickets y el estado en el que se encuentran, para tener una visión general de su estado actual.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de soporte

Tabla 26. HU18: Resumen del ticketing. Fuente: Propia

Historia de usuario 19: Calendario	
Descripción	Como usuario, quiero ver de manera mensual o semanal el total de intervenciones realizadas para mantener un control y seguimiento efectivo del trabajo realizado por la empresa.
Criterios de aceptación	• El usuario debe tener permisos para ver el área de soporte

Tabla 27. HU19: Calendario. Fuente: Propia

7.5 No funcionales

Para especificar los requisitos no funcionales, se ha decidido utilizar el modelo de especificación de requisitos *Volere* **[20]**, un estándar reconocido en la industria de la informática para la documentación sistemática de requisitos de software. Este modelo nos permite definir de manera clara y estructurada los requisitos de la aplicación, asegurando que todos los aspectos no funcionales sean cuidadosamente considerados y documentados.

Requisito	**10a. Requisitos de apariencia**
Descripción	Este requisito aborda todos los aspectos visuales del producto, incluyendo el cumplimiento de las normas de marca corporativa y el uso de colores específicos.
Justificación del requisito	Es indispensable asegurar que la apariencia del proyecto se alinee con las expectativas de la empresa y refuerce la identidad corporativa.
Criterio de aceptación	El diseño web será realizado por el proveedor designado de la empresa, que ya está familiarizado con los requisitos específicos de marca y estética de la empresa.

Tabla 28. Requisito no funcional: Requisitos de apariencia. Fuente: Propia

Requisito	**11a. Requisitos de facilidad de uso**
Descripción	Este requisito establece que la aplicación debe ser intuitiva y eficiente en su uso. Debe incluir aspectos como la eficiencia de uso y la facilidad de recordar procedimientos.
Justificación del requisito	La facilidad de uso es fundamental para asegurar que la aplicación no solo cumpla con las necesidades funcionales de los usuarios sino que también ofrezca una experiencia positiva que fomente su uso continuo.
Criterio de aceptación	Es diseño de la interfaz de la aplicación ha sido desarrollado por un proveedor, familiarizado con la empresa, cuyos diseños anteriores han demostrado una excelente facilidad de uso. Esta experiencia previa ha sido positivamente valorada por los clientes, quienes han destacado la buena usabilidad de otras aplicaciones creadas por el mismo proveedor.

Tabla 29. Requisito no funcional: Requisitos de facilidad de uso. Fuente: Propia

Requisito	12c. Requisitos de precisión o exactitud
Descripción	Este requisito especifica la precisión necesaria en los resultados obtenidos por la aplicación
Justificación del requisito	Al tener importes dentro de la aplicación, ajustar las expectativas de los clientes y usuarios respecto a la precisión de los importes es crucial para garantizar claridad y evitar confusiones o malas interpretaciones en cualquier movimiento financiero.
Criterio de aceptación	Todos los importes dentro de la aplicación estarán mostrados con dos decimales como máximo.

Tabla 30. Requisito no funcional: Requisitos de precisión o exactitud. Fuente: Propia

Requisito	12d. Requisitos de confiabilidad y disponibilidad
Descripción	Este requisito define la confiabilidad y disponibilidad necesarias de la aplicación, expresadas en términos de tiempo permisible entre fallos y la disponibilidad esperada a lo largo del tiempo.
Justificación del requisito	Los usuarios tienen que poder acceder a la aplicación en cualquier momento del día y en cualquier día del año.
Criterio de aceptación	El sistema tiene un tasa de tiempo en funcionamiento superior al 95%.

Tabla 31. Requisito no funcional. Requisitos de confiabilidad y disponibilidad. Fuente: Propia

Requisito	12g. Requisitos de expansión o crecimiento
Descripción	Este requisito detalla las expectativas de crecimiento en la capacidad de la aplicación para adaptarse a nuevas funcionalidades.
Justificación del requisito	Al ser la primera fase de la aplicación, la aplicación debe estar diseñada e implementada de manera que sea escalable a nuevos requisitos y funcionalidades.
Criterio de aceptación	El sistema está diseñado por áreas y páginas, lo que permite que cualquier nueva funcionalidad pueda ser añadida sin problema.

Tabla 32. Requisito no funcional: Requisitos de expansión o crecimiento. Fuente: Propia

Requisito	15a. Requisitos de acceso
Descripción	Este requisito define quién tiene acceso autorizado a la aplicación. Esto incluye acceso a funcionalidades específicas, asegurando que solo las personas adecuadas tengan acceso según su rol.
Justificación del requisito	Al permitir acceso a cualquier usuario de una empresa, esta tiene que poder filtrar la información y las funcionalidades que quiere que vea e utilice cada usuario.
Criterio de aceptación	El sistema se ha diseñado para que existan roles y permisos específicos para cada usuario.

Tabla 33. Requisito no funcional: Requisitos de acceso. Fuente: Propia

Requisito	15b. Requisitos de integridad
Descripción	Este requisito detalla las medidas necesarias para mantener la integridad de las bases de datos y los datos obtenidos por los servicios que utiliza la empresa. Incluye prevención de la visualización de datos incorrectos o duplicados.
Justificación del requisito	Para el proyecto, este requisito es crítico ya que la información que se muestra es sensible (facturas, abonos, etc.) y no puede contener ni un error, ya que pueden generar transacciones de dinero erróneos que pueden provocar problemas con el cliente.
Criterio de aceptación	Antes de dar acceso a clientes a la aplicación, se realizará una auditoría interna para comprobar la integridad de los datos.

Tabla 34. Requisito no funcional: Requisitos de integridad. Fuente: Propia

Requisito	15c. Requisitos de privacidad
Descripción	Este requisito especifica las medidas que la aplicación debe implementar para proteger la privacidad de los datos personales que almacena. Esto incluye asegurar la conformidad con las leyes de privacidad vigentes, informar a los usuarios sobre las prácticas de manejo de datos antes de recolectar cualquier información, y permitir a los usuarios gestionar su consentimiento y acceso a sus datos personales.
Justificación del requisito	Asegurar la privacidad de los datos es obligatorio para cumplir con las regulaciones legales. En un mundo donde la privacidad es cada vez más valorada, es vital que la aplicación mantenga altos estándares de protección de datos para evitar repercusiones legales y daños a la reputación de la empresa.
Criterio de aceptación	• Antes de dar acceso a un cliente a la aplicación, deberán firmar un contrato de privacidad de datos por parte de la empresa. • Dentro de la aplicación, al subir una imagen personal como foto de perfil, se debe aceptar el RGPD. • Dentro de la aplicación, se puede acceder a una pantalla con las políticas de privacidad de la empresa y de la aplicación.

Tabla 35. Requisito no funcional: Requisitos de privacidad. Fuente: Propia

7.6 Modelo conceptual

La aplicación está específicamente diseñada para integrar y optimizar los esquemas de las aplicaciones Microsoft Dynamics 365 Business Central y Dynamics 365 Customer Service. Por ello, el proyecto distingue tres modelos conceptuales diferentes, los cuales se pueden conectar en un modelo global.

Tal y como se puede observar en la Figura 9, el color amarillo representa el modelo de Dynamics 365 Customer Service, el color azul el modelo de Business Central y el que no tiene color representa el modelo propio.

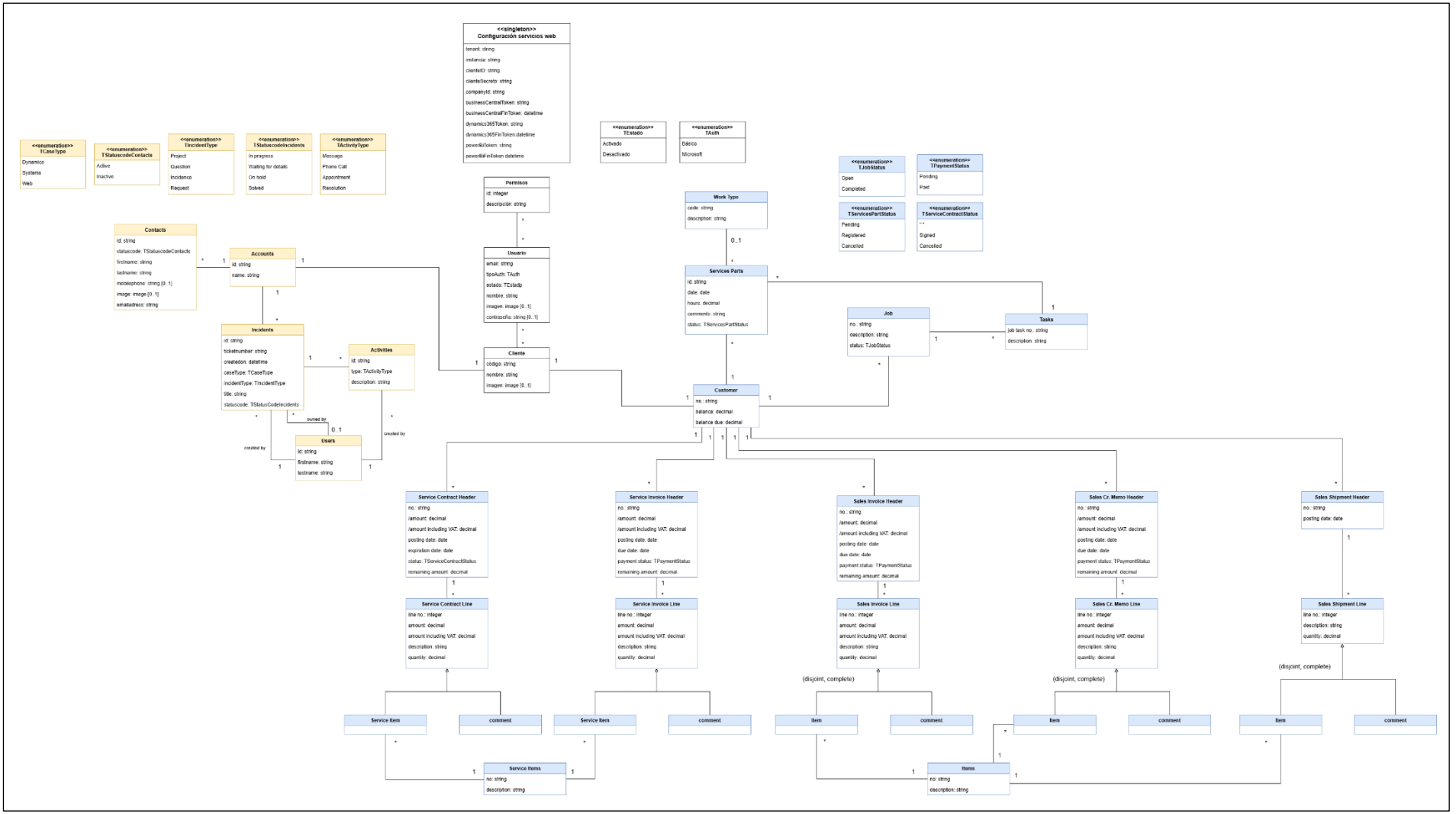


Figura 9. Modelo conceptual global. Fuente: Propia

7.6.1 Modelo Propio

El modelo propio es un modelo sencillo, que tiene el objetivo de controlar la gestión de usuarios, empresas y permisos. En la Figura 10 se puede observar el diagrama de clases del modelo:

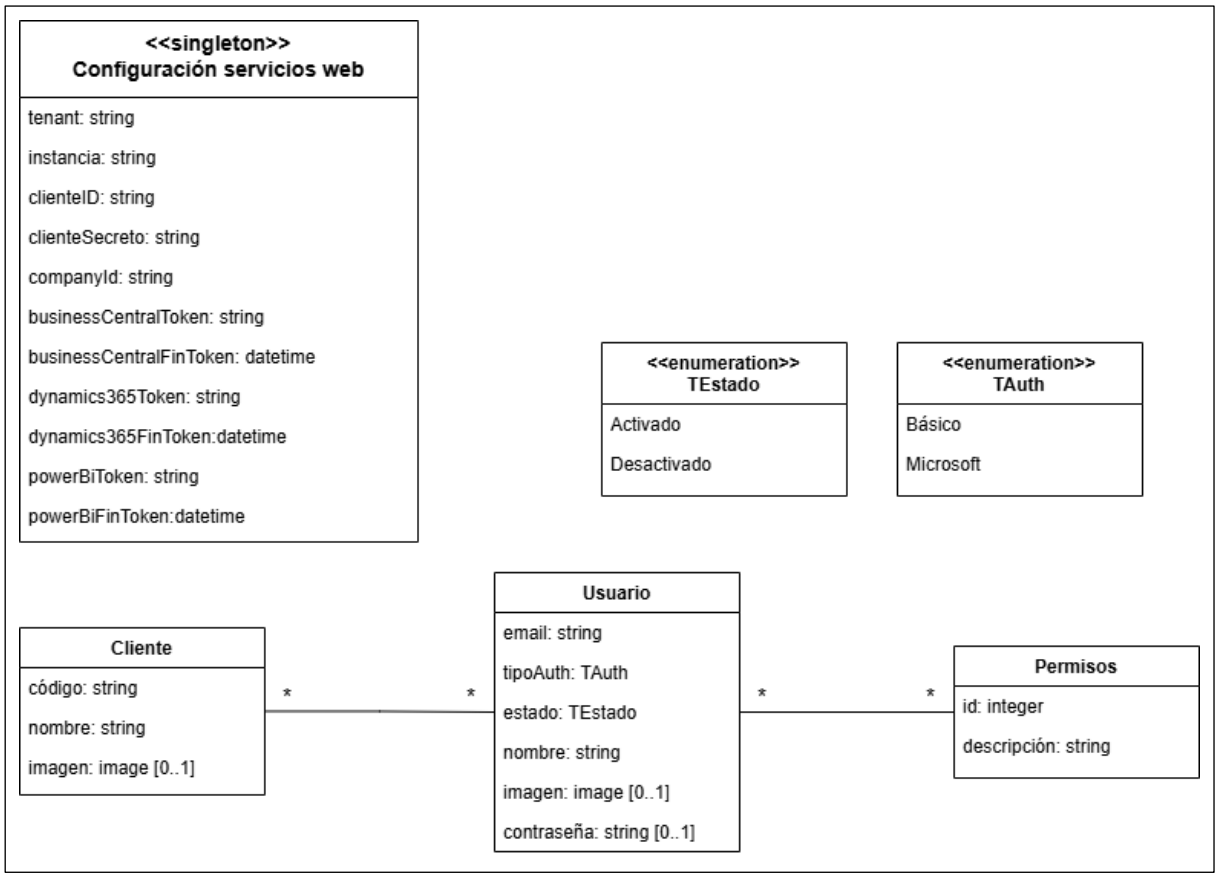


Figura 10. Modelo conceptual propio. Fuente: Propia

Claves externas:

(Cliente, código)

(Usuario, email)

(Permisos, id)

Restricciones textuales:

1. Si el método de autenticación del usuario es “Básico”, la contraseña no podrá estar vacía.

Descripción de las clases:

Cliente	
Descripción	Esta clase almacena la información principal para poder conectar a los servicios externos.
Atributos	
código	El código único que identifica al cliente. Debe coincidir con el código del Business Central.
nombre	Nombre de la empresa cliente
imagen	Imagen de la empresa cliente

Tabla 36. Descripción de la clase cliente. Fuente: Propia

Usuario	
Descripción	Esta clase almacena la información del usuario dentro de la aplicación.
Atributos	
email	El email del usuario
tipoAuth	Tipo de autenticación del usuario.
estado	Describe el estado del usuario en la aplicación
nombre	Nombre y apellidos del usuario
imagen	Imagen del usuario
contraseña	La clave para poder acceder a la aplicación

Tabla 37. Descripción de la clase usuario. Fuente: Propia

Permisos	
Descripción	Esta clase almacena los diferentes permisos dentro de la aplicación.
Atributos	
id	El identificador único del permiso
descripción	La descripción del permiso

Tabla 38. Descripción de la clase permisos. Fuente: Propia

7.6.2 Modelo Business Central

Principalmente, este modelo se utiliza para la gestión de administración. Recalcar que el modelo original que utiliza Business Central es mucho más extenso, en el caso del proyecto solo se escogido las clases y campos necesarios. Además, dentro del modelo se ha respetado el idioma original de Business Central que es el inglés. En la Figura 11 se puede observar el diagrama de clases del modelo:

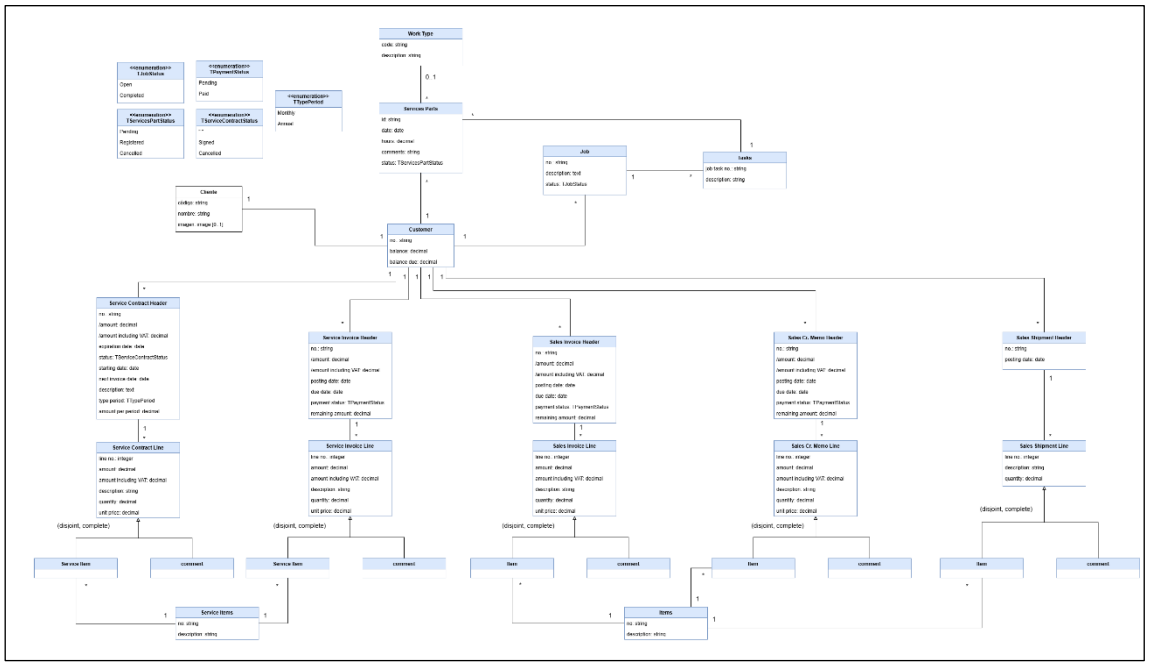


Figura 11. Modelo conceptual de Business Central. Fuente: Propia

Claves externas: (Customer, no.), (Services Parts, id), (Work Type, code), (Job, no.), (Tasks, job task no.), (Items, no.), (Services Items, no.), (Service Contract Header, no.), (Service Contract Line, line no.), (Service Invoice Header, no.), (Service Invoice Line, line no.), (Sales Invoice Header, no.), (Sales Invoice Line, line no.), (Sales Cr. Memo Header, no.), (Sales Cr. Memo Line, line no.), (Sales Shipment Header, no.), (Sales Shipment Line, line no.)

Restricciones textuales:

1. La tarea de proyecto que se incluye en una hoja de servicio tiene que pertenecer al mismo cliente.

Sales Invoice Header (Cabecera de Facturas de Venta)	
Descripción	Esta clase almacena la información principal de las facturas de venta en Microsoft Dynamics 365 Business Central. Cada registro representa la cabecera de una factura enviada a un cliente.
Atributos	
no.	El identificador principal que se utiliza para referenciar la factura en el sistema.
/amount	El total del importe de la factura sin incluir el IVA. • **Campo calculado:** Suma el campo “amount” de las clases “Sales Invoice Line” que tiene relacionadas.
/amount including VAT	El total del importe de la factura incluyendo el IVA. • **Campo calculado:** Suma el campo “amount including VAT” de las clases “Sales Invoice Line” que tiene relacionadas
posting Date	La fecha en que la factura fue registrada y contabilizada en el sistema.
payment Status	El estado actual del pago de la factura. ***Opciones:*** - Pendiente - Pagada
remaining Amount	El importe que aún queda por pagar en la factura.

Tabla 39. Descripción de la clase "Sales Invoice Header". Fuente: Propia

Sales Invoice Line (Líneas de Facturas de Venta)	
Descripción	Esta clase representa la información detallada sobre las líneas específicas facturadas dentro de cada factura de venta. Cada línea está vinculada a una factura.
Atributos	
line No.	Un número secuencial que identifica cada línea dentro de una factura específica.
amount	El total del importe de la línea sin incluir el IVA .
amount including VAT	El total del importe de la línea incluyendo el IVA.
description	La descripción del producto o un comentario añadido manualmente por el usuario.
quantity	La cantidad de la línea.
unit Price	El precio unitario de la línea.

Tabla 40. Descripción de la clase "Sales Invoice Line". Fuente: Propia

Services Parts (Historial partes de servicio)	
Descripción	Esta clase almacena la información de las entradas en el diario de proyecto en Microsoft Dynamics 365 Business Central. Cada registro representa un parte de servicio enviado a un cliente.
Atributos	
id	Identificador único del parte de servicio. Incluye validación de series de numeración al modificar.
date	Fecha en la que se registró el parte de servicio.
hours	Cantidad de horas trabajadas
comments	Comentarios del trabajo realizado
status	Estado de la hoja de servicio. ***Opciones:*** - Pendiente - Registrado - Cancelado

Tabla 41. Descripción de la clase "Services Parts". Fuente: Propia

Job (Proyecto)	
Descripción	Esta clase almacena información detallada sobre los proyectos en Microsoft Dynamics 365 Business Central. Cada registro representa un proyecto individual.
Atributos	
no.	Identificador único del proyecto.
description	Descripción del proyecto.
status	Estado del proyecto. ***Opciones:*** -Abierto -Completado

Tabla 42. Descripción de la clase "Job". Fuente: Propia

Tasks (Tareas)	
Descripción	Esta clase almacena información detallada sobre las tareas asociadas a proyectos en Microsoft Dynamics 365 Business Central. Cada registro representa una tarea individual relacionada a un proyecto.
Atributos	
job task no.	Identificador único de la tarea
description	Descripción de la tarea.

Tabla 43. Descripción de la clase "Tasks". Fuente: Propia

Items (Articulos)	
Descripción	Esta clase almacena información detallada sobre los productos de venta en Microsoft Dynamics 365 Business Central.
Atributos	
no.	Identificador único del producto
description	Descripción del producto

Tabla 44. Descripción de la clase "Items". Fuente: Propia

Service Items (Articulos)	
Descripción	Esta clase almacena información detallada sobre los productos de servicio en Microsoft Dynamics 365 Business Central.
Atributos	
no.	Identificador único del producto de servicio
description	Descripción del producto de servicio

Tabla 45. Descripción de la clase "Service Items". Fuente: Propia

Service Invoice Header (Cabecera de Facturas de Servicios)	
Descripción	Esta clase almacena la información principal de las facturas de servicio registradas en Microsoft Dynamics 365 Business Central. Cada registro representa la cabecera de una factura de servicio enviada a un cliente.
Atributos	
no.	El número único de la factura.
/amount	El total del importe de la factura sin incluir el IVA. • **Campo calculado:** Suma el campo “amount” de las clases “Service Invoice Line” que tiene relacionadas.
/amount including VAT	El total del importe de la factura incluyendo el IVA. • **Campo calculado:** Suma el campo “amount including VAT” de las clases “Service Invoice Line” que tiene relacionadas
posting Date	La fecha en que la factura fue registrada y contabilizada en el sistema.
payment Status	El estado actual del pago de la factura. ***Opciones:*** - Pendiente - Pagada
remaining Amount	El importe que aún queda por pagar en la factura.

Tabla 46. Descripción de la clase "Service Invoice Header". Fuente: Propia

Service Invoice Line (Líneas de Facturas de Servicio)	
Descripción	Esta clase representa la información detallada sobre las líneas específicas facturadas dentro de cada factura de servicio.
Atributos	
line No.	Un número secuencial que identifica cada línea.
amount	El total del importe de la línea sin incluir el IVA .
amount Including VAT	El total del importe de la línea incluyendo el IVA.
description	La descripción del producto o un comentario.
quantity	La cantidad de la línea.
unit Price	El precio unitario de la línea.

Tabla 47. Descripción de la clase "Service Invoice Line". Fuente: Propia

Service Contract Header (Cabecera de Contratos de Servicios)	
Descripción	Esta clase almacena la información principal de los contratos de servicio registrados Business Central.
Atributos	
no.	El número único del contrato de servicio.
/amount	El total del importe de la factura sin incluir el IVA. • **Campo calculado:** Suma el campo “amount” de las clases “Service Contract Line” que tiene relacionadas.
/amount including VAT	El total del importe de la factura incluyendo el IVA. • **Campo calculado:** Suma el campo “amount including VAT” de las clases “Service Contract Line” que tiene relacionadas
expiration date	Fecha en la que acaba el contrato
status	El estado actual del contrato. ***Opciones:*** - Abierto - Firmado - Cancelado
starting date	Fecha de inicio del contrato
next invoice date	Fecha de la próxima factura asociada al contrato
description	Descripción general del contrato
type period	Tipo de periodo de servicio ***Opciones:*** - Mensual - Anual
amount per period	Importe por periodo

Tabla 48. Descripción de la clase "Service Contract Header". Fuente: Propia

Service Contract Line (Líneas de Contratos de Servicio)	
Descripción	Esta clase representa la información detallada sobre las líneas específicas dentro de cada contrato de servicio.
Atributos	
line No.	Un número secuencial que identifica cada línea dentro de un contrato de servicio específico.
amount	El total del importe de la línea sin incluir el IVA .
amount Including VAT	El total del importe de la línea incluyendo el IVA.
description	La descripción del producto o un comentario añadido manualmente por el usuario.
quantity	La cantidad de la línea.
unit Price	El precio unitario de la línea.

Tabla 49. Descripción de la clase "Service Contract Line". Fuente: Propia

Sales Cr. Memo Header (Cabecera de Abonos de Venta)	
Descripción	Esta clase almacena la información principal de los abonos de venta registrados en Business Central.
Atributos	
no.	El número único del abono de venta.
/amount	El total del importe de la factura sin incluir el IVA. • **Campo calculado:** Suma el campo “amount” de las clases “Sales Cr. Memo Line” que tiene relacionadas.
/amount including VAT	El total del importe de la factura incluyendo el IVA. • **Campo calculado:** Suma el campo “amount including VAT” de las clases “Sales Cr. Memo Line” que tiene relacionadas
posting Date	La fecha en que el abono fue registrado y contabilizado en el sistema.
payment Status	El estado actual del pago del abono. ***Opciones:*** - Pendiente - Pagada
remaining Amount	El importe que aún queda por pagar del abono.

Tabla 50. Descripción de la clase "Sales Cr. Memo Header". Fuente: Propia

Sales Cr. Memo Line (Líneas de Abonos de venta)	
Descripción	Esta clase representa la información detallada sobre las líneas específicas dentro de cada abono de venta.
Atributos	
line No.	Un número secuencial que identifica cada línea dentro de un abono específico.
amount	El total del importe de la línea sin incluir el IVA .
amount including VAT	El total del importe de la línea incluyendo el IVA.
description	La descripción del producto o un comentario añadido manualmente por el usuario.
quantity	La cantidad de la línea.
unit Price	El precio unitario de la línea.

Tabla 51. Descripción de la clase "Sales Cr. Memo Line". Fuente: Propia

Sales Shipment Header (Cabecera de Albaranes de Venta)	
Descripción	Esta clase almacena la información principal de los albaranes de venta registrados en Business Central.
Atributos	
no.	El número único del albarán de venta.
posting date	La fecha en que el albarán fue registrado y contabilizado en el sistema.

Tabla 52. Descripción de la clase "Sales Shipment Header". Fuente: Propia

Sales Shipment Line (Líneas de Albaranes de venta)	
Descripción	Esta clase representa la información detallada sobre las líneas específicas dentro de cada albarán de venta.
Atributos	
line No.	Un número secuencial que identifica cada línea dentro de un abono específico.
description	La descripción del producto o un comentario añadido manualmente por el usuario.
quantity	La cantidad de la línea.

Tabla 53. Descripción de la clase "Sales Shipment Line". Fuente: Propia

7.6.3 Modelo Customer Service

Este modelo se encarga de la gestión de contactos y del área de soporte, donde son necesarios los datos de Customer Service. Las clases mostradas en la Figura 12 no representan la totalidad de campos que contiene Customer Service, sino solo aquellos que se utilizan en el proyecto.

Respetando el idioma oficial de Customer Service, se han dejado las clases y los campos en inglés.

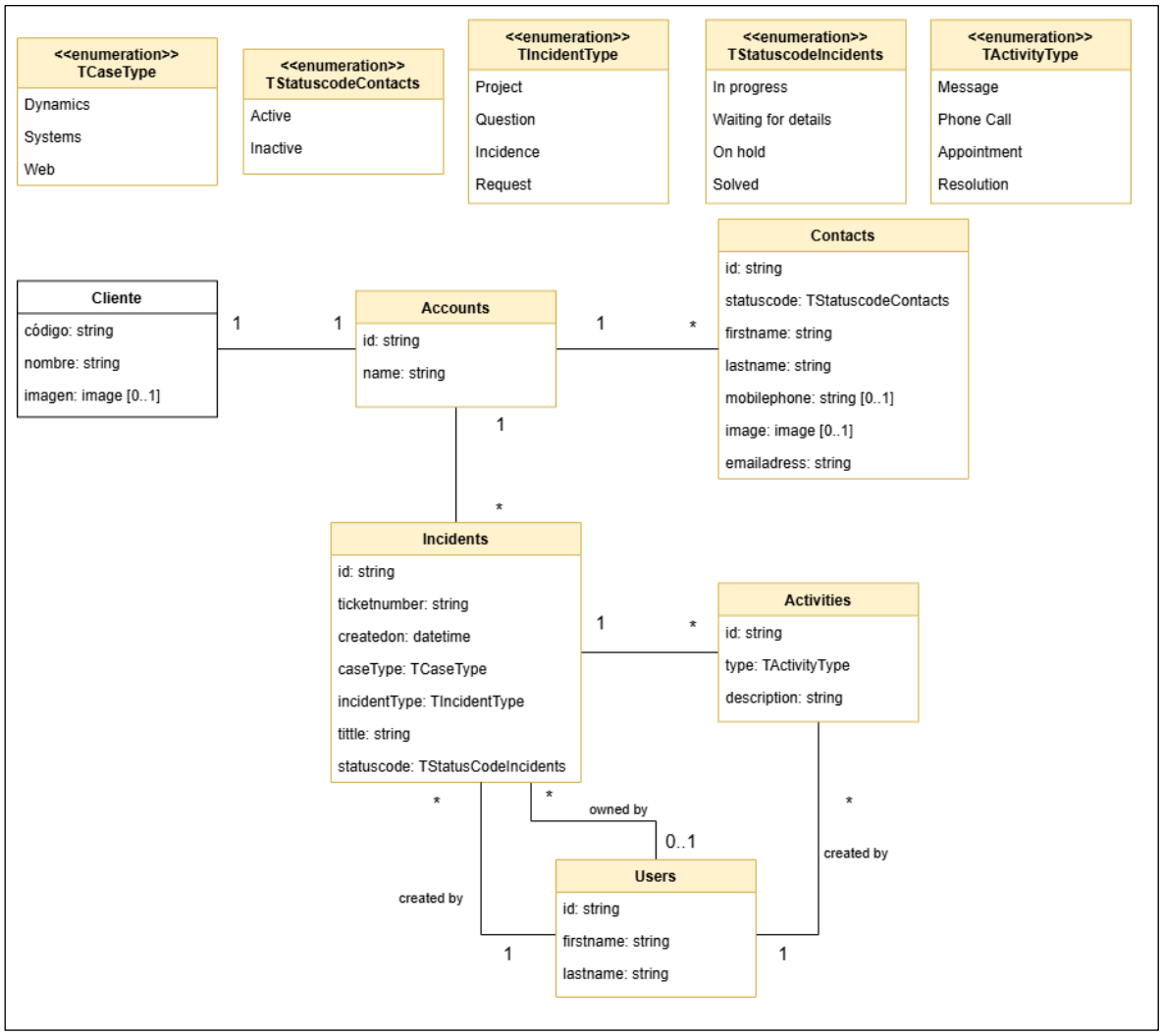


Figura 12. Modelo conceptual de Customer Service. Fuente: Propia

Claves externas:

(Accounts, id), (Contacts, id), (Incidents, id), (Users, id), (Activities, id)

Descripción de las clases:

Accounts (Cuentas)	
Descripción	Esta clase representa la información relacionada con el cliente en Microsoft Dynamics 365 Customer Service
Atributos	
id	Es el identificador principal
name	El nombre del cliente

Tabla 54. Descripción de la clase "Accounts". Fuente: Propia

Contacts (Contactos)	
Descripción	Esta clase representa la información detallada sobre los contactos de una empresa en Microsoft Dynamics 365 Customer Service.
Atributos	
id	El identificador único del contacto.
status code	El estado del contacto ***Opciones:*** - Activo - Inactivo
firstname	El nombre del contacto
lastname	Los apellidos del contacto
mobilephone	El número de teléfono del contacto
image	La imagen del contacto
emailadress	La cuenta de correo electrónico del contacto

Tabla 55. Descripción de la clase "Contacts". Fuente: Propia

Incidents (Casos)	
Descripción	Esta clase representa la información detallada sobre los casos de una empresa en Microsoft Dynamics 365 Customer Service.
Atributos	
id	El identificador único de un caso.
ticketnumber	El identificador secundario para poder filtrar de una manera más fácil. Es el número que se le da al cliente al abrir un caso.
createdon	La fecha y hora en que se creó el caso.

caseType	La categorización del caso ***Opciones:*** - Dynamics - Sistemas - PowerBI - Web - Administración - Comercial
incidentType	El tipo de caso ***Opciones:*** - Proyecto - Consulta - Incidencia - Solicitud
title	El título del caso, se utiliza como descripción.
statuscode	El estado del caso ***Opciones:*** - En progreso - Esperando detalles - En espera - Resuelto

Tabla 56. Descripción de la clase "Incidents". Fuente: Propia

Activities (Actividades)	
Descripción	Esta clase representa la información detallada sobre las actividades realizadas, estas estan relacionadas con los casos en Microsoft Dynamics 365 Customer Service.
Atributos	
id	El identificador único de un actividad.
type	El tipo de actividad ***Opciones:*** - Email - Llamada telefónica - Reunión - Resolución
description	Describe lo que se ha hecho en la actividad

Tabla 57. Descripción de la clase "Activities". Fuente: Propia

Users (Usuarios)	
Descripción	Esta clase representa la información detallada sobre los usuarios dados de alta en Microsoft Dynamics 365 Customer Service.
Atributos	
id	El identificador único de un usuario.
firstname	El nombre del usuario.
lastname	Los apellidos del usuario.

Tabla 58. Descripción de la clase "Users". Fuente: Propia

8 Arquitectura del sistema

8.1 Arquitectura física

La arquitectura física del sistema está separada en tres capas, la capa de presentación, la capa de negocio y la capa de persistencia. Dentro de la capa de negocio, se encuentran dos partes, la parte interna y la parte de servicios externos proporcionados por Microsoft.

Tal y como se observa en la Figura 13, el sistema dispone de un servidor en la nube proporcionado por clouding.io, donde se alojan tanto el frontend, el backend y la base de datos.

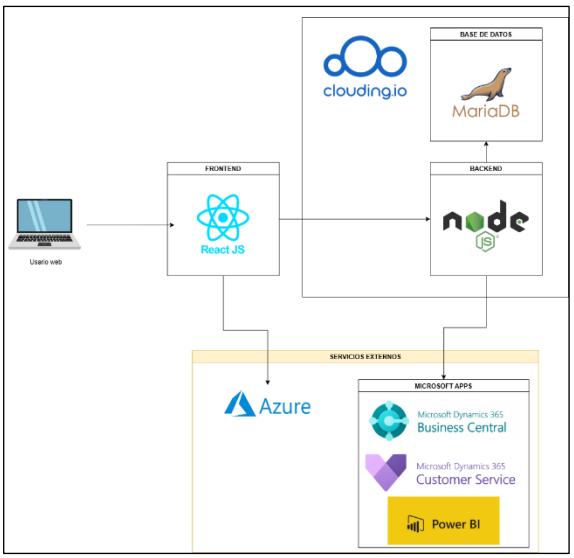


Figura 13. Arquitectura física. Fuente: Propia

El frontend, que actúa como capa de presentación, se construye con React JS, una biblioteca de JavaScript para el desarrollo de interfaces de usuario. Este se ejecuta en el navegador del usuario y se encarga de presentar la interfaz de la aplicación, gestionando la interactividad y la visualización de los datos en tiempo real. React JS permite una experiencia de usuario altamente dinámica. Además, facilita el desarrollo de componentes reutilizables, lo que optimiza el proceso de desarrollo y mejora la coherencia del diseño.

El backend, que actúa como capa de negocio, está implementado en Node.js, un entorno de ejecución para JavaScript del lado del servidor que es ideal para construir aplicaciones en tiempo real y orientadas a servicios. El objetivo del backend es hacer de intermediario entre el frontend, la base de datos MariaDB y los servicios externos. Este se encarga de procesar las solicitudes que recibe del frontend, ejecutando la lógica de negocio.

En la capa de persistencia, el sistema emplea MariaDB como su sistema de gestión de bases de datos. Aquí se almacenan y recuperan todos los datos esenciales de la gestión de usuarios y permisos para el funcionamiento de la aplicación, asegurando que la información esté disponible y organizada para su uso.

Además, la arquitectura integra servicios externos a través de API, destacándose Azure para la autenticación y conexión con aplicaciones de Microsoft como Dynamics 365 Business Central, Dynamics 365 Customer Service y Power Bi.

La totalidad de la infraestructura interna del sistema, que abarca el *frontend*, el *backend* y la base de datos, está alojada en la nube, específicamente en clouding.io. Es importante recalcar que la decisión de centralizar todas las operaciones en un único entorno de nube es una estrategia empresarial.

8.2 Arquitectura lógica

El flujo del sistema empieza cuando el usuario accede a la capa de presentación a través de un navegador web, utilizando el protocolo HTTPS **[21]** para llegar a la URL pública del sistema. Una vez que el usuario se ha conectado a la interfaz de usuario, la capa de presentación internamente establece comunicación con la capa de negocio. Esta conexión se realiza dentro del mismo servidor alojado en clouding.io, utilizando el protocolo HTTP **[22]**, que, aunque es menos seguro que HTTPS, es viable internamente ya que la comunicación no sale del servidor y está configurada para que no pueda acceder ninguna otra conexión.

La capa de negocio para interactuar con la base de datos se conecta a la instancia de MariaDB utilizando el protocolo TCP **[23]** apuntando al puerto en el que se encuentra la instancia, asegurando una comunicación eficiente y directa para operaciones de consulta y actualización de datos.

Además, la capa de negocio también establece conexiones con servicios externos, como las aplicaciones de *Microsoft Dynamics 365* y *Azure*. Estas conexiones se efectúan mediante el protocolo HTTP, permitiendo la integración y el consumo de servicios externos.

En resumen, este flujo detallado demuestra cómo se estructuran las conexiones entre las diferentes capas y componentes del sistema, desde la interacción inicial del usuario hasta el procesamiento interno y la integración con sistemas externos, todo dentro de un entorno controlado y centralizado en la nube de clouding.io.

8.3 Diseño de la interfaz

Para garantizar la coherencia con la estética e imagen corporativa, se contrataron los servicios del proveedor de diseño web de la empresa. Sin embargo, a medida que el proyecto avanzaba, los costes asociados al diseño gráfico comenzaron a aumentar significativamente. Esto llevó a la decisión de incorporar tecnologías de inteligencia artificial para la modificación y optimización de detalles en la interfaz. Así, aunque el diseño inicial fue creado por un diseñador gráfico, con el tiempo se integró la inteligencia artificial para refinar y adaptar la interfaz de manera más eficiente y con un coste significativamente menor.

8.3.1 Diseñador gráfico

Para la fase inicial del diseño web del proyecto, se utilizó Figma **[24]**, una plataforma de diseño de interfaz líder en la industria. Esta herramienta permitía al diseñador gráfico publicar todos los prototipos allí, facilitando así el acceso para revisar y exportar todos los recursos gráficos necesarios, como SVGs[3] y PNGs[4], directamente desde la plataforma.

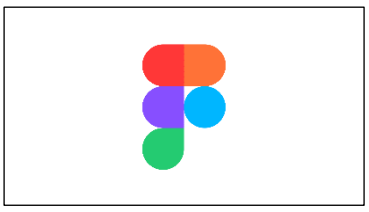


Figura 14. Logo de Figma. Fuente: Figma

[3] SVG: Es un tipo de archivo de gráficos vectoriales que se utiliza para representar imágenes y gráficos en formato vectorial.

[4] PNG: Es un tipo de archivo de gráficos utilizado para imágenes digitales, especialmente en la web, que soporta transparencias y ofrece una compresión sin pérdida de calidad.

A continuación, se pueden ver los mockups[5] iniciales proporcionados por el proveedor de diseño web:

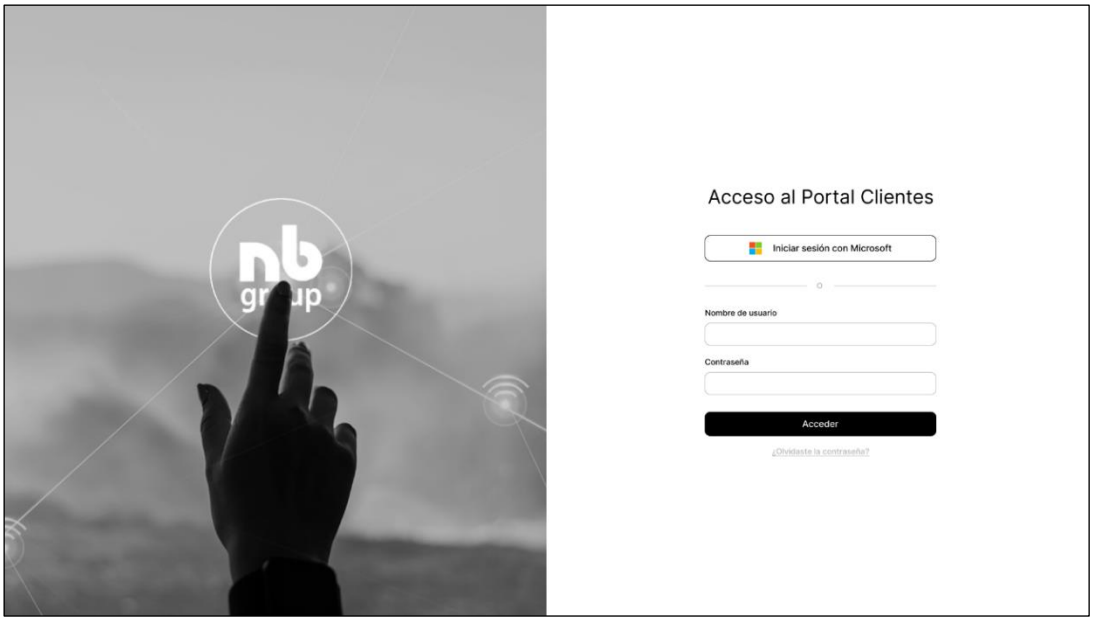


Figura 15. Mockup: Pantalla Login. Fuente: Propia

En la Figura 15, se observa la pantalla inicial antes de iniciar sesión, dónde esta se divide la pantalla en dos. La primera mitad es una representación gráfica del logo de la empresa y la segunda mitad es el login. Dentro del login se observan dos secciones:

- La primera sección contiene el botón “Iniciar sesión con Microsoft” que tal y como indica su propio nombre, permite iniciar sesión con una cuenta de Microsoft sin necesidad de rellenar ningún otro parámetro.

- La segunda sección es un formulario, en concreto nos permite rellenar el nombre de usuario y la contraseña. Por otro lado, tenemos otro botón que nos permite cambiar de formulario para restablecer la contraseña.

[5] Mockups: Es una representación visual que simula el aspecto final de un producto o diseño.

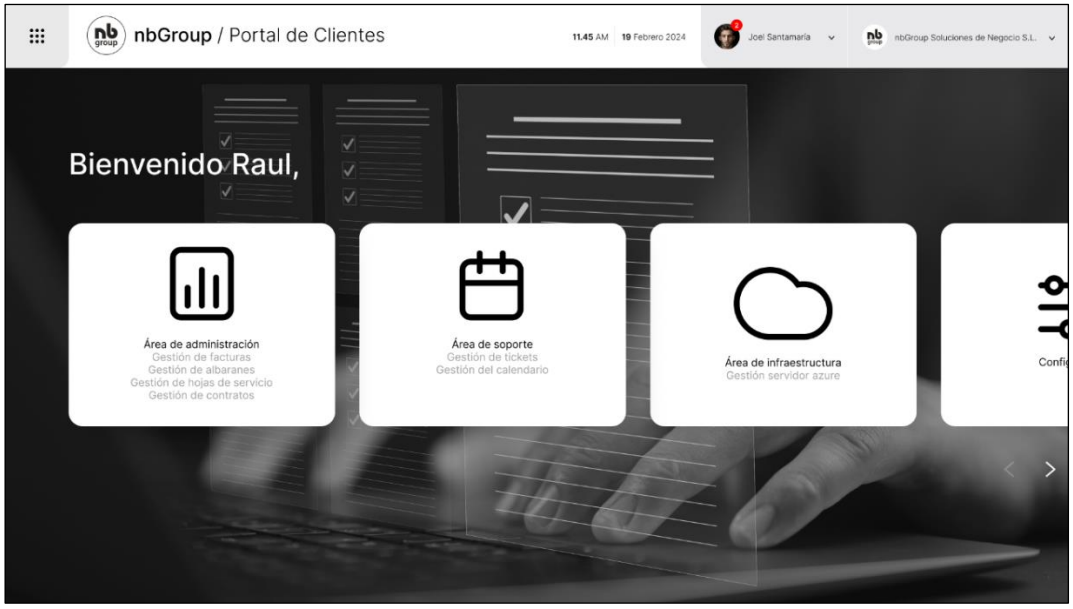


Figura 16. Mockup: Pantalla inicial. Fuente: Propia

La Figura 16 es la pantalla principal una vez el usuario está autenticado. En esta se encuentra la cabecera, que se mantendrá para todas las pantallas. Dentro de esta se puede observar:

- **Botón de buscador:** En primer lugar, se encuentra un botón que nos abre una pequeña sección para poder cambiar de área. En la Figura 17 se observa la pantalla con el selector y buscador.

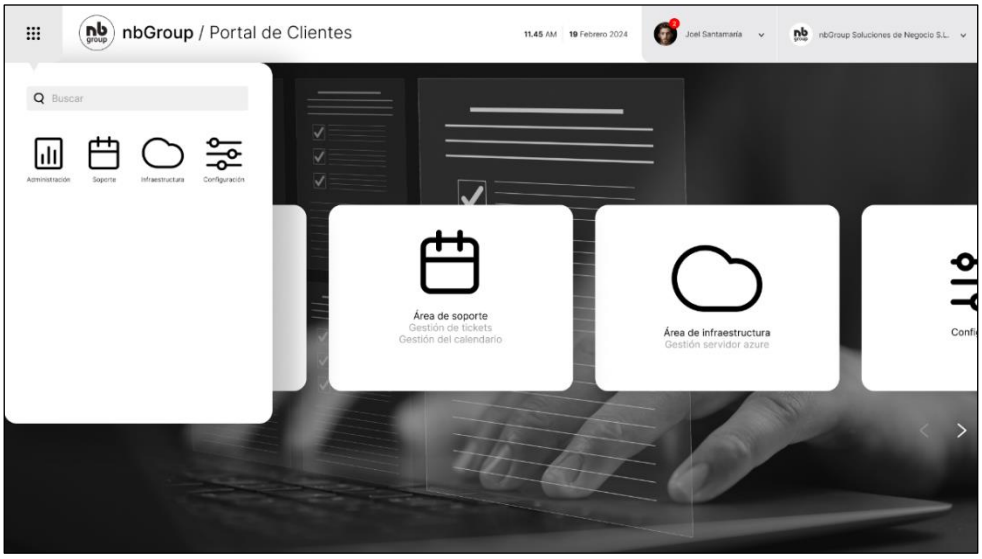


Figura 17. Mockup. Pantalla inicial con buscador. Fuente: Propia

- **Logo de la pantalla:** En la pantalla principal, el header muestra el logo de la empresa. Este logo cambia dinámicamente según el área a la que accede el usuario, mostrando el logo correspondiente a esa área específica para reforzar la identificación visual del entorno en el que se encuentra el usuario.

- **Título de la pantalla:** El título también varía en función de la pantalla y el área en la que se encuentra el usuario.

- **Reloj en tiempo real:** Se incluye un reloj que muestra la fecha y hora exacta.

- **Imagen y nombre del usuario:** Se muestra la imagen y el nombre del usuario autenticado. Al lado del nombre, se encuentra un menú desplegable, que permite al usuario editar su perfil o cerrar sesión fácilmente.

- **Imagen y nombre de la empresa:** Se muestra la imagen y el nombre de la empresa actualmente seleccionada por el usuario autenticado. Si el usuario tiene acceso a más de una empresa, se habilita un menú desplegable que permite seleccionar entre las empresas disponibles.

No obstante, el contenido principal de la pantalla es un mensaje de bienvenida al usuario juntamente con un slider[6] que nos permite seleccionar entre las áreas disponibles para acceder de una forma rápida y visual.

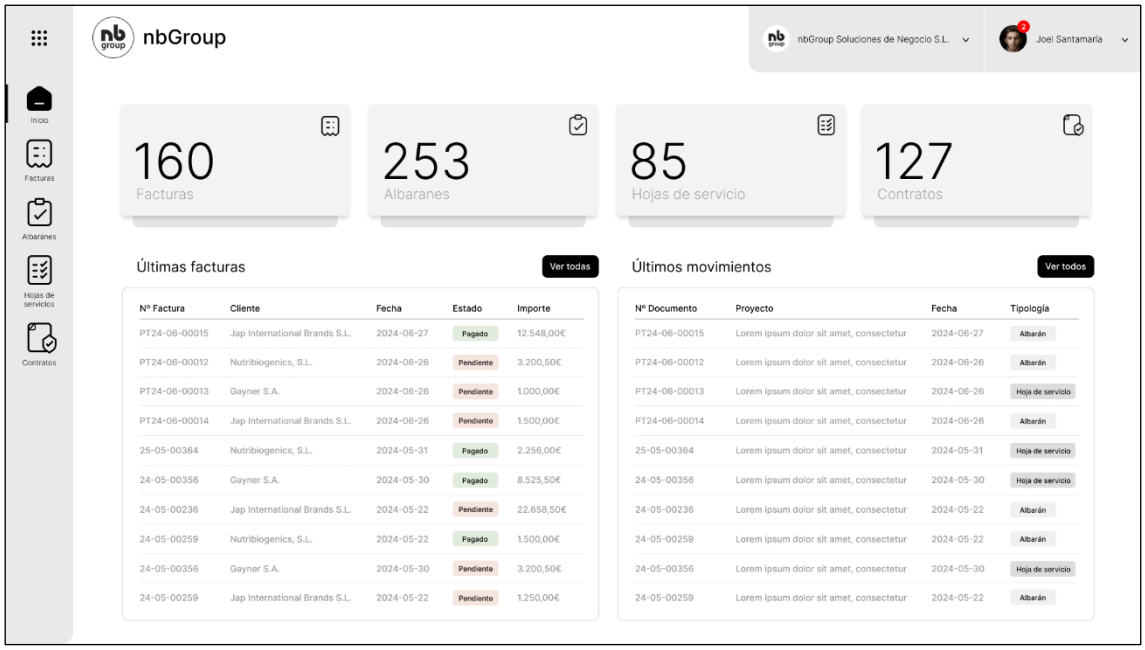


Figura 18. Mockup: Pantalla resumen cliente. Fuente: Propia

En la Figura 18, se observa la pantalla “Resumen de Cliente”. Esta pantalla contiene la información administrativa más relevante de una manera visual y clara. Además, se ha habilitado la sidebar izquierda dónde se encuentran todas las secciones disponibles para el usuario.

[6] Slider: Es un recurso de diseño web que consiste en mostrar distintas imágenes o componentes que se van deslizando en la pantalla.

El contenido de esta pantalla esta formado por 4 KPIs, dónde nos indican el número total de facturas, el número total de albaranes, el número total de hojas de servicio y el número total de contratos (incluyendo activos y cancelados). Además, cada KPI redirige al usuario al contenido clickado. Por último, se encuentran dos secciones:

- Últimas facturas: Se incluyen las últimas facturas registradas y el estado en el que están actualmente.

- Últimos movimientos: Se incluyen todos los últimos movimientos que la empresa ha tenido con el cliente, sin importar el tipo de registro que sea.

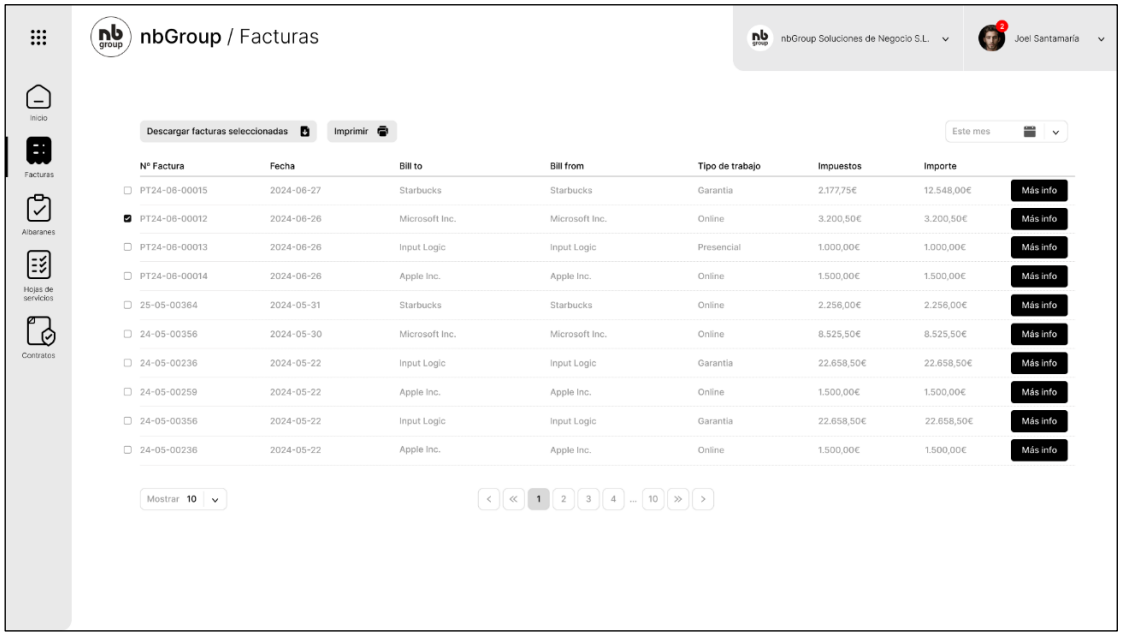


Figura 19. Mockup: Pantalla listados. Fuente: Propia

En la Figura 19, se observa la plantilla que se utiliza para todos los listados. En esta pantalla se puede filtrar por la fecha a través del calendario, ordenar cualquier columna y seleccionar líneas para descargar el pdf de manera masiva.

Al final de cada línea se encuentra el botón “Más info” que es el encargado de abrir el detalle del registro, en este caso una factura.

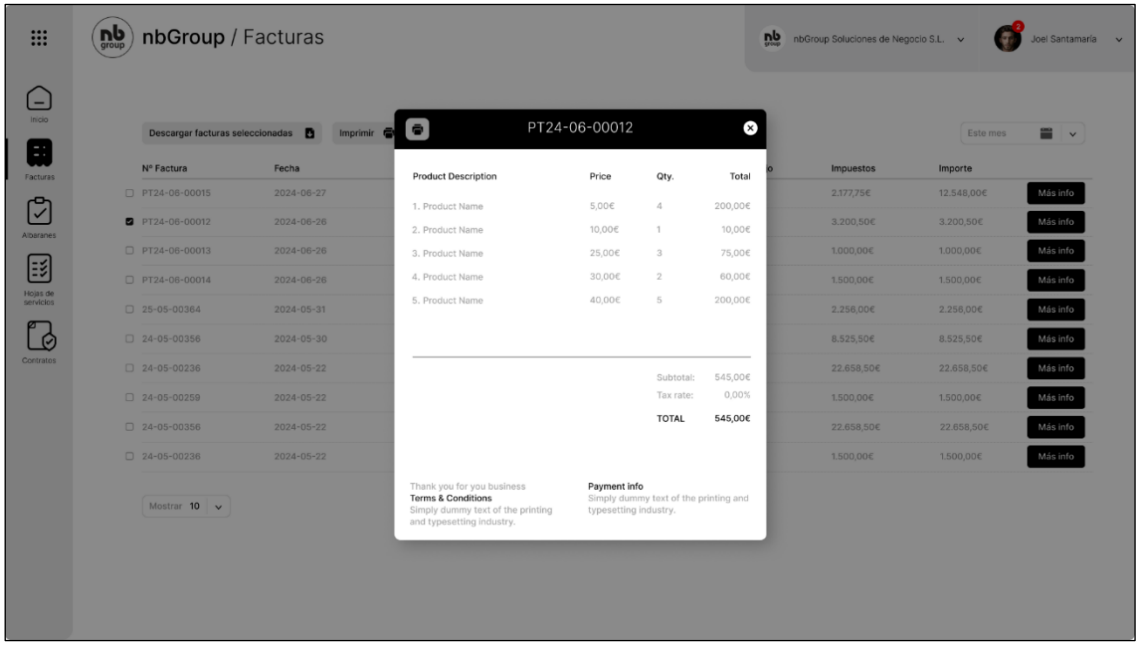


Figura 20. Mockup: Plantilla detalle. Fuente: Propia

La Figura 20, enseña el detalle de un registro a través de un popup. En este se puede ver el número del documento en la cabecera, juntamente con el botón de una impresora, que permite descargar en pdf ese registro en concreto. Cada popup cambia dependiendo del tipo de registro que sea, ya que los campos a mostrar y la información varia.

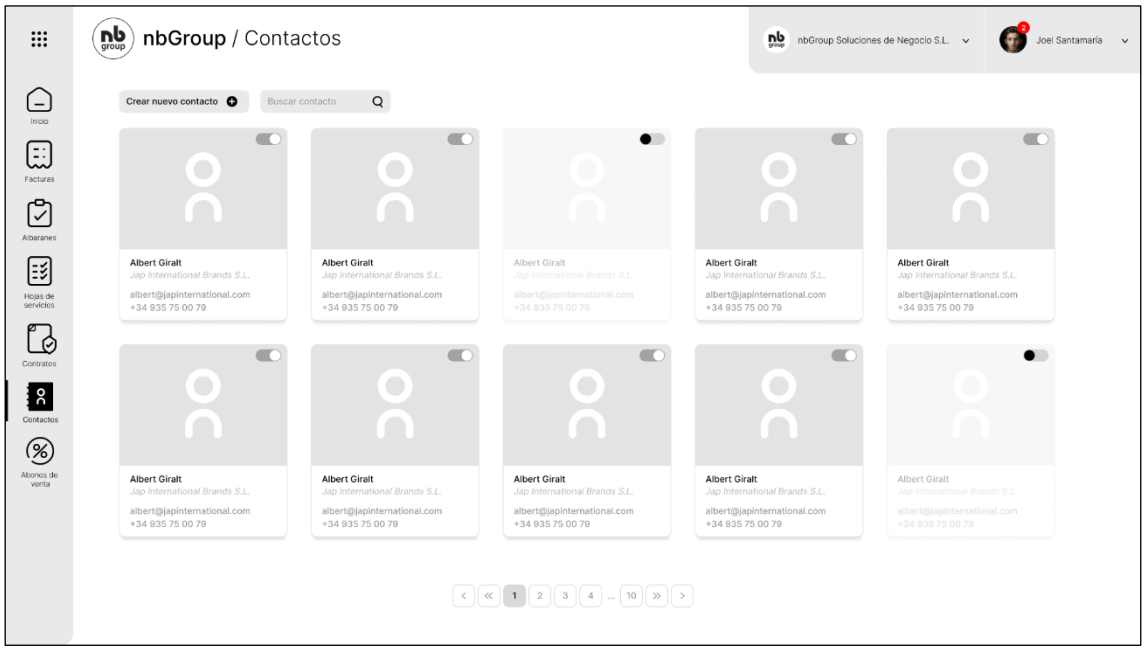


Figura 21. Mockup: Pantalla contactos. Fuente: Propia

La Figura 21 muestra la pantalla de contactos. En esta pantalla también se puede filtrar, pero no ordenar. Además, existe un botón que permite crear un contacto desde cero. Principalmente, en esta pantalla se puede observar cards, dónde cada card es un contacto que contiene:

- Nombre
- Empresa a la que pertenece
- Email
- Número de teléfono
- Imagen (En caso de que tenga)

Además, cada card también indica si el contacto está habilitado en el sistema. Un contacto marcado como invisible se considera deshabilitado, lo que significa que ya no se utiliza activamente. Esto ayuda a los usuarios a distinguir rápidamente entre contactos actuales y aquellos que han quedado obsoletos sin necesidad de eliminar la información histórica.

8.3.2 Inteligencia Artificial

Una vez que se empezaron a desarrollar los mockups iniciales, resultó ser problemático tener que solicitar cambios frecuentes al diseñador. Esta dinámica generaba retrasos significativos, ya que la respuesta del diseñador podía tardar días, seguida de más tiempo aún para implementar los ajustes necesarios. Dada esta situación, se decidió buscar si existian alternativas que pudieran agilizar el proceso.

En la búsqueda de soluciones, se encontró v0.dev **[25]**, una plataforma de inteligencia artificial desarrollada por Vercel. Esta herramienta prometía ser una solución innovadora para estos pequeños desafíos de diseño gráfico, gracias a su capacidad para generar y modificar diseños de manera autónoma y casi instantánea. v0.dev destacó por su competencia y la confianza que transmite Vercel, una empresa reconocida por su robustez tecnológica y su enfoque en soluciones de desarrollo web.

La integración de v0.dev en el proyecto permitió que se pudieran ajustar y crear nuevos prototipos de manera rápida y autónoma, sin depender de la disponibilidad del diseñador gráfico. Esto no solo aceleró el proceso de desarrollo, sino que también mejoró la capacidad para iterar sobre la marcha. Al adoptar esta herramienta de IA, se consiguió una mayor flexibilidad y eficiencia en el diseño y desarrollo de la interfaz, asegurando que se pudiera mantener un ritmo de trabajo constante y efectivo a lo largo del proyecto.

El proceso de aprendizaje para usar esta herramienta ha sido ágil, dado que su funcionamiento es similar al de otras plataformas de inteligencia artificial. Simplemente se introduce un prompt, ya sea una descripción o una imagen, y la herramienta automáticamente genera un prototipo basándose en esa información.

A continuación, se pueden observar varios ejemplos de componentes incorporados en la aplicación gracias a esta herramienta:

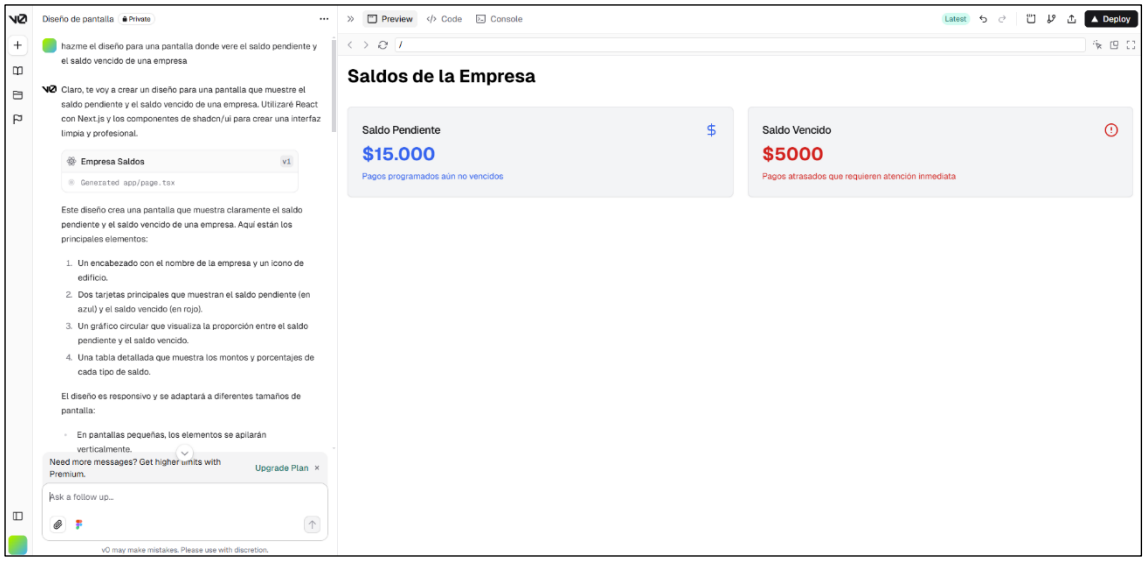


Figura 22. Mockup: Componentes de saldos de cliente. Fuente: Propia

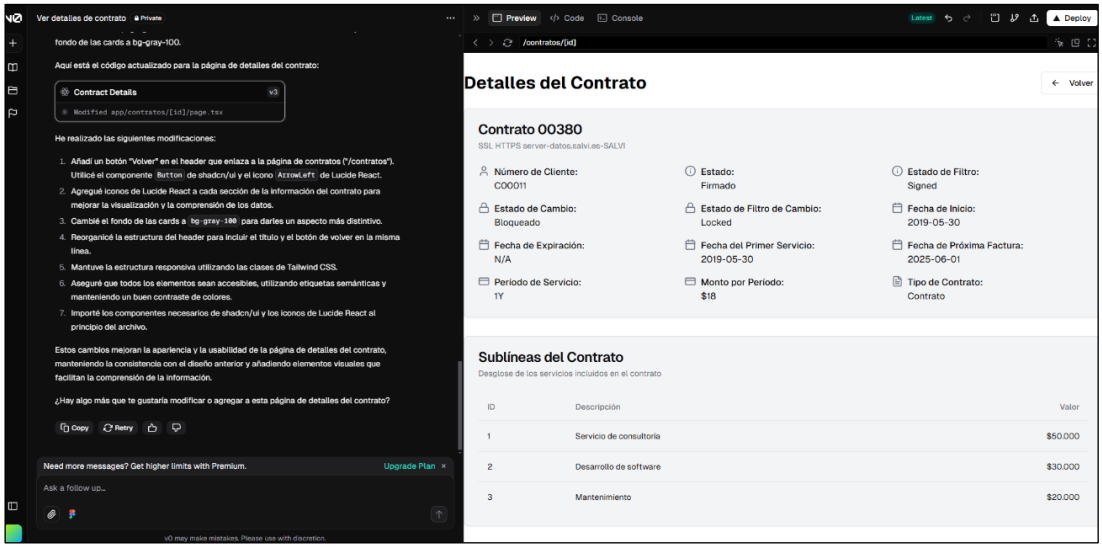


Figura 23. Mockup: Detalles de un contrato. Fuente: Propia

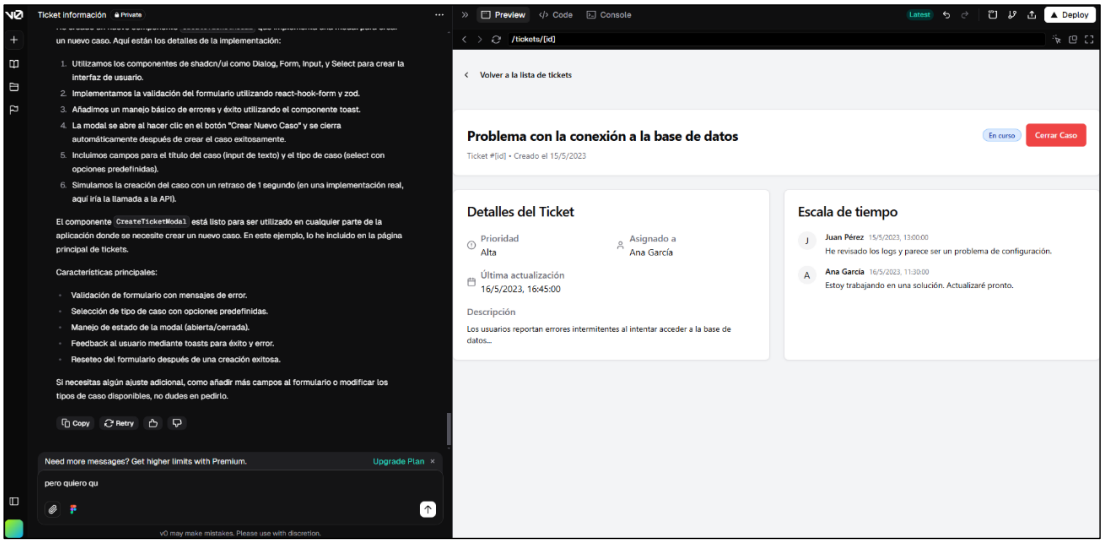


Figura 24. Mockup: Detalles de un ticket. Fuente: Propia

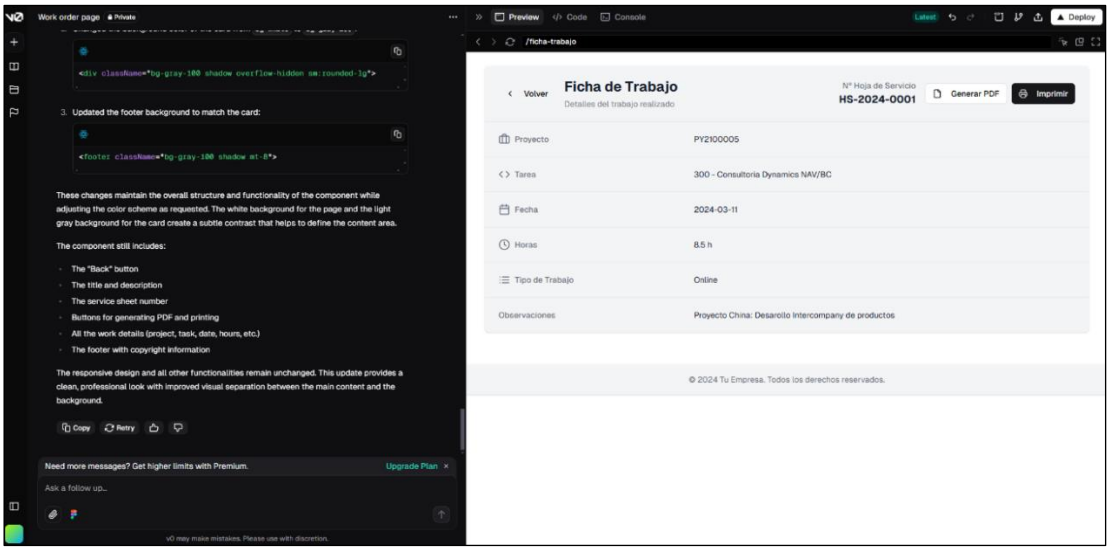


Figura 25. Mockup: Detalles de la hoja de servicio. Fuente: Propia

8.4 Arquitectura del Frontend

El proyecto se desarrolla utilizando React **[26]** y Vite **[27]**. Se ha elegido React debido a su capacidad para construir componentes reusables y eficientes, lo que es fundamental dada la estructura y los requerimientos visuales que se han podido observar en los mockups anteriores. La reusabilidad de los componentes en React permite mantener un código más limpio y modular, facilitando la mantenibilidad y la escalabilidad del proyecto.

Por otro lado, Vite es utilizado como el entorno de desarrollo y herramienta de construcción debido a su impresionante velocidad y eficiencia. Vite optimiza el proceso de desarrollo al permitir recargas instantáneas y un tiempo de arranque prácticamente nulo. Esto se logra mediante su enfoque moderno de servir sobre esmodules **[28]**, lo que reduce drásticamente el tiempo necesario para que los cambios se reflejen en el navegador.

La combinación de React y Vite proporciona una plataforma robusta y ágil para desarrollar una interfaz de usuario que no solo cumple con los requisitos técnicos y estéticos presentados en los mockups, sino que también ofrece una excelente experiencia de desarrollo y usuario final. Esta elección asegura que se puede construir una aplicación altamente interactiva y adaptable, con componentes que pueden ser fácilmente ajustados y reutilizados a través de diferentes partes del proyecto.

Cabe destacar que la escalabilidad y eficiencia de la aplicación se garantizan mediante el uso de dos herramientas clave de React: la reutilización de componentes y los hooks.

8.4.1 Reutilización de componentes

React permite la reutilización de componentes, lo cual aporta eficiencia y escalabilidad a la hora de desarrollar una aplicación. Al utilizar componentes previamente creados en diversas partes de una aplicación, se puede reducir el tiempo y el esfuerzo necesarios para el desarrollo, asegurando a la vez una experiencia de usuario consistente y facilitando el mantenimiento del código.

Ventajas:

- **Reducción de tiempo**: La utilización de componentes ya desarrollados previamente para nuevas funcionalidades disminuye el tiempo de desarrollo, permitiendo avanzar más rápido en los proyectos.

- **Menos errores**: La reutilización de código probado y comprobado reduce la posibilidad de sufrir nuevos errores.

- **Comportamiento predecible**: La interacción con componentes consistentes en diferentes partes de la aplicación permite a los usuarios aprender rápidamente cómo funcionan estos elementos, mejorando así la experiencia de usuario.

- **Actualizaciones centralizadas**: Modificar un componente reutilizable en un solo lugar puede actualizar su funcionalidad en todas las partes de la aplicación donde se utilice, lo que simplifica significativamente el mantenimiento.

- **Código menos fragmentado**: Al tener menos duplicidades en el código, hay menos fragmentación y es más fácil gestionar y actualizar la base de código.

- **Componentes Optimizados**: Los componentes que se utilizan en múltiples partes de una aplicación se pueden optimizar para mejorar el rendimiento global. Por ejemplo, se pueden memorizar para evitar renderizados innecesarios.

- **Carga de Recursos**: La reutilización de componentes también puede contribuir a una mejor gestión de la carga de recursos, ya que se minimiza la cantidad de código diferente que necesita cargarse y ejecutarse.

Como se ha visto anteriormente, la reutilización de componentes trae muchas ventajas y mejoras en el desarrollo de proyectos. Sin embargo, es importante evitar extremos como la fragmentación excesiva. No es práctico convertir todo en componentes; hacerlo puede tener el efecto contrario y complicar innecesariamente el proyecto. Es vital encontrar un equilibrio adecuado que permita maximizar la eficiencia sin sacrificar la claridad y la mantenibilidad del código.

8.4.2 Hooks

Los hooks **[29]** en React son esenciales para crear componentes funcionales de forma más limpia y eficiente. Estos permiten manejar el estado y los efectos secundarios sin tener que convertir sus componentes en clases. Esto simplifica el código y facilita tanto la reutilización de la lógica como la gestión del ciclo de vida de los componentes.

Ventajas:

- **Simplificación del código**: Los hooks permiten utilizar estado y otras características de React sin escribir una clase, lo que simplifica significativamente el código y lo hace más legible. Por ejemplo, el hook useState permite añadir estado local a un componente funcional, y useEffect permite realizar efectos secundarios en componentes funcionales.

- **Reutilización de lógica de estado**: Antes de los hooks, reutilizar la lógica de estado entre componentes podía ser complicado y requería patrones menos intuitivos como los componentes de orden superior o los componentes de renderizado. Los hooks permiten extraer la lógica de estado a funciones reutilizables, facilitando la compartición de esta lógica entre varios componentes sin alterar la jerarquía del árbol de componentes.

- **Menos bugs y complejidad**: Al usar clases, es fácil cometer errores con las peculiaridades del manejo de *this*, o tener problemas con el comportamiento inesperado debido a la reutilización incorrecta de los métodos del ciclo de vida. Los hooks eliminan muchas de estas trampas al ofrecer una API más directa y funcional.

- **Mejor desempeño con memorización**: Hooks como useMemo y useCallback permiten optimizar el rendimiento de los componentes evitando operaciones costosas o innecesarias.

8.5 Arquitectura del Backend

8.5.1 Interno

En el backend de la aplicación se utiliza Node.js **[30]**, ya que actúa como intermediario entre el frontend, los servicios de Microsoft y la base de datos. Esta configuración permite centralizar y gestionar las lógicas de negocio de manera eficiente, asegurando que todas las interacciones entre el usuario final y los sistemas subyacentes sean coherentes y seguras.

Para garantizar esta eficiencia y seguridad, se utilizan dos herramientas dentro de Node.js:

Express [31]: Este framework ayuda a gestionar las solicitudes y respuestas del servidor de manera eficiente. Con Express, se pueden crear rutas específicas y añadir middlewares que ayudan a controlar la información que circula y el tipo de peticiones que se realizan, facilitando así una mejor organización y manejo del tráfico en el servidor.

Axios [32]: Esta librería de JavaScript se especializa en realizar peticiones HTTP. Con esta librería se realizan todas las llamadas a los servicios externos de Microsoft.

8.5.1.1 Documentación de la API

Para futuros cambios en la aplicación es importante tener una buena documentación de la API. Por ese motivo, se ha implementado una documentación interactiva utilizando Swagger **[33]**, una herramienta líder en el mundo del desarrollo de software para describir, consumir y visualizar servicios web RESTful **[34]**.

Swagger permite documentar cada llamada a la API de manera detallada, especificando los parámetros necesarios para cada método y la estructura de la respuesta esperada. Además, la documentación generada es interactiva, lo que permite a los desarrolladores probar las llamadas a la API directamente desde la interfaz de usuario del navegador, ofreciendo una manera rápida y práctica de ver cómo la API responde a diferentes entradas y condiciones sin necesidad de escribir código adicional.

Esta documentación se ha segmentado de la siguiente manera:

Auth

Dentro de esta sección se encuentran los endpoints relacionados con la gestión de la autenticación al sistema. En este caso, el backend solo conecta con la base de datos o ejecuta lógicas de negocio internas, como la encriptación de la password o la generación de una cookie para poder acceder a la aplicación.

En la Figura 26 se puede observar el listado de todos los endpoints disponibles de este apartado.

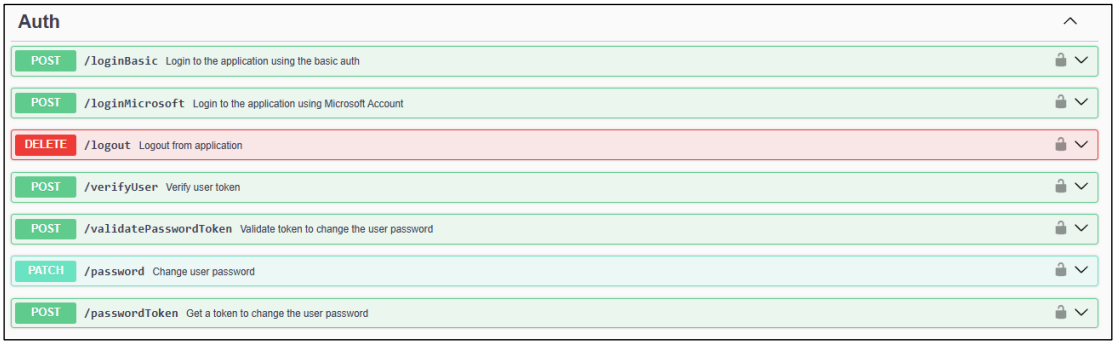


Figura 26. Endpoints de la sección "Auth". Fuente: Propia

Las llamadas */loginBasic* y */loginMicrosoft* son excepciones en el sistema, ya que no requieren de la cookie que verifica la autenticación del usuario. De hecho, si la autenticación es correcta, estas funciones son las encargadas de generar la cookie que registra al usuario autenticado. En la Figura 27 se observa el ejemplo de la llamada */loginBasic*.

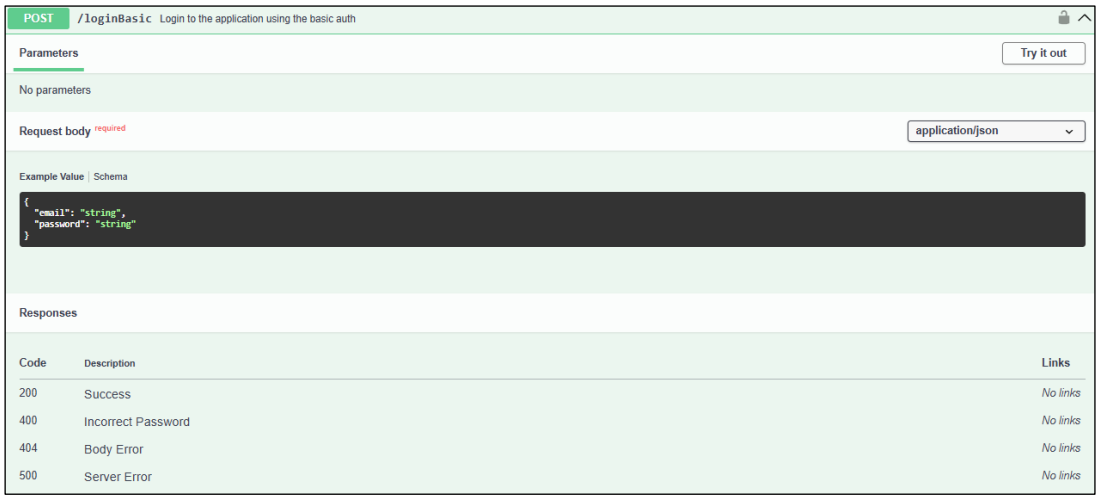


Figura 27. Ejemplo de la llamada "/loginBasic". Fuente: Propia

La llamada *"/logout"* se encarga de eliminar la cookie del sistema para eliminar la sesión del usuario. El ejemplo de la llamada se puede ver en la Figura 28.

Figura 28. Ejemplo de la llamada "/logout". Fuente: Propia

Internal

Dentro de esta sección se encuentran los endpoints relacionados con la gestión de usuarios y permisos. En este caso, el backend solo conecta con la base de datos o ejecuta lógicas de negocio internas.

En la Figura 29 se puede observar el listado de todos los endpoints disponibles de este apartado.

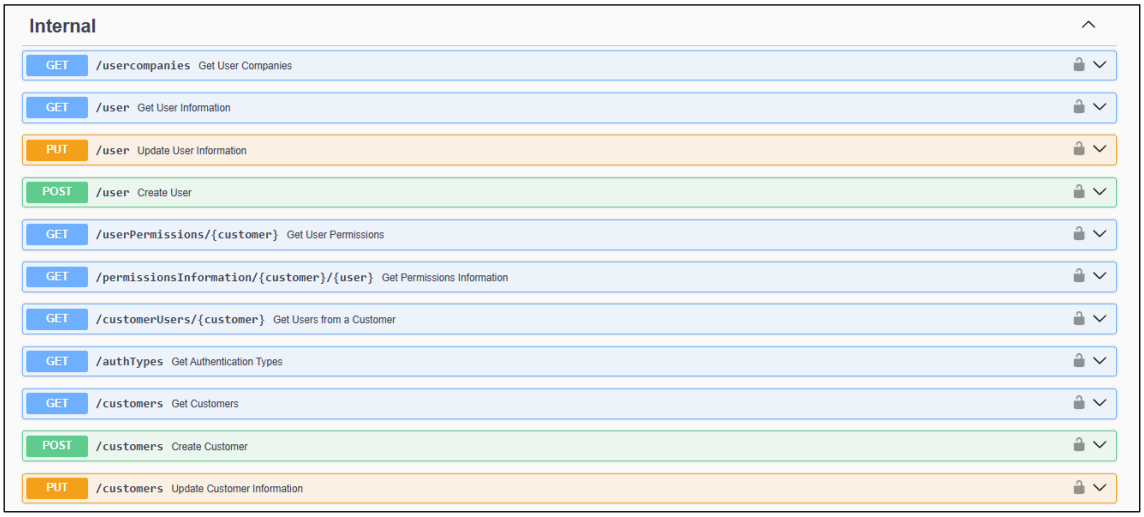


Figura 29. Endpoints de la sección "Internal". Fuente: Propia

Todos los endpoints de esta sección que utilizan GET funcionan de la misma manera. Para identificar al usuario, se transmite su información mediante una cookie en la cabecera, que el backend luego descomprime. Si es necesario algún parámetro adicional para identificar a un cliente, este se incluye en la URL, siguiendo los principios de REST. En la Figura 30 se puede ver un ejemplo de una llamada GET.

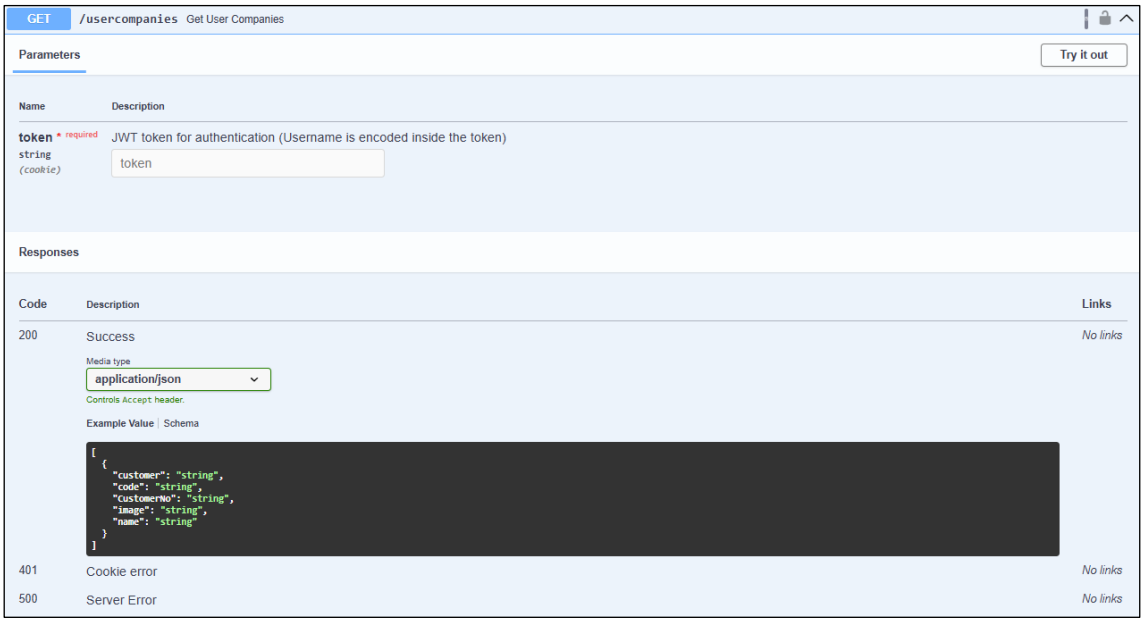


Figura 30. Ejemplo de un GET en la sección "Internal". Fuente: Propia

Los endpoints que emplean el método PUT o POST en esta sección requieren tanto el body de la solicitud, que contiene la información a modificar, como una cookie que verifica que el usuario esté autenticado. En la Figura 31 se observa un ejemplo de PUT y en la Figura 32 se observa un ejemplo de POST.

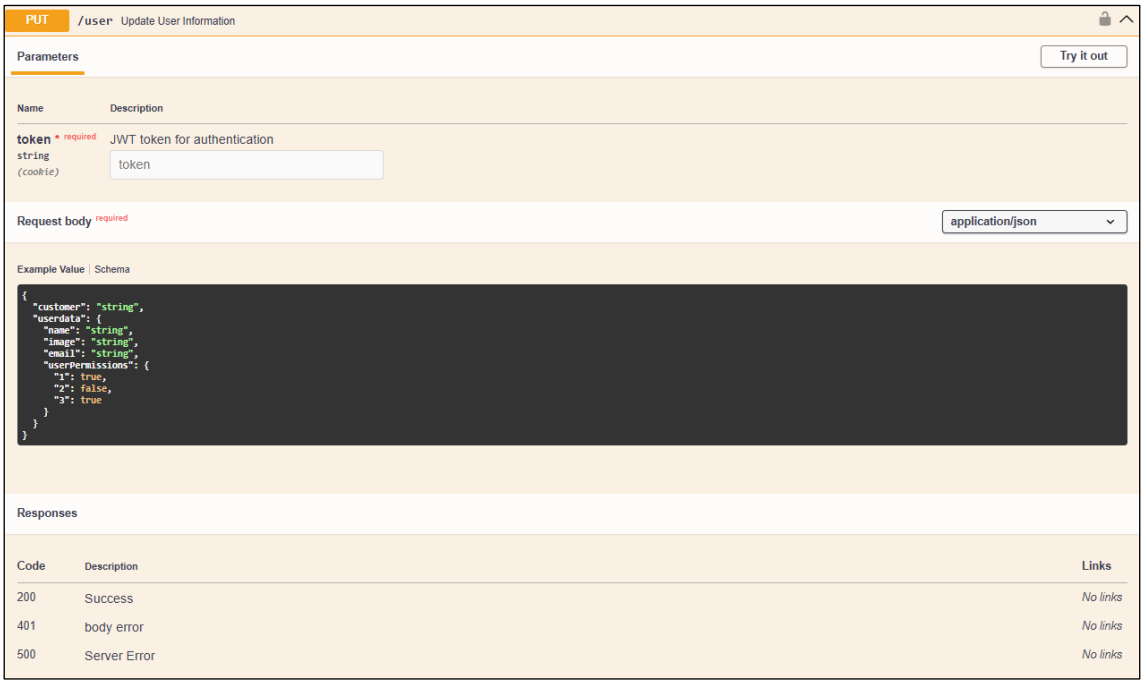


Figura 31. Ejemplo de un PUT en la sección “Internal”. Fuente: Propia

Figura 32. Ejemplo de un POST en la sección "Internal". Fuente: Propia

Business Central

Esta sección incluye los endpoints principalmente enfocados en la gestión de administración. En este caso, el backend establece conexión con el servicio de Microsoft Dynamics 365 Business Central y se aplican las lógicas de negocio requeridas.

En la Figura 33 se puede observar el listado de todos los endpoints disponibles de este apartado.

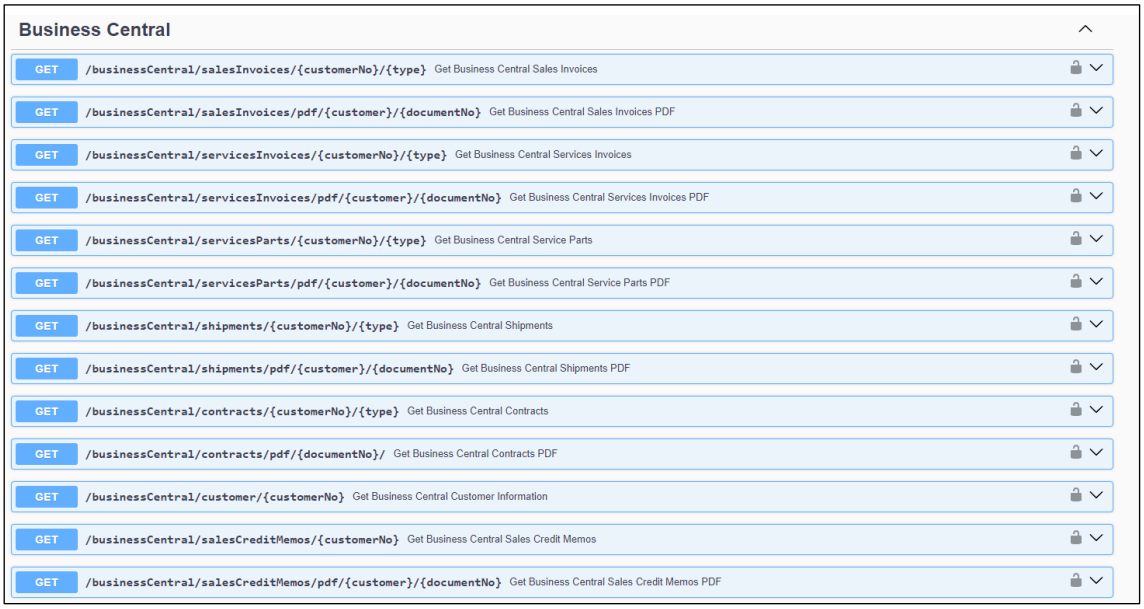


Figura 33. Endpoints de la sección "Business Central". Fuente: Propia

Todos los endpoints de esta sección utilizan el método GET. Tal y como se ha comentado en la anterior sección, el usuario tiene acceso a los Endpoints mediante una cookie y si es necesario algún parámetro adicional, este se incluye en la URL, siguiendo los principios de REST. En la Figura 34 se puede ver un ejemplo de una llamada GET.

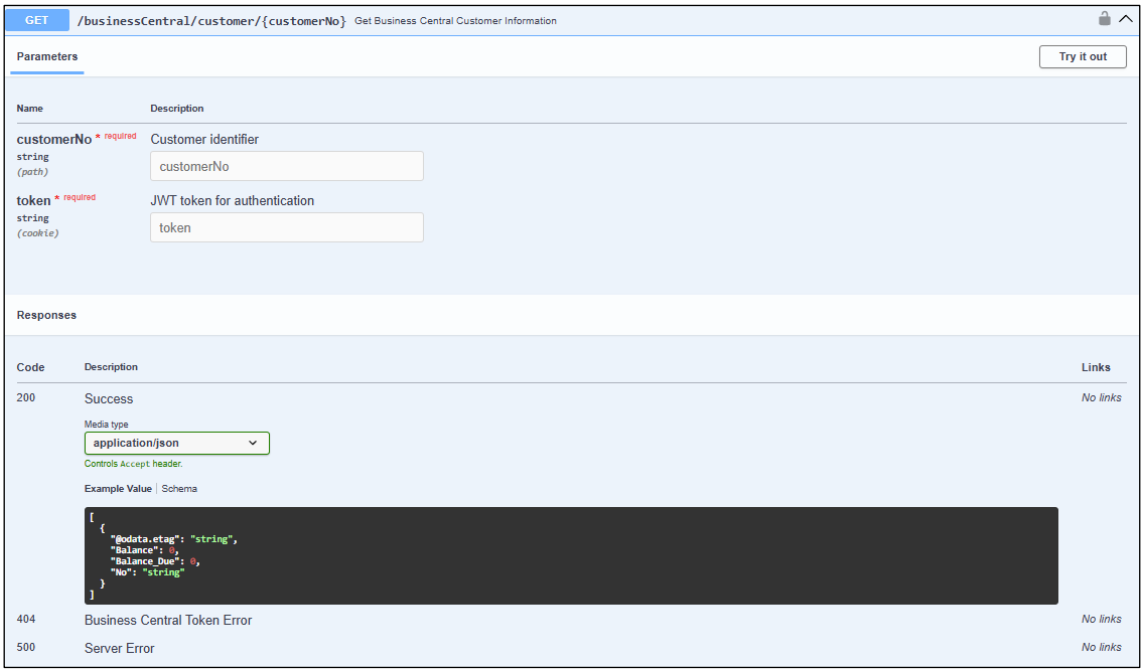


Figura 34. Ejemplo de un GET en la sección "Business Central". Fuente: Propia

Customer Service

Esta sección incluye los endpoints principalmente enfocados en la gestión de contactos y de soporte. En este caso, el backend establece conexión con el servicio de Microsoft Dynamics 365 Customer Service y se aplican las lógicas de negocio requeridas.

En la Figura 35 se puede observar el listado de todos los endpoints disponibles de este apartado. En la figura Figura 36 se observa un ejemplo de una llamada GET, en la Figura 37 un ejemplo de una llamada en POST y en la Figura 38 un ejemplo de una llamada PATCH.

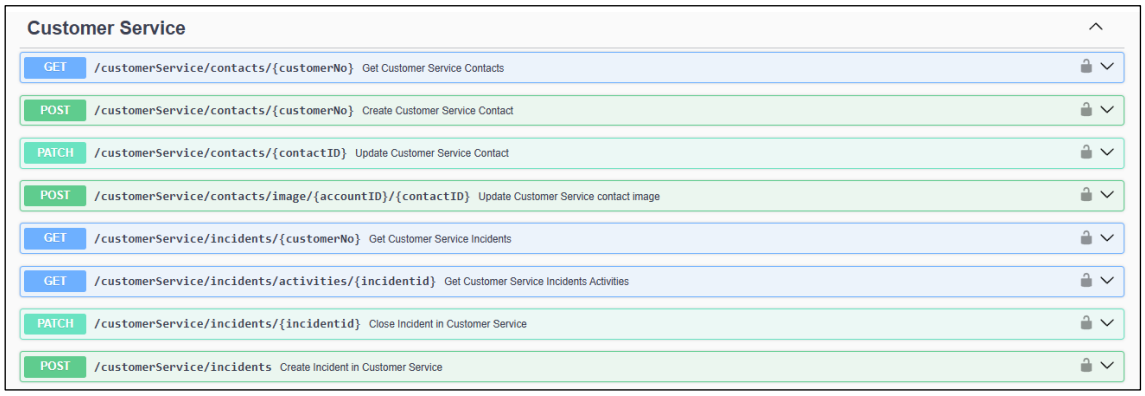


Figura 35. Endpoints de la sección "Customer Service". Fuente: Propia

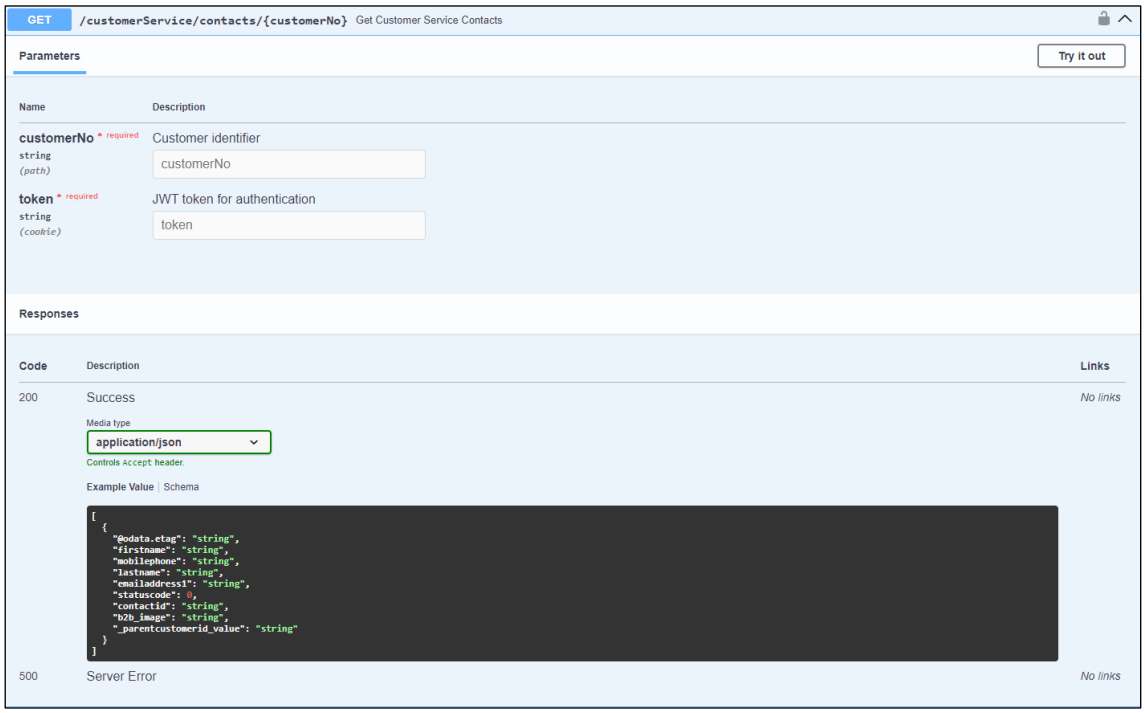


Figura 36. Ejemplo de un GET en la sección “Customer Service”. Fuente: Propia

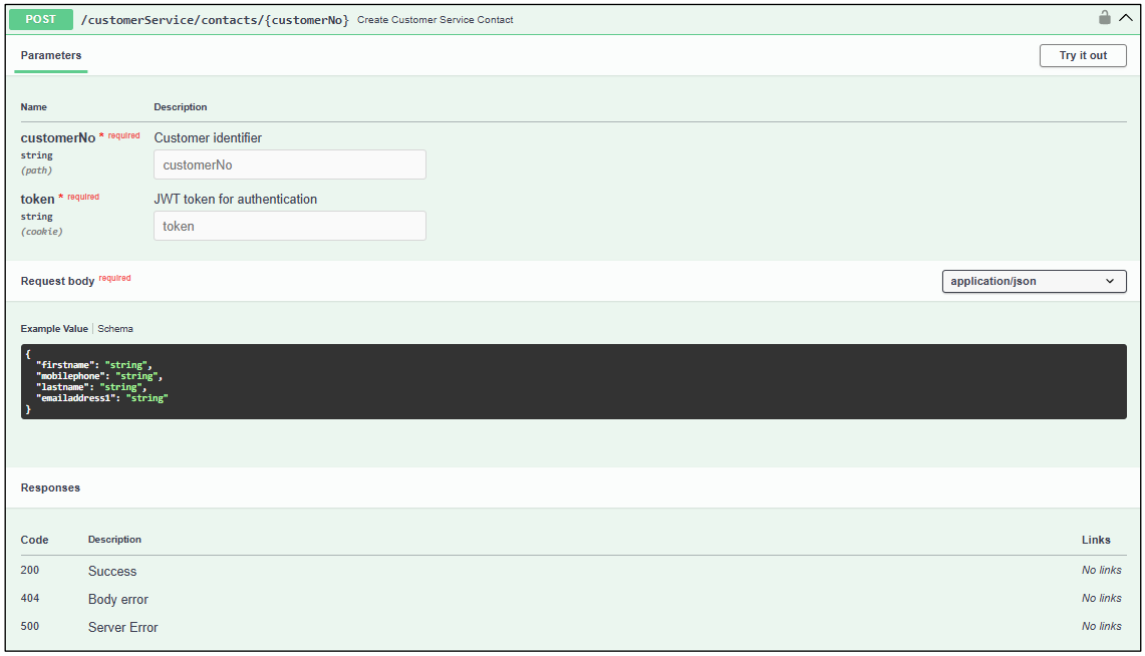


Figura 37. Ejemplo de un POST en la sección "Customer Service". Fuente: Propia

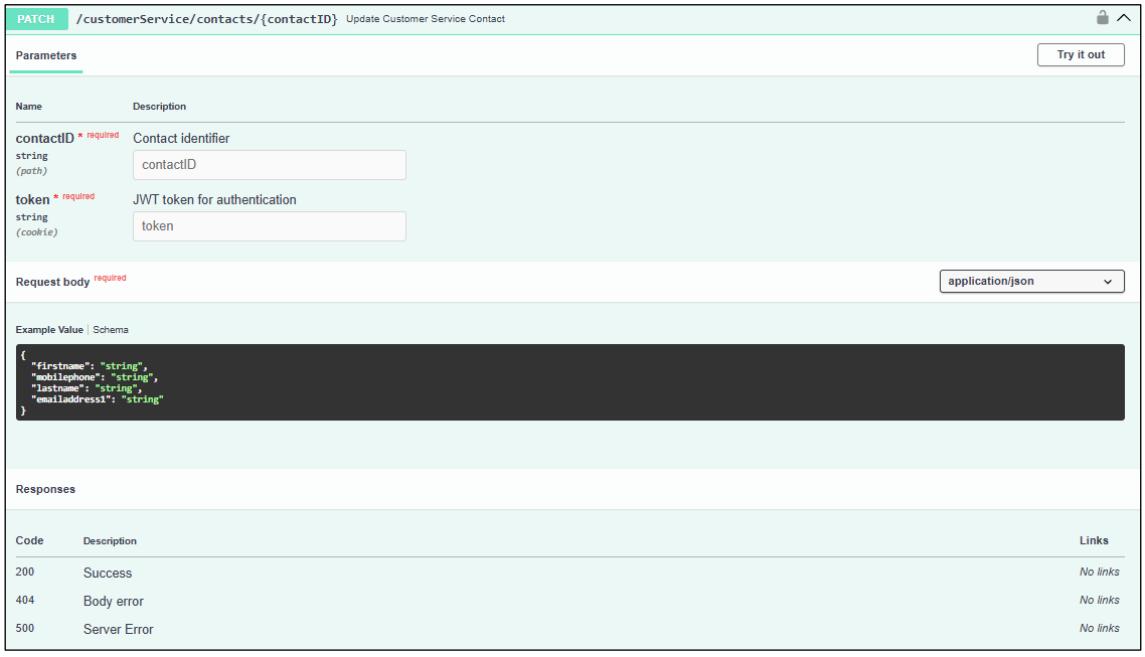


Figura 38. Ejemplo de un PATCH en la sección "Customer Service". Fuente: Propia

POWER BI

Esta sección incluye el único endpoint que conecta con el servicio de Microsoft Dynamics 365 Power Bi para poder obtener el informe de casos que se muestra en la gestión de soporte.

La sección y la descripción del único endpoint que se encuentra en esta segmentación se muestra en la Figura 39.

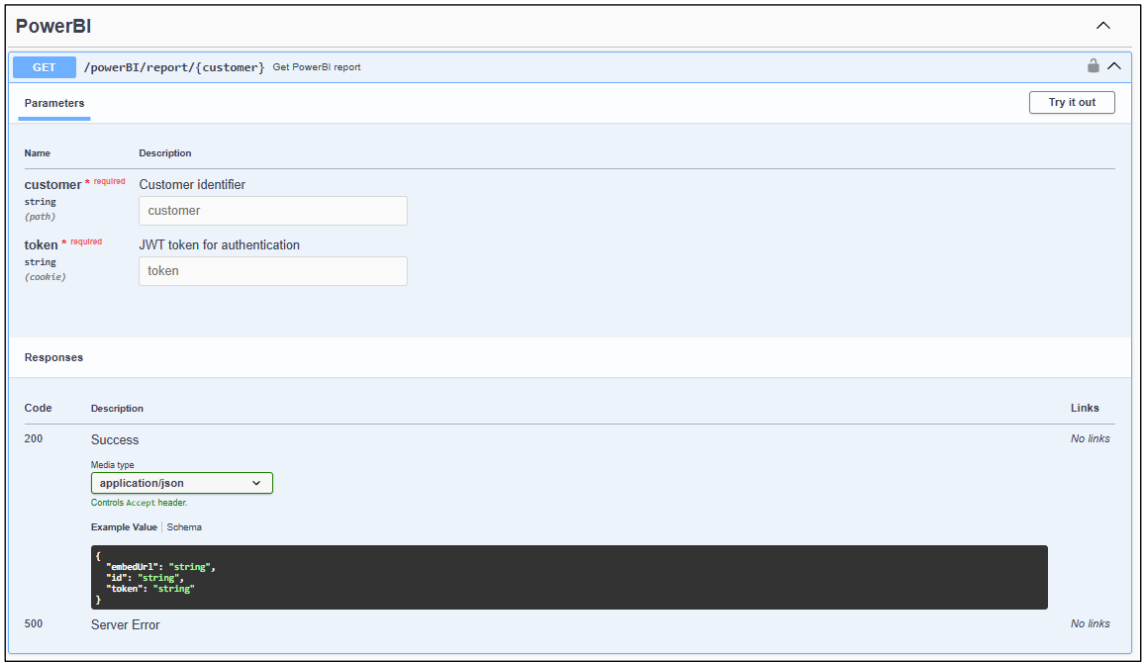


Figura 39. Sección "Power BI" completa. Fuente: Propia

8.5.2 Servicios Externos

En el ámbito de la gestión de recursos empresariales, la aplicación utiliza endpoints generados con el lenguaje AL (Application Language) **[35]**, que es el lenguaje requerido por Microsoft para la personalización de Business Central. AL es un lenguaje de programación específicamente diseñado para este sistema ERP, ofreciendo una sintaxis y funcionalidades que comparten similitudes con SQL, lo cual facilita su adopción por parte de desarrolladores familiarizados con bases de datos relacionales.

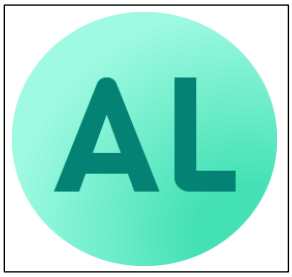


Figura 40. Logo del lenguaje AL. Fuente: Microsoft

AL permite a los desarrolladores construir y extender tablas dentro del entorno de Dynamics 365 Business Central, habilitando la creación de estructuras de datos personalizadas que se alinean perfectamente con los procesos empresariales específicos de cada organización. Más allá de la manipulación de datos, AL es también una herramienta poderosa para implementar lógicas de negocio complejas y adaptativas. Esto incluye la capacidad de escribir procedimientos almacenados, subscribirse a eventos, y diseñar interfaces de usuario personalizadas dentro del mismo entorno del ERP.

Para la implementación del servicio Dynamics 365 Customer Service en la aplicación, se ha considerado el uso de Power Platform **[36]** en conjunto con Power Automate **[37]** para personalizar el núcleo de la aplicación. Sin embargo, en este caso, no ha sido necesario recurrir a personalizaciones ya que los endpoints estándar que proporciona la API de Dynamics 365 Customer Service son suficientes para obtener la información requerida.

8.5.3 Base de datos

En el proyecto se utiliza MariaDB **[38]** como base de datos principal. Es una base de datos relacional conocida por su robustez y excelente rendimiento. En ella se almacenan todos los datos relacionados con la información de los usuarios y sus permisos. Este hecho, garantiza un acceso controlado y seguro a la plataforma. Además, MariaDB ofrece la velocidad necesaria para realizar operaciones rápidas directamente en la base de datos, evitando la necesidad de recurrir a servicios externos para tareas que requieren inmediatez. Esto optimiza los tiempos de respuesta.

8.6 Patrones de diseño

Para simplificar la estructura y aumentar la escalabilidad de la aplicación, se han implementado varios patrones de diseño. Estos patrones no solo ayudan a resolver problemas comunes de desarrollo de software, sino que también mejoran la mantenibilidad y flexibilidad del sistema. Para el proyecto, principalmente se han utilizado los siguientes patrones:

8.6.1 Patrón Contenedor/Presentación

Este patrón, también conocido como "*Container/Presentational Pattern*" **[39]**, es una metodología popular en el desarrollo de aplicaciones con React, aunque puede aplicarse en otros frameworks de UI también. Este patrón ayuda a separar la lógica de la gestión de datos y el estado de la UI de la presentación visual y la disposición de los componentes. Dentro de este patrón se encuentran dos componentes principales:

Componentes Presentacionales: Estos componentes solo se preocupan por el aspecto visual. Reciben datos exclusivamente a través de props[7] para que puedan mostrarlos. Al no depender de servicios ni datos hacen que sean más fáciles de integrar y reutilizar.

Componentes Contenedores: Los componentes contenedores manejan la lógica de negocio y el estado del componente. Se encargan de mantener estado y llevar a cabo la gestión de datos. Además, son los encargados de transmitir datos a los componentes presentacionales.

Al utilizar este patrón se obtienen varias ventajas:

- **Separación de responsabilidades:** Este patrón mantiene la lógica de negocio separada de los componentes más dedicados al aspecto visual con los que gestionan datos y lógicas de negocio.

- **Reutilización de componentes:** Los componentes presentacionales pueden ser fácilmente reutilizados, ya que no están fuertemente acoplados a la lógica de datos.

- **Facilita el Testing:** Testear componentes que contienen exclusivamente UI sin lógica de estado o de negocio es mucho más sencillo.

- **Mejora en la mantenibilidad:** Es más fácil hacer cambios en la lógica de negocio o en la UI sin afectar el uno al otro.

En la Figura 41, se ilustra un ejemplo de un componente presentacional. Este componente está enfocado exclusivamente en la interfaz de usuario y recibe datos a través de parámetros para mostrarlos, sin involucrarse en la gestión de los datos. Por otro lado, la Figura 42 muestra cómo un componente contenedor se encarga de manejar la obtención y procesamiento de datos antes de enviarlos al componente presentacional.

[7] props: Son los atributos que el componente padre envía al componente hijo y este los recibe para poder utilizarlos. Se asemejaría a pasar un parámetro en una función.

```jsx
src > Areas > Administration > Components > KPI.jsx > ...
...

export function KPI({pImage, pNumber, pText}) {
    return (
        <div className="flex flex-col col-span-1 p-4 rounded-lg bg-[#F3F3F3] relative">
            <img className={"ml-auto"} src={pImage} alt="" width={30}/>
            <p className="text-administration-card-number font-light leading-none">{pNumber}</p>
            <p className="text-administration-card-title font-extralight text-[#ABA9A9] leading-none">{pText}</p>
            <div className="absolute bg-[#E4E4E4] h-[10px] w-[90%] bottom-[-10px] rounded-bl-lg rounded-br-lg left-[5%]"></div>
        </div>
    )
}
```

Figura 41. Ejemplo de componente presentacional. Fuente: Propia

```jsx
useEffect(() => {
    setLoading(true);
    Promise.all([
        f_getKPIInvoices(selectedCompany, setnumber_invoices),
        f_getTopInvoices(selectedCompany,settopInvoices),
        f_getKPIServicesPart(selectedCompany, setnumber_servicesParts),
        f_getKPIShipments(selectedCompany, setnumber_shipments),
        f_getKPIContracts(selectedCompany, setnumber_ActiveContracts)
    ]).then(() => {
        setLoading(false);
    })
}, [selectedCompany])

return (
    <>
        {loading ? (
            <p>Cargando</p>
        ) : (
            <>
                <div className="col-span-12 grid grid-cols-4 gap-4 mt-12">
                    <KPI pImage={"administration/Invoices.svg"} pNumber={number_invoices} pText={"Facturas"}></KPI>
                    <KPI pImage={"administration/Services.svg"} pNumber={number_servicesParts} pText={"Servicios"}></KPI>
                    <KPI pImage={"administration/Shipments.svg"} pNumber={number_shipments} pText={"Albaranes"}></KPI>
                    <KPI pImage={"administration/Contracts.svg"} pNumber={number_ActiveContracts} pText={"Contratos activos"}></KPI>
                </div>
```

Figura 42. Ejemplo de componente contenedor. Fuente: Propia

8.6.2 Patrón *Provider*

El patrón Provider **[40]** de React es una técnica que utiliza el contexto de React para pasar datos a través del árbol de componentes sin tener que pasar props manualmente en cada nivel. Este patrón es especialmente útil en aplicaciones grandes donde ciertos datos o comportamientos necesitan estar disponibles en muchos componentes a diferentes niveles de la jerarquía. La Figura 43 representa el problema descrito y la solución con el patrón.

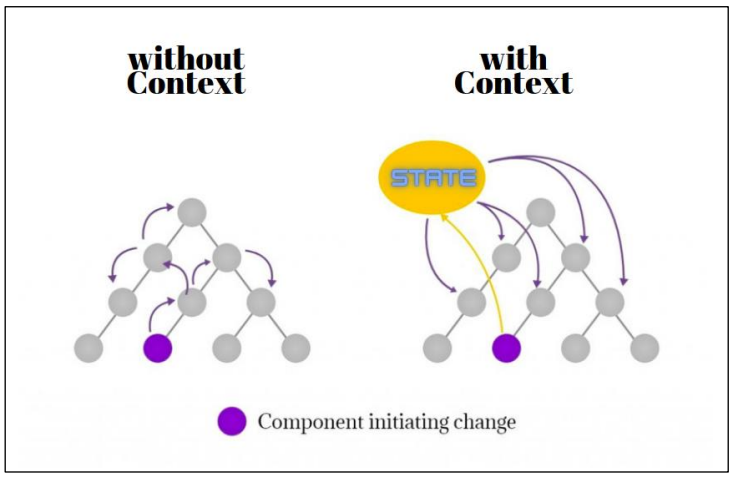


Figura 43. Patrón Provider. Fuente: Medium

El contexto proporciona una forma de pasar datos a través del árbol de componentes sin tener que pasar props manualmente a cada nivel. En React, esto se logra utilizando un Provider para el contexto que encapsula parte del árbol de componentes, y luego utilizando un Consumer o el hook *useContext* para acceder a esos datos en los componentes descendientes. Al utilizar este patrón se obtienen varias ventajas:

- **Simplifica la gestión de estado:** Permite manejar el estado global de manera más sencilla sin recurrir a soluciones más complejas como Redux, especialmente en aplicaciones que no requieren un manejo de estado muy complejo.

- **Reusabilidad:** Facilita la reutilización de lógicas y estados entre componentes que no están directamente conectados.

- **Mejora la legibilidad:** Reduce el prop-drilling[8], lo que hace que el código sea más limpio y fácil de entender.

Sin embargo, el uso de este patrón no está exento de desventajas. Un uso excesivo del contexto puede resultar en una sobrecarga si se intenta manejar cada pequeño aspecto del estado a través de diferentes contextos. Además, los componentes que dependen en gran medida del contexto pueden volverse menos reutilizables, ya que su funcionamiento puede estar estrechamente ligado a un contexto específico. Esto puede complicar su integración en otras partes de la aplicación o su uso en proyectos diferentes. Por lo tanto, es crucial utilizar este patrón de manera consciente y moderada para evitar estos problemas.

La Figura 44 enseña la creación de un contexto, en concreto, se genera un contexto donde se puede acceder a los valores dentro de *áreas*, el valor de la *selectedPage* y la funcion setselectedPage para cambiar el estado de la variable *selectedPage*. Por otro lado, la Figura 45 enseña como consumir un contexto para poder acceder a los valores en cualquier componente.

```
<GlobalContext.Provider value={{areas, selectedPage, setSelectedPage}}>
```

Figura 44. Ejemplo de provider en React. Fuente: Propia

```
const {areas, setSelectedPage} = useContext(GlobalContext);
```

Figura 45. Ejemplo de consumidor en react. Fuente: Propia

[8] Prop-drilling: Es pasar datos directamente a través de múltiples niveles de componentes en una aplicación React.

8.6.3 Patrón *Middleware*

Este patrón **[41]** ha sido ampliamente empleado en el backend del proyecto, sirviendo como una capa intermedia que ofrece una variedad de servicios. En particular, se ha centrado en la gestión de la autenticación. Todas las llamadas a los endpoints del backend requieren un token de autorización válido; de lo contrario, no recibirán respuesta y resultarán en un error. La Figura 46 muestra una representación gráfica de cómo funciona este patrón.

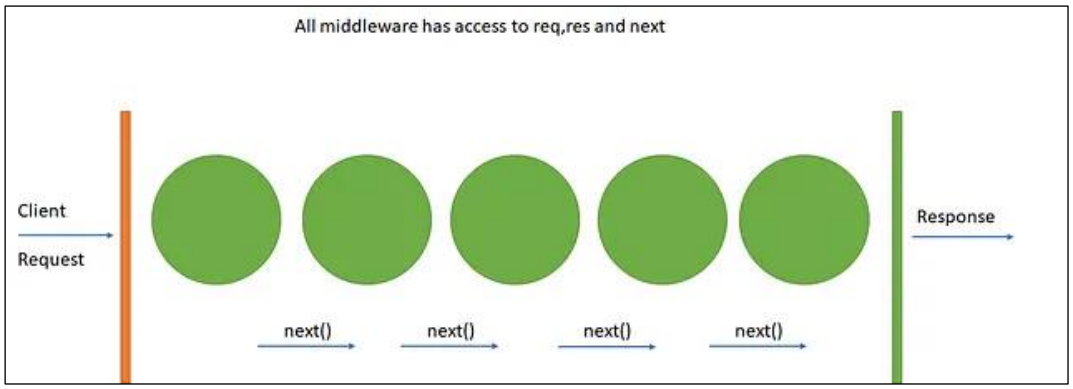


Figura 46. Patrón middleware. Fuente: Medium

8.6.4 Singleton

Este patrón, que se ha utilizado tanto en backend como en frontend, es un principio de diseño de software que restringe la instanciación de una clase a un solo objeto. Este es útil cuando necesitas asegurar que una clase tenga exactamente una instancia, con un punto de acceso global a esa instancia. Un ejemplo en el backend de singleton se muestra en la Figura 53, ya la instancia que contiene la configuración de la base de datos debe de ser única para todo el proyecto.

8.7 Ejemplos de diagramas de secuencia

Todos los diagramas de secuencia de la aplicación siguen el mismo ejemplo que el de Figura 47. En este flujo, cuando el usuario hace presiona el botón "Servicios", el frontend navega hacia esa página específica. En ese momento, el componente correspondiente se actualiza y comienza a esperar la recepción de las hojas de servicio. Para obtener estas hojas, el frontend invoca una función en el controlador de administración, que a su vez realiza una petición GET al backend.

El backend, antes de recuperar cualquier dato, verifica la validez del token de autenticación. Si el token no es válido o ha expirado, el backend realiza una solicitud para obtener un nuevo token de Business Central. Una vez obtenido, el backend actualiza este nuevo token en la base de datos para su uso en futuras solicitudes. Solo después de asegurar que el token es válido, el backend procede a obtener los datos requeridos de la API de Business Central y los envía de vuelta al frontend para su visualización.

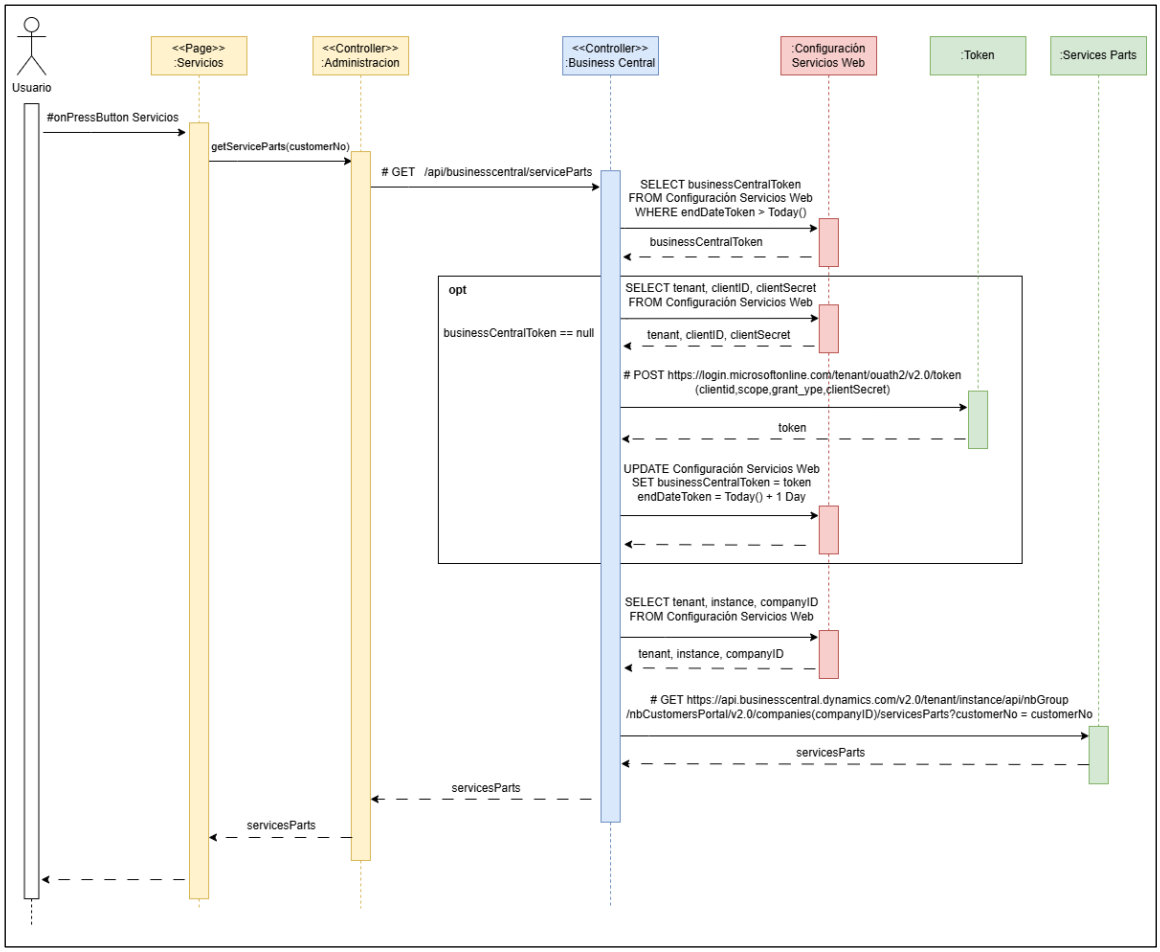


Figura 47. Diagrama de secuencia. Fuente: Propia

9 Implementación

En este capítulo se detalla cómo se han implementado el frontend y el backend, y cómo se ha desplegado la aplicación. Tanto para el frontend como para el backend, se ha utilizado el editor *Visual Studio Code*, compatible con todas las tecnologías empleadas.

9.1 Implementación del frontend

Se ha creado un proyecto exclusivo para el frontend, dónde está el código desarrollado en React. La organización de las carpetas y los archivos raíz utilizados se muestra en la Figura 48.

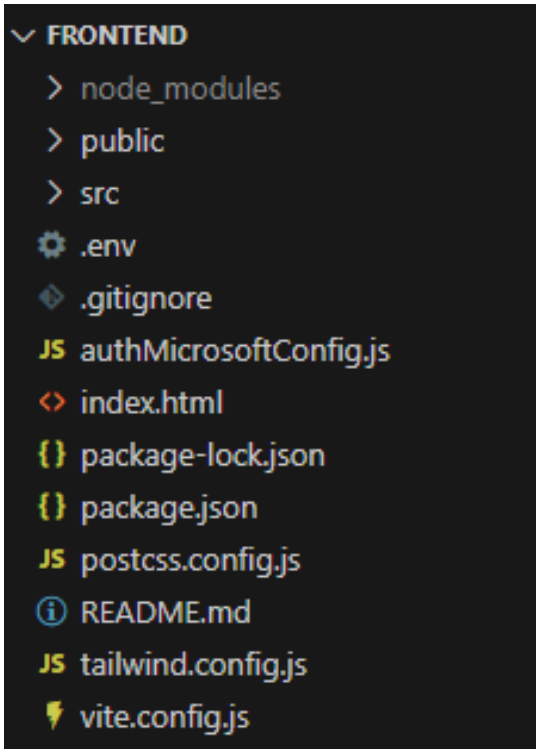


Figura 48. Estructura de carpetas – Frontend. Fuente: Propia

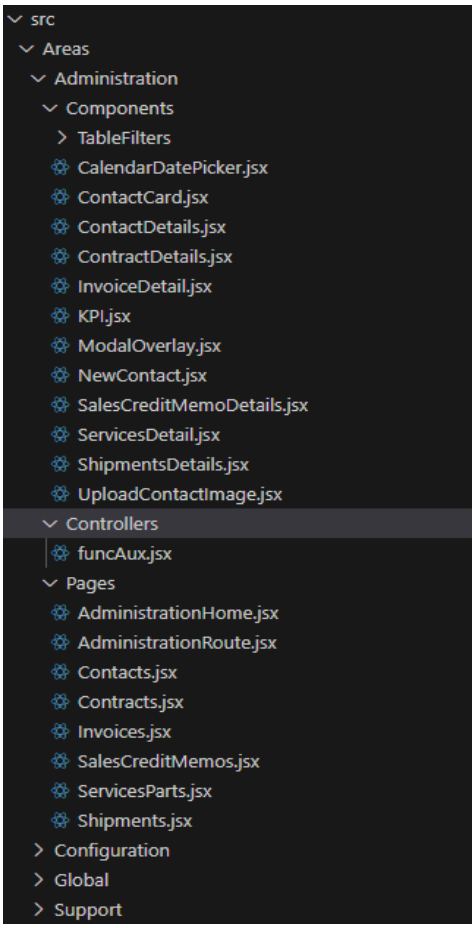


Figura 49. Estructura de ficheros en un área - Frontend. Fuente: Propia

Dentro de la carpeta *src*, que es la carpeta raíz que contiene todo el código fuente de la aplicación, se encuentra un directorio que contiene las diferentes áreas organizadas en subcarpetas específicas. Dentro de cada área, se encuentran tres subcarpetas más:

- **Components:** Esta carpeta contiene los componentes reutilizables en toda la área. Incluye elementos de interfaz de usuario y cualquier otro componente visual que pueda ser empleado en varias partes de la aplicación para mantener una consistencia en el diseño y funcionalidad.

- **Controllers:** Contiene la lógica del área en concreto, donde se implementan funciones complejas para procesar la información de manera adecuada antes de presentarla al usuario. Esto incluye las llamadas a la API que permiten que la aplicación funcione correctamente.

- **Pages:** En esta carpeta se encuentran las páginas creadas dentro del área. Estas páginas son donde se da uso a los componentes previamente creados y a los controladores.

Dentro del *src*, también se encuentran los siguientes archivos:

- **CustomersPortal.jsx:** Un archivo JSX representando la página principal de la aplicación.
- **index.css:** Archivo que contiene el CSS para todo el proyecto.
- **main.jsx**: Archivo principal que actúa como punto de entrada de la aplicación, inicializando y cargando los componentes principales.

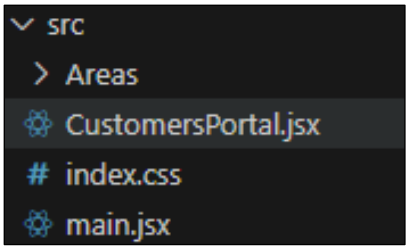


Figura 50. Ficheros principales - Frontend. Fuente: Propia

Por otro lado, volviendo a Figura 48, los archivos a tener en cuenta son:

- **.env**: Archivo donde se definen las variables de entorno.
- **.gitignore**: Archivo usado por Git para excluir archivos y directorios..
- **authMicrosoftConfig.js**: Archivo de configuración específico para la autenticación mediante Microsoft.
- **index.html**: Punto de entrada HTML para la aplicación
- **package-lock.json** & **package.json**: package.json guarda la configuración del proyecto y la lista de dependencias. package-lock.json asegura que las instalaciones de dependencias sean consistentes en diferentes entornos.
- **postcss.config.js**: Configuración para PostCSS, una herramienta utilizada para transformar CSS de tailwindcss.
- **tailwind.config.js**: Archivo de configuración de Tailwind CSS, uno de los framework de CSS que se han utilizado para el desarrollo.
- **vite.config.js**: Archivo de configuración para Vite, especificando cómo se debe construir y servir la aplicación.

9.2 Implementación del backend

En esta sección se describe cómo se ha implementado el backend de la aplicación. Al tener que generar endpoints en Business Central, el backend se ha dividido en dos subproyectos.

9.2.1 Node.Js

Al igual que en el frontend, el backend también tiene su propio directorio. La Figura 51 muestra la estructura del proyecto de Node.Js.

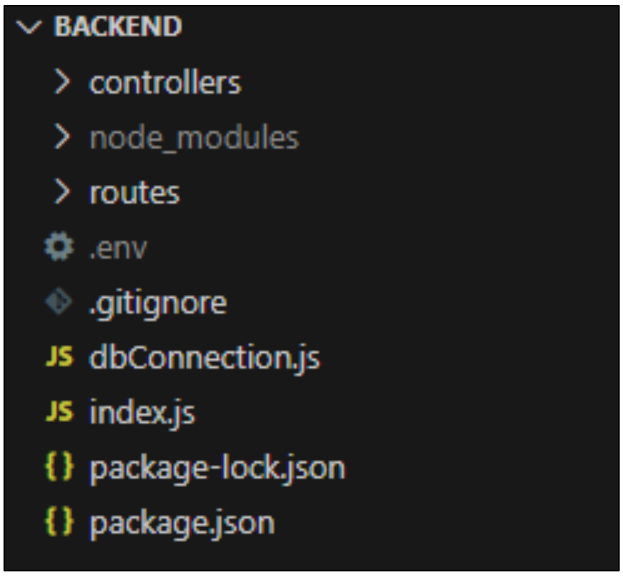


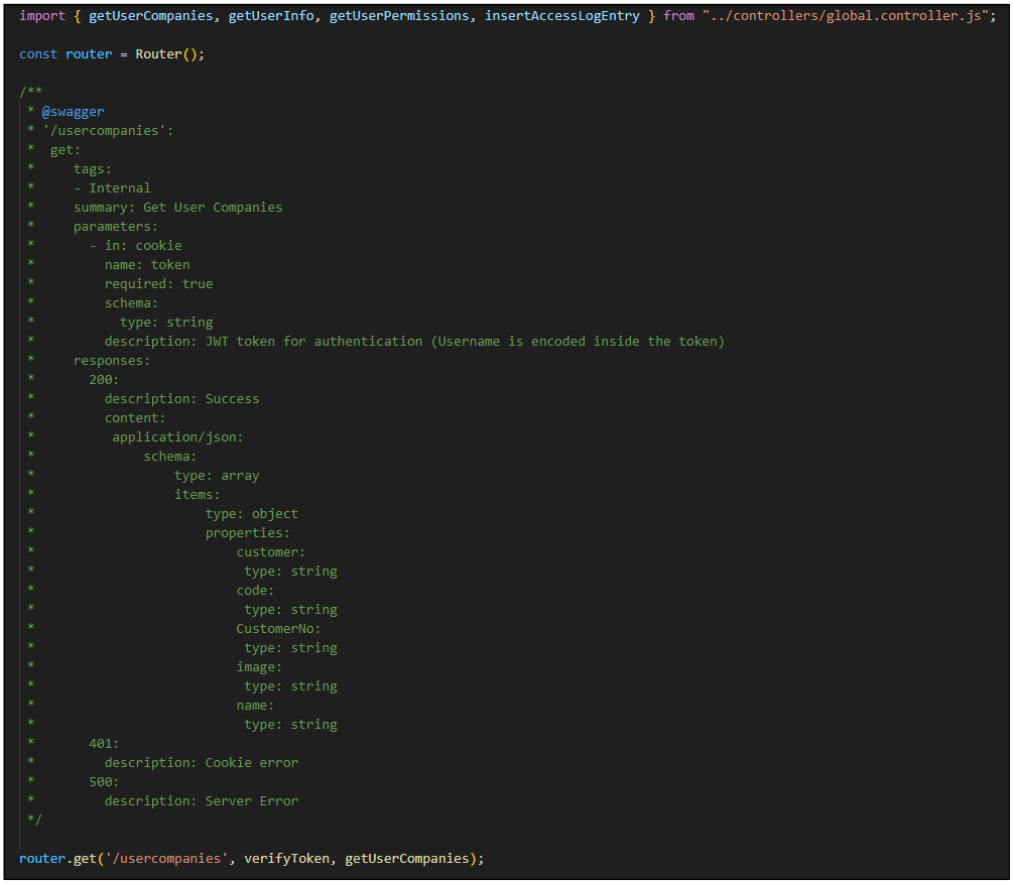
Figura 51. Estructura de carpetas de Node.Js. Fuente: Propia

Dentro de este proyecto, se han creado las siguientes carpetas para estructurar los diferentes componentes y funcionalidades:

- **controllers**: Esta carpeta contiene los archivos que manejan la lógica de la aplicación. Estos archivos procesan las solicitudes, ejecutan las operaciones que consultan a la base de datos o realizan llamadas a APIs de los servicios externos, y devuelven respuestas al cliente.

- **node_modules**: Contiene todos los paquetes instalados a través de npm.

- **routes**: En esta carpeta se definen las rutas de la aplicación. Las rutas son los diferentes endpoints de la API a los que el frontend puede acceder. Aquí se especifica qué controlador debe manejar cada solicitud, organizando así cómo se accede a las funcionalidades del backend. También dentro de cada ruta especificamos su referencia en el Swagger. En la Figura 52 muestra un ejemplo, en concreto, la ruta de */usercompanies*.

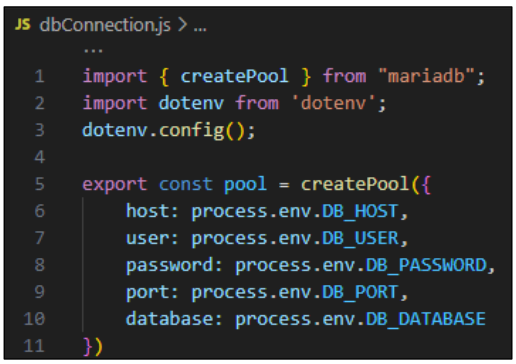


```
import { getUserCompanies, getUserInfo, getUserPermissions, insertAccessLogEntry } from "../controllers/global.controller.js";

const router = Router();

/**
 * @swagger
 * '/usercompanies':
 *  get:
 *     tags:
 *     - Internal
 *     summary: Get User Companies
 *     parameters:
 *       - in: cookie
 *         name: token
 *         required: true
 *         schema:
 *           type: string
 *         description: JWT token for authentication (Username is encoded inside the token)
 *     responses:
 *       200:
 *         description: Success
 *         content:
 *          application/json:
 *              schema:
 *                  type: array
 *                  items:
 *                      type: object
 *                      properties:
 *                          customer:
 *                           type: string
 *                          code:
 *                           type: string
 *                          CustomerNo:
 *                           type: string
 *                          image:
 *                           type: string
 *                          name:
 *                           type: string
 *       401:
 *         description: Cookie error
 *       500:
 *         description: Server Error
 */

router.get('/usercompanies', verifyToken, getUserCompanies);
```

Figura 52. Ejemplo de un archivo de ruta en el backend. Fuente: Propia

Dentro de la raíz del proyecto, también se han creado los siguientes archivos:

- **.env**: Archivo para definir variables de entorno que serán utilizadas en la aplicación.
- **.gitignore**: Archivo usado por Git para excluir archivos y directorios que no deben ser commiteados. Por ejemplo, la carpeta node_modules no tiene que estar en el repositorio, ya que contiene miles de archivos que no son necesarios transmitirlos.
- **dbConnection.js**: Archivo que gestiona la conexión a la base de datos. La Figura 53 enseña el archivo configurado para este proyecto.



```
JS dbConnection.js > ...
    ...
1   import { createPool } from "mariadb";
2   import dotenv from 'dotenv';
3   dotenv.config();
4
5   export const pool = createPool({
6       host: process.env.DB_HOST,
7       user: process.env.DB_USER,
8       password: process.env.DB_PASSWORD,
9       port: process.env.DB_PORT,
10      database: process.env.DB_DATABASE
11  })
```

Figura 53. Archivo dbConnection.js. Fuente: Propia

- **index.js**: Este archivo actúa como el punto de entrada y maneja la configuración general del backend de la aplicación. Es el responsable de iniciar el servidor y establecer las rutas iniciales de la API. La Figura 54 muestra el archivo configurado para el proyecto.

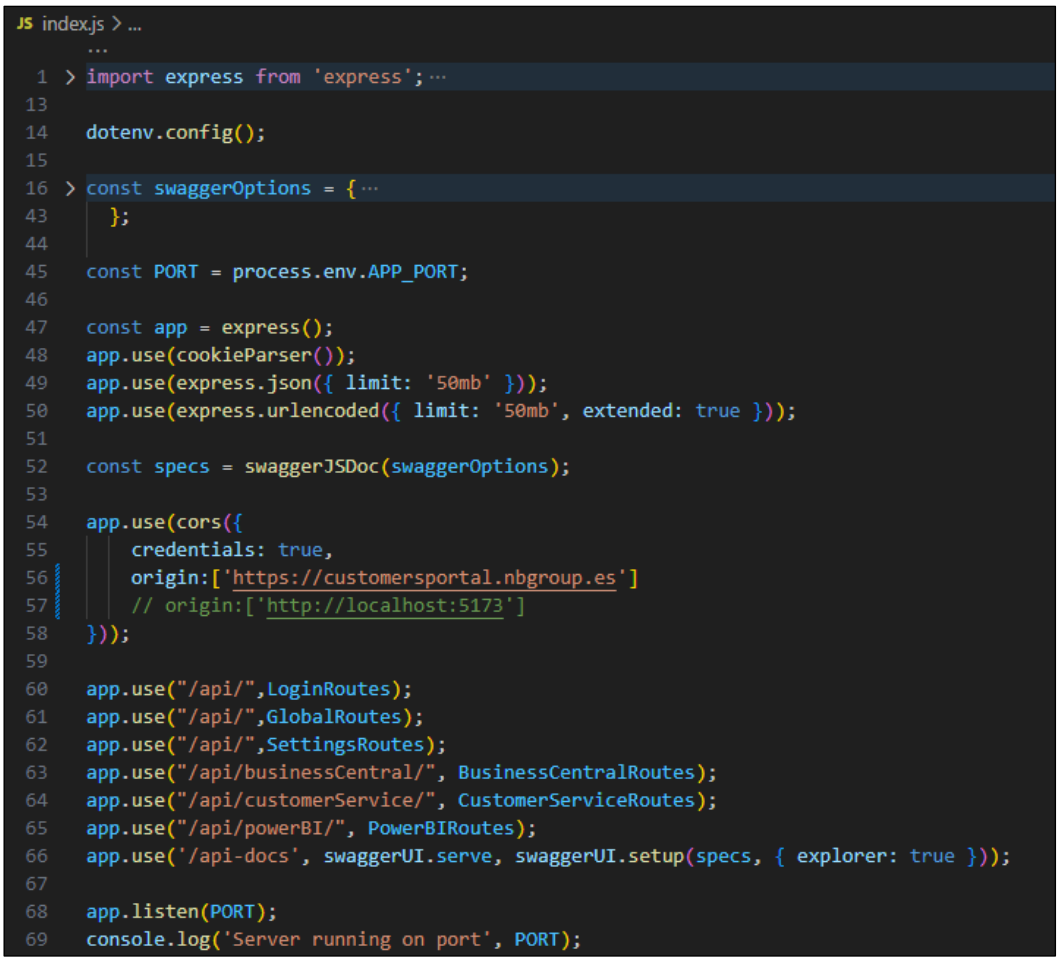


Figura 54. Archivo index.js. Fuente: Propia

- **package.json**: Archivo que contiene la configuración del proyecto y la lista de dependencias.

- **package-lock.json:** Archivo que asegura que las instalaciones de dependencias sean consistentes en diferentes instalaciones.

9.2.2 AL

La organización que se ha seguido para el proyecto de AL es la que marca la empresa. Cada directorio segmenta tipos de archivos de AL. En la Figura 55 se puede observar la estructura de carpetas y archivos de la raíz del directorio.

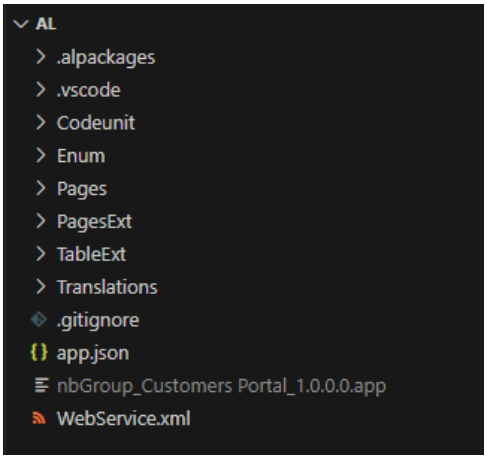


Figura 55. Estructura de archivos en la raíz - AL. Fuente: Propia

Cada carpeta o archivo tiene un propósito concreto:

- ***.alpackages:*** Carpeta que contiene los objetos necesarios para poder desarrollar y compilar el proyecto.

- **.vscode:** Carpeta donde se encuentran los ficheros de configuración del entorno para poder desarrollar. El archivo más popular es el launch.json, , donde se configura el entorno al que se quiere apuntar y se realizan los cambios.

- **Codeunit:** Esta carpeta almacena los archivos responsables de la lógica de negocio y los endpoints más complejos, como, por ejemplo, los que se encargan de obtener PDFs codificados en base64.

```al
codeunit 80700 "Customer Portal Management"
{
    [ServiceEnabled]
    0 references
    procedure getPDF(pType: Integer; pDocumentNo: Text) vResponse: Text
    begin
        case pType of
            Enum::"Document Type"::Invoices.AsInteger():
                begin
                    vResponse := getInvoiceDocumentPDF(pDocumentNo);
                end;
            Enum::"Document Type"::"Services Part".AsInteger():
                begin
                    vResponse := getServicesPartsPDF(pDocumentNo);
                end;
            Enum::"Document Type"::Shipments.AsInteger():
                begin
                    vResponse := getShipmentsPDF(pDocumentNo);
                end;
            Enum::"Document Type"::"Credit Memo".AsInteger():
                begin
                    vResponse := getCreditMemoPDF(pDocumentNo);
                end;
            Enum::"Document Type"::Contracts.AsInteger():
                begin
                    vResponse := getContractDocumentPDF(pDocumentNo);
                end;
        end;
    end;

    1 reference
    local procedure getInvoiceDocumentPDF(pDocumentNo: Text) vResponse: Text ···

    5 references
    local procedure getBase64Pdf(pReportId: Integer; var pRecordRef: RecordRef) vResponse: Text; ···

    4 references
    local procedure getReportID(pUsage: Enum "Report Selection Usage") reportID: Integer ···

    1 reference
    local procedure getServicesPartsPDF(pDocumentNo: Text) vResponse: Text; ···

    1 reference
    local procedure getShipmentsPDF(pDocumentNo: Text) vResponse: Text; ···

    1 reference
    local procedure getCreditMemoPDF(pDocumentNo: Text) vResponse: Text; ···
```

Figura 56. Codeunit "Customer Portal Management". Fuente: Propia

La Figura 56 es la codeunit “Customer Portal Management”, donde se ha creado el endpoint *getPDF*, ya que la función tiene la propiedad *ServiceEnabled*.

- **Enum:** Carpeta donde se encuentran las estructuras de datos que se han creado exclusivamente para el proyecto. La Figura 57 muestra el archivo “Document Type”.

- **Pages:** Carpeta más importante del proyecto, en esta se encuentran todos los endpoints para poder obtener la gestión administrativa. En Figura 58 se observan todos los archivos que hay dentro de la carpeta.

```al
enum 80700 "Document Type"
{
    Extensible = true;

    1 reference
    value(0; "Invoices")
    {
        Caption = 'Invoices';
    }
    1 reference
    value(1; "Services Part")
    {
        Caption = 'Services Part';
    }
    1 reference
    value(2; "Shipments")
    {
        Caption = 'Shipments';
    }
    1 reference
    value(3; "Credit Memo")
    {
        Caption = 'Credit Memo';
    }
    1 reference
    value(4; "Contracts")
    {
        Caption = 'Contracts';
    }
}
```

Figura 57. Enum "Document Type" - AL. Fuente: Propia

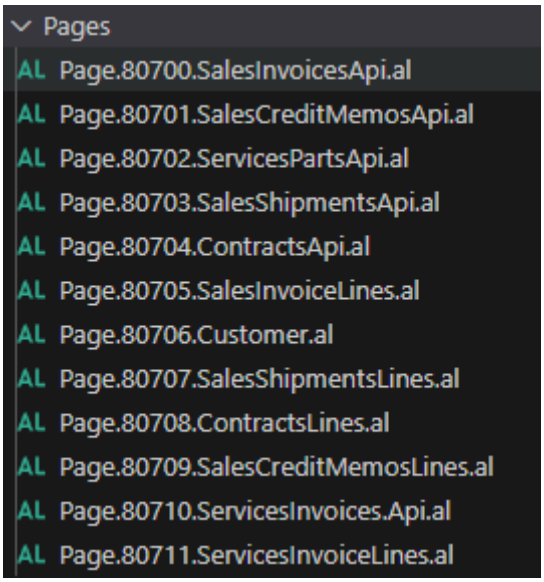


Figura 58. Archivos dentro de la carpeta Pages. Fuente: Propia

Los archivos que generan endpoints, como se enseña en la Figura 59, tienen en la cabecera varios atributos obligatorios a informar. Entre ellos, destacan *APIVersion*, *APIPublisher*, *APIGroup*, *EntityName*, que son los atributos que definen la ruta de la API. Por otro lado, el atributo "SourceTable", define a que tabla se apunta para obtener los datos. Finalmente, se decide si se permite la inserción, la edición y la eliminación de los datos a través del endpoint. En este caso son endpoints solo de lectura, por tanto los atributos *DeleteAllowed*, *ModifyAllowed* y *InsertAllowed* están en false. Por último, los *fields* son los campos que se exponen en el endpoint.

```
page 80700 "Sales Invoices Api"
{
    PageType = API;
    Caption = 'Sales Invoices Api';
    APIVersion = 'v2.0';
    APIPublisher = 'nbGroup';
    APIGroup = 'nbCustomersPortal';
    EntityName = 'SalesInvoices';
    EntitySetName = 'SalesInvoices';
    SourceTable = "Sales Invoice Header";
    DeleteAllowed = false;
    ModifyAllowed = false;
    InsertAllowed = false;

    layout
    {
        area(Content)
        {
            repeater(GroupName)
            {
                field(No; Rec."No.")
                {
                    ApplicationArea = All;
                }
                field(CustomerNo; Rec."Bill-to Customer No.")
                {
                    ApplicationArea = All;
                }
                field(Amount; Rec.Amount)
                {
                    ApplicationArea = All;
                }
```

Figura 59. Archivo que genera un endpoint - AL. Fuente: Propia

```
{
  "id": "0c999204-9dbb-4486-b280-77c42ba2daa8",
  "name": "Customers Portal",
  "publisher": "nbGroup",
  "version": "1.0.0.0",
  "brief": "",
  "description": "",
  "privacyStatement": "",
  "EULA": "",
  "help": "",
  "url": "",
  "logo": "",
  "screenshots": [],
  "platform": "1.0.0.0",
  "application": "24.0.0.0",
  "idRanges": [
    {
      "from": 80700,
      "to": 80799
    }
  ],
  "resourceExposurePolicy": {
    "allowDebugging": true,
    "allowDownloadingSource": true,
    "includeSourceInSymbolFile": true
  },
  "runtime": "13.0",
  "features": [
    "NoImplicitWith",
    "TranslationFile"
  ]
}
```

Figura 60. Archivo app.json - AL. Fuente: Propia

- **App.json:** Este archivo es el más importante del proyecto, ya que contiene toda la configuración necesaria. Destacan varios atributos clave, como el *id*, que proporciona una identificación única de la aplicación dentro de Business Central, y *application*, que especifica la versión de Business Central para la cual se está desarrollando la extensión. También es esencial tratar correctamente los *idRanges*, que son los rangos de ID de objetos permitidos para la publicación en Business Central. La duplicidad de *ids* dentro de estos rangos causará errores, impidiendo que la extensión se cargue en Business Central. La Figura 60 muestra el app.json del proyecto.

9.3 Integración de los servicios de Microsoft

Para integrar los servicios de Microsoft **[42]**, es necesario configurar el tenant de Azure de la empresa. Esto permitirá obtener los permisos necesarios para acceder a Azure, Business Central y Customer Service al realizar llamadas mediante web service.

El primer paso es iniciar sesión en el portal de Azure, https://portal.azure.com, utilizando una cuenta que tenga permisos de administrador para poder hacer cambios dentro del tenant. Una vez dentro de la plataforma, hay que crear registrar una aplicación para representar la solución o servicio que se quiere hacer. Este paso generará un id, este id será el que luego se necesitará para la autenticación en el Node.Js. En la Figura 61 se muestra la aplicación registrada para el proyecto.

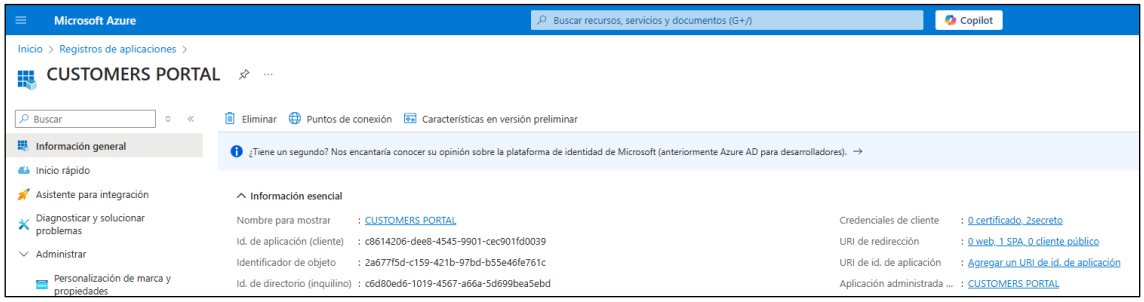


Figura 61. Aplicación registrada en Azure. Fuente: Propia

Después de registrar la aplicación, es necesario configurar los permisos para las URIs que están autorizadas para recibir el token de autenticación del sistema. En el caso del proyecto, como se muestra en la Figura 62, se ha configurado acceso tanto para localhost para realizar pruebas en el entorno local mientras se desarrolla, como para el dominio donde se aloja la aplicación. Esto garantiza que solo las URIs especificadas puedan recibir y manejar el token de autenticación.

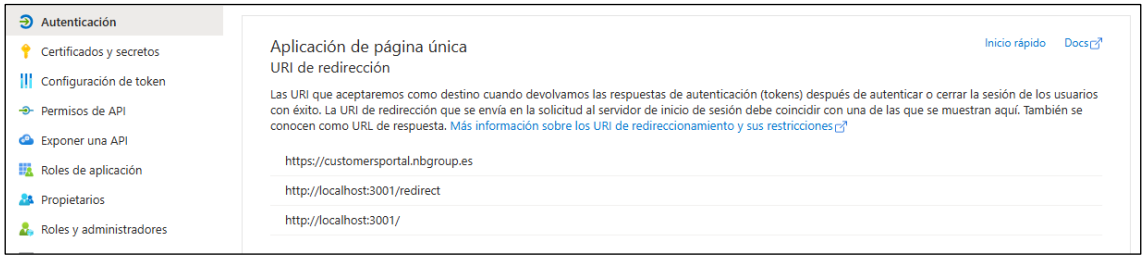


Figura 62. Autenticación de la aplicación en Azure. Fuente: Propia

El siguiente paso es generar una clave secreta, que es la que se utiliza para autenticar y generar el token de acceso. Esta clave secreta se envía en cada petición para solicitar el token. Si la clave está expirada o es incorrecta, Microsoft no emitirá el token. La clave es la columna “Valor”, tal y como se enseña ver en la Figura 63.

Figura 63. Clave secreta en Azure. Fuente: Propia

Por último, es necesario asignar permisos a las aplicaciones que se necesitan utilizar para que el token sea válido. Si el token se utiliza para alguna aplicación que no ha sido configurada previamente, se producirá un error en la llamada. Para el proyecto, como se detalla en Figura 64, se han configurado permisos en:

- **Business Central:** Lectura y escritura
- **Customer Service:** Lectura y escritura
- **Power Bi:** Lectura

Nombre de permisos/API	Tipo	Descripción
Dynamics 365 Business Central (6)		
AdminCenter.ReadWrite.All	Aplicación	Full access to Admin Center API
API.ReadWrite.All	Aplicación	Full access to web services API
app_access	Aplicación	Access according to the application's permissions in Dynamics 365 Business Central
Automation.ReadWrite.All	Aplicación	Full access to automation
Financials.ReadWrite.All	Delegada	Access Dynamics 365 Business Central as the signed-in user
user_impersonation	Delegada	Access as the signed-in user
Dynamics CRM (1)		
user_impersonation	Delegada	Access Common Data Service as organization users
Microsoft Graph (7)		
Contacts.Read	Delegada	Leer contactos de usuario
Contacts.Read	Aplicación	Read contacts in all mailboxes
Contacts.Read.Shared	Delegada	Leer los contactos compartidos y de usuario
Contacts.ReadWrite	Delegada	Tener acceso total a los contactos de usuario
Contacts.ReadWrite	Aplicación	Read and write contacts in all mailboxes
Contacts.ReadWrite.Shared	Delegada	Leer y escribir en calendarios compartidos y de usuario
User.Read	Delegada	Iniciar sesión y leer el perfil del usuario
Power BI Service (2)		
App.Read.All	Delegada	View all Power BI apps
Environment.Read.All	Delegada	Make API calls that require read permissions on all environment items

Figura 64. Permisos de aplicación concedidos en Azure. Fuente: Propia

9.4 Despliegue

Tal y como se ha explicado anteriormente en el punto 8.1 la aplicación se despliega en clouding.io. En concreto, este es un servidor Ubuntu Server 24.

En clouding.io, se tiene la capacidad de controlar los firewalls y configurar las reglas de red, lo que nos permite ajustar la seguridad y el acceso según las necesidades específicas de la empresa. Esto asegura que solo los tráficos de red autorizados puedan acceder a nuestros servicios, aumentando la seguridad de la aplicación y los datos alojados.

Para la administración del servidor, se utiliza SSH (Secure Shell) **[43]**, lo que garantiza una conexión segura y cifrada. Las conexiones SSH se realizan utilizando Putty **[44]**, una aplicación cliente para Windows que facilita el acceso remoto a sistemas Linux. La autenticación se realiza a través de claves privadas.

Para organizar los archivos y recursos necesarios para la aplicación, se crea un directorio específico dentro del servidor. Este directorio servirá como ubicación centralizada para almacenar todos los componentes de la **aplicación**, tanto el frontend como el backend.

```
root@cloudserver07:/# mkdir nbCustomersPortal
```

Figura 65. Inicialización del proyecto en el servidor. Fuente: Propia

- mkdir (make directory): Comando utilizado para crear un nuevo directorio en el sistema de archivos del servidor.
- nbCustomersPortal: Nombre del nuevo directorio. Este nombre indica que el directorio está destinado a contener los recursos del portal de clientes de la empresa.

Una vez creado el directorio nbCustomersPortal en el servidor, el siguiente paso es transferir los archivos necesarios. Esta tarea se realiza utilizando un cliente de FTP **[45]**, que permite gestionar y transferir archivos entre un ordenador local y un servidor remoto de forma segura, en este caso se utiliza la aplicación WINSCP **[46]**.

A continuación, como se observa en la Figura 66, se han subido los archivos del frontend de la aplicación al servidor.

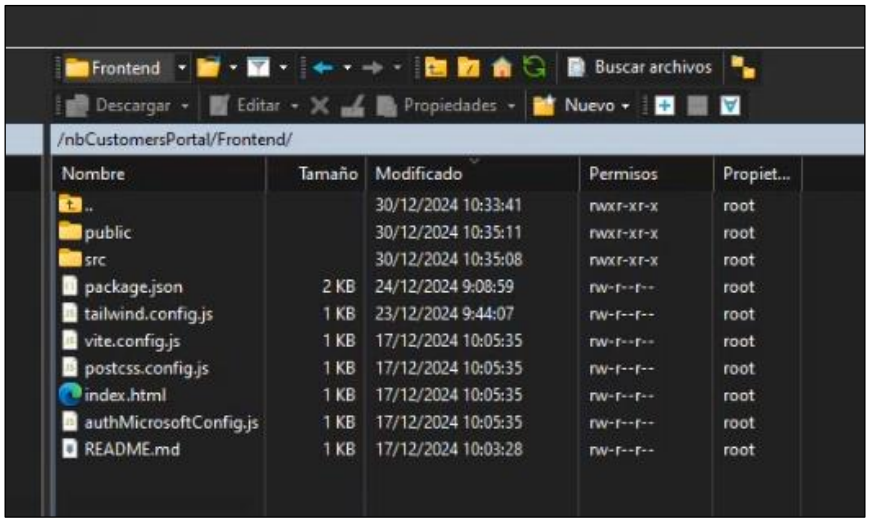


Figura 66. WINSCP - Frontend. Fuente: Propia

Una vez subidos los archivos del frontend, se suben los archivos del backend.

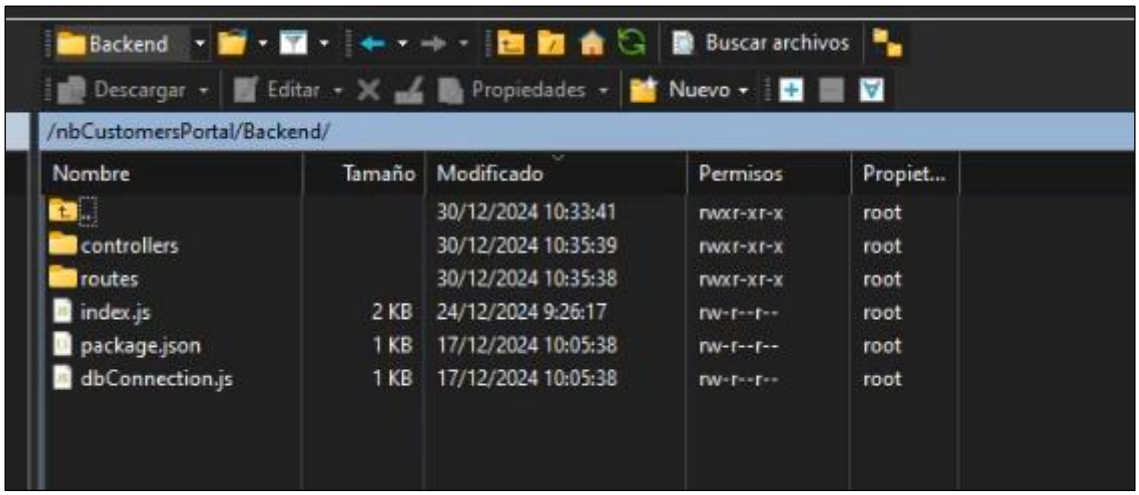


Figura 67. WINSCP - Backend. Fuente: Propia

Después de subir los archivos del frontend y backend al servidor, se realiza la instalación de las dependencias necesarias para que la aplicación funcione correctamente. Al utilizar React y Node.Js, esto se consigue mediante el comando **npm i** (*abreviatura de npm install)*, que descarga e instala paquetes desde el registro de npm basado en las especificaciones del archivo package.json.

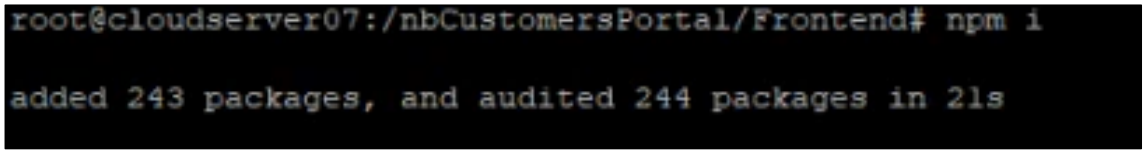


Figura 68. Instalación de paquetes - Frontend. Fuente: Propia

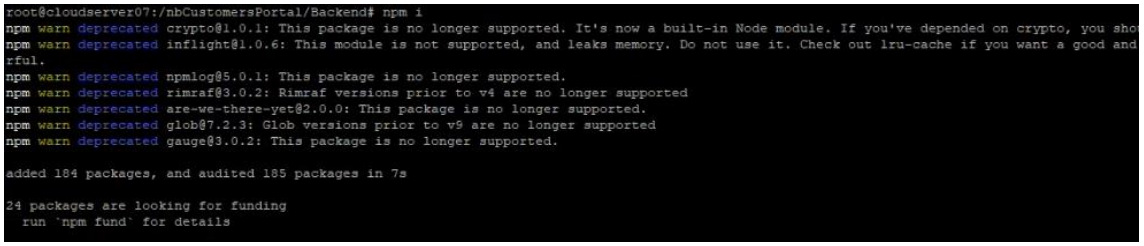


Figura 69. Instalación de paquetes - Backend. Fuente: Propia

Para configurar adecuadamente el entorno de la aplicación frontend y backend y asegurar que todos los parámetros operativos específicos estén establecidos, se edita el archivo de variables de entorno “**.env”** utilizando el editor de texto “nano”.

Editar el archivo “.env” permite personalizar cómo se comporta la aplicación en diferentes entornos (desarrollo, producción, etc), asegurando que se utilicen los parámetros correctos sin necesidad de cambiar el código base. Este archivo es esencial para manejar configuraciones que no deben ser expuestas públicamente o que necesitan ser fácilmente modificables sin despliegues adicionales.

Figura 70. Comando para editar el archivo .env - Frontend. Fuente: Propia

Figura 71. Comando para editar el archivo .env - Backend. Fuente: Propia

Para poder correr el frontend y el backend dentro del servidor se utiliza PM2 **[47]**. Este es un gestor de procesos que permite iniciar, detener, y monitorear aplicaciones Node.js de manera continua.

En la Figura 72 y Figura 73 se puede observar el listado de servicios de PM2 una vez inicializados el backend y el frontend.

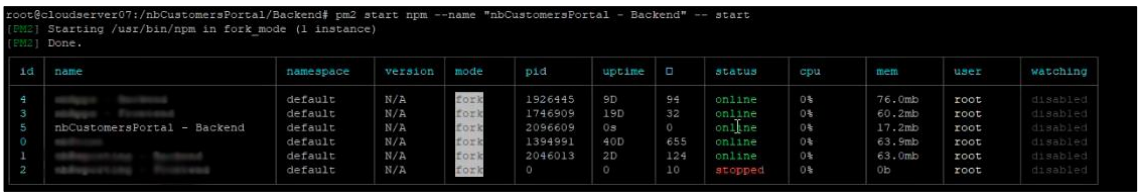


Figura 72. Servicios del PM2 al arrancar el Backend. Fuente: Propia

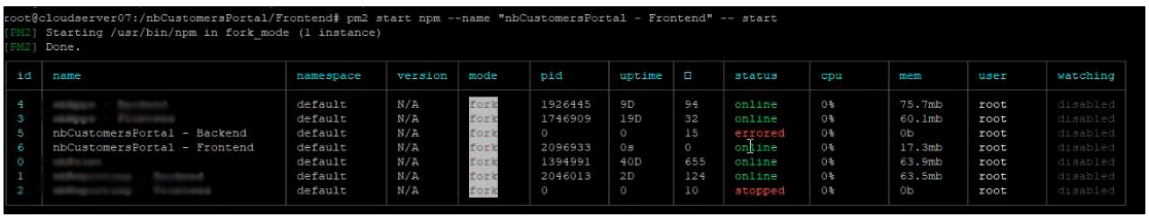


Figura 73. Servicios del PM2 al arrancar el frontend. Fuente: Propia

Para verificar que el backend del proyecto está activo y accesible se realiza una prueba de conectividad. Se usa el comando curl para enviar una petición HTTP al servidor local en el puerto 3008, que es el puerto en el que se ha alojado el backend de la aplicación, el cual está configurado para manejar las solicitudes del frontend de la aplicación.

Figura 74. Petición de prueba HTTP al backend. Fuente: Propia

Se ha añadido un script de inicio en el archivo package.json del directorio del frontend para facilitar el despliegue y la ejecución local de la aplicación. Este script está diseñado para iniciar la aplicación usando un servidor estático que sirve los archivos necesarios para el funcionamiento del frontend.

Detalles de la Configuración:

- serve: Es el comando utilizado para iniciar un servidor estático.
- -s dist: Indica que el servidor debe servir los archivos desde el directorio dist.
- -l 3020: Define el puerto 3020 como el puerto en el cual el servidor estará escuchando las solicitudes, lo cual es importante para asegurar que el puerto esté libre y no cause conflictos en el entorno del servidor.

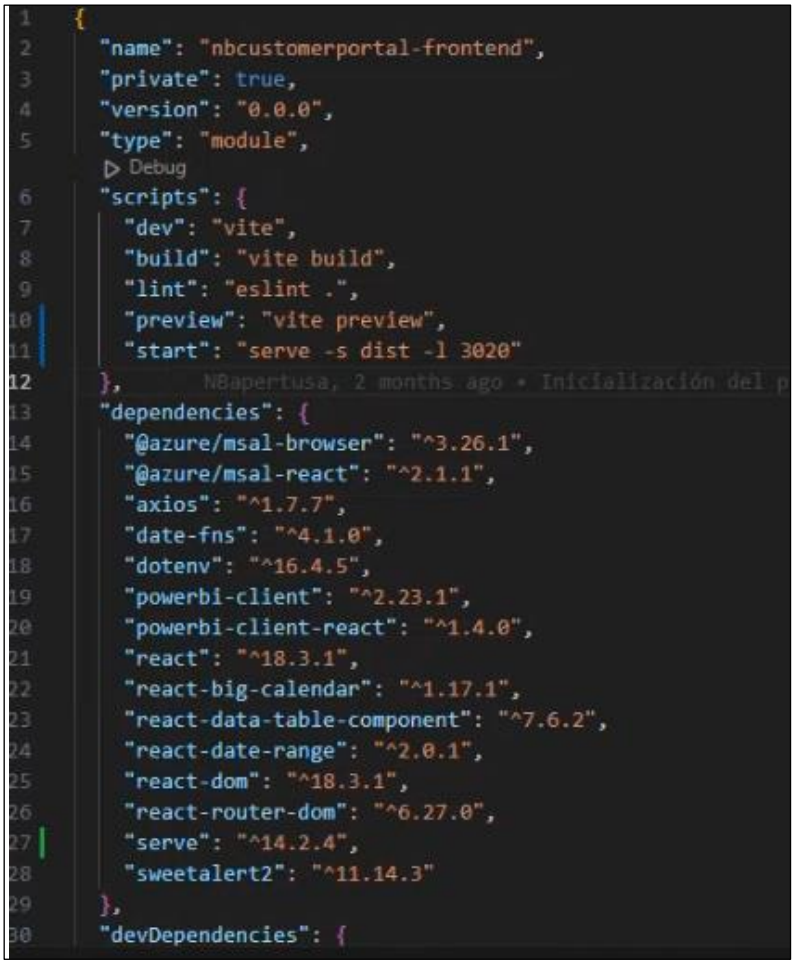


Figura 75. Archivo package.json del frontend. Fuente: Propia

Al acabar de configurar el package.json, se ejecuta el comando *npm run build* para compilar el frontend de la aplicación para un entorno de producción. Este comando inicia el proceso de construcción que optimiza y minimiza los archivos, preparándolos para un despliegue eficiente.

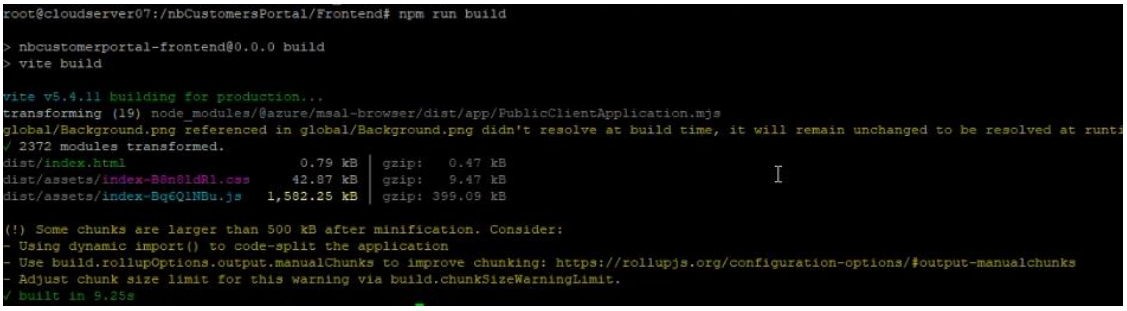


Figura 76. Hacer un build del frontend. Fuente: Propia

Tras compilar los archivos del frontend en el servidor, se realiza una prueba de conectividad para verificar que la página inicial está siendo servida correctamente. Para ello, se utiliza el comando curl, que permite hacer una solicitud al servidor local en el puerto especificado y obtener la estructura HTML de la página inicial, confirmando que los archivos están accesibles y se sirven correctamente.

Figura 77. Petición de prueba HTTP al frontend. Fuente: Propia

Para garantizar que tanto el frontend como el backend de la aplicación sean accesibles desde fuera del servidor, se ha configurado el firewall para permitir el tráfico a través de los puertos específicos utilizados por estos componentes.

La configuración del firewall se ha modificado a través de la propia plataforma web de clouding.io.

Figura 78. Configuración del firewall. Fuente: Propia

Se ha configurado un registro DNS de tipo A **[48]** para facilitar el acceso a la aplicación mediante un nombre de dominio. Esta configuración se realiza en la herramienta de administración de DNS que ofrece Cloudflare **[49]**.

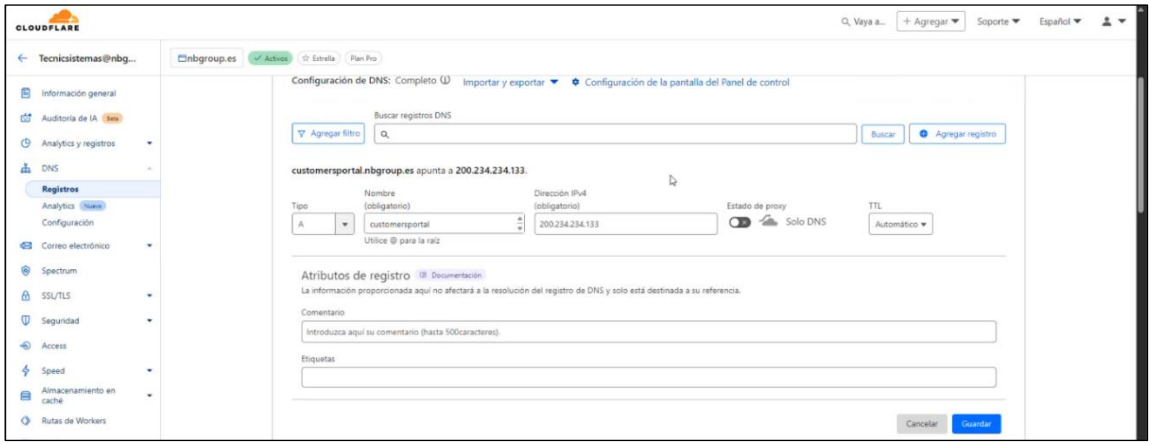


Figura 79. Configuración dominio en Cloudflare. Fuente: Propia

Una vez configurado el dominio, se duplica un archivo de configuración existente de Nginx **[50]** para adaptarlo a un nuevo dominio.

```
root@cloudserver07:/etc/nginx/sites-available# cp apps.nbgroup.es.conf customersPortal.nbgroup.es.conf
```

Figura 80. Copiar el archivo de configuración de Nginx. Fuente: Propia

Una vez clonado el archivo de configuración, ya está listo el Nginx para servir el frontend de la aplicación y redirigir las solicitudes de la API al backend correspondiente.

```
 GNU nano 6.2                                                customersPortal.nbgroup.es.conf
erver {
    server_name customersportal.nbgroup.es;
    root /nbCustomersPortal/Frontend/dist;

    location / {
        try_files $uri $uri/ /index.html;
    }

    location /api/ {
        proxy_pass http://localhost:3008;
        proxy_set_header Connection "";
        proxy_read_timeout 300s;
        proxy_connect_timeout 75s;
        proxy_buffer_size   128k;
        proxy_buffers   4 256k;
        proxy_busy_buffers_size   256k;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    location ~* \.(jpg|jpeg|png|gif|ico|css|js)$ {
        expires 1y;
        add_header Cache-Control "public, must-revalidate";
    }

erver {
    if ($host = customersportal.nbgroup.es) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name customersportal.nbgroup.es;
    return 301 https://$host$request_uri;
```

Figura 81. Archivo de configuración de NGINX. Fuente: Propia

Para activar la nueva configuración del sitio web del proyecto, en el servidor Nginx, se crea un enlace simbólico del archivo de configuración del sitio desde el directorio 'sites-available' al directorio 'sites-enabled'. Esto le indica a Nginx que cargue y aplique esta configuración cuando el servidor se inicie o se recargue.

Figura 82. Creación del enlace en Nginx. Fuente: Propia

Una vez creado el enlace, hay que reiniciar el servicio del Nginx para aplicar los cambios hechos en los archivos de configuración.

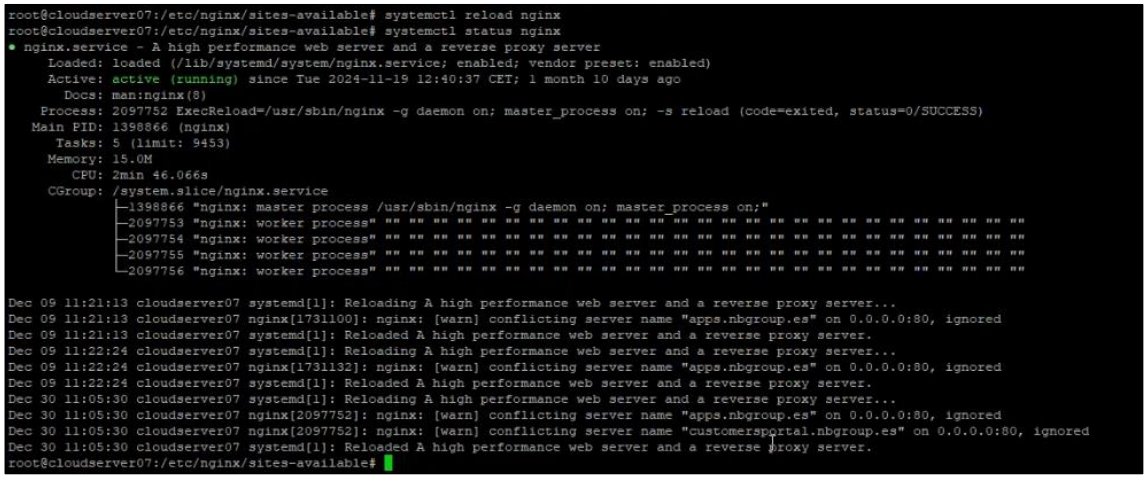


Figura 83. Reiniciar el servicio de Nginx. Fuente: Propia

Para configurar el certificado SSL/TLS **[51]** para el dominio customersportal.nbgroup.es se ha utilizado Certbot **[52]**. Esto asegura que todas las comunicaciones hacia y desde la aplicación sean cifradas.

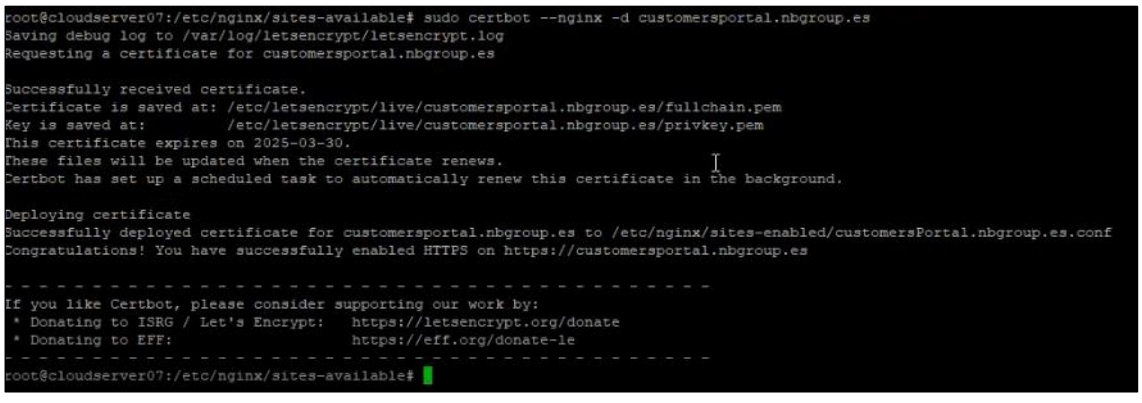


Figura 84. Obtener certificado SSL/TLS. Fuente: Propia

Al finalizar este último paso, ya se dispone de la aplicación desplegada en el dominio correspondiente. En la Figura 85 se puede observar que la aplicación ya está corriendo en **https://customersPortal.nbgroup.es** .

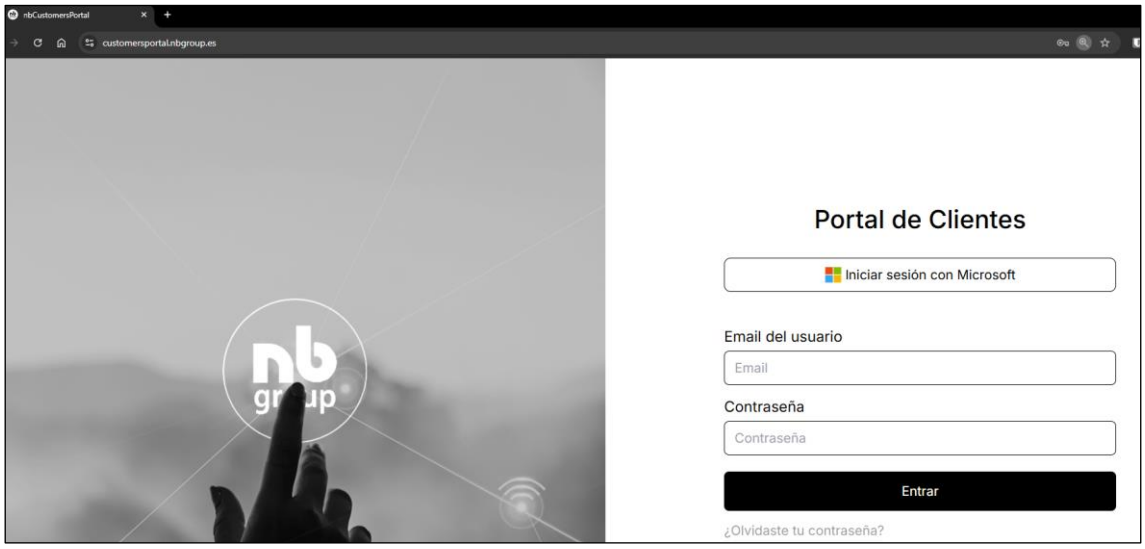


Figura 85. Aplicación desplegada. Fuente: Propia

10 Sostenibilidad

A lo largo de los años, la importancia de la sostenibilidad ha crecido significativamente en cualquier proyecto de software. Por esta razón, el grado de GEI pone un gran énfasis en sus asignaturas para concienciar a los estudiantes sobre el impacto que tienen estos proyectos a nivel ambiental, económico y social.

Para resaltar este aspecto y llevar a cabo una reflexión sobre la sostenibilidad, primero se ha revisado el informe de sostenibilidad de la asignatura GEI. Posteriormente, se ha completado la encuesta EDINSOST2-ODS, cuyo objetivo es recopilar información y datos sobre los conocimientos de sostenibilidad entre el alumnado de las universidades españolas.

10.1 Autoevaluación

En el informe y con la encuesta realizada, se destaca la conexión y el equilibrio que tiene que existir entre el impacto ambiental, económico y social. Si cualquiera de estos no está alineado con los otros, el proyecto carece de sostenibilidad y repercute a la sociedad de manera negativa.

A nivel ambiental, hay muchos aspectos de los cuales no tenía conocimiento previo. En este ámbito es importante considerar tanto el uso eficiente de los recursos como la reducción de residuos. En medida de lo posible, en proyectos como el que he realizado hay que buscar la manera de reutilizar el material hardware y tratar de consumir los menores recursos a nivel de software y hardware. Aun así, no tengo el conocimiento para definir si el proyecto es sostenible en la rama ambiental.

La parte económica es la más sencilla de entender y generalmente la primera en considerarse a la hora de desarrollar un proyecto. La mayor parte de la gente se suele centrar en este aspecto para poder tener un presupuesto detallado y preciso. Sin embargo, si se maximiza únicamente esta dimensión, sin tener en cuenta las otras ramas, el proyecto corre el riesgo de no ser sostenible.

Por último, la dimensión social es fundamental, ya que afecta directamente a las personas. En cualquier proyecto, es esencial tener en cuenta este aspecto, ya que las soluciones pueden influir en la vida diaria. En mi caso, el proyecto mejora la calidad de vida de los empleados y clientes, pero en otros casos, puede tener repercusiones negativas. Por ello, es crucial ser consciente de que cualquier aplicación o desarrollo puede cambiar el comportamiento y la forma de vivir de las personas, lo que requiere un enfoque responsable y ético.

10.2 Ambiental

- **¿Se ha estimado el impacto ambiental que tendrá la realización del proyecto?**

  El impacto medioambiental que tiene el proyecto es de una magnitud muy reducida. El principal impacto proviene del material hardware (portátil, pantallas, ratón y teclado) ya que ha sido comprado para el proyecto. Por otro lado, el uso de la aplicación no conlleva un elevado consumo de electricidad, dado que está alojada en un servidor en la nube proporcionado por un proveedor que optimiza

sus recursos para minimizar el impacto medioambiental. Por último, el autor se ha desplazado a la oficina en tren, lo que evita el uso del coche que podría tener un mayor impacto negativo en el medio ambiente.

- **¿Se ha planteado minimizar el impacto, por ejemplo, reutilizando recursos?**

  El único caso en el que podríamos reutilizar recursos es en el material de hardware. En el caso de este proyecto, lamentablemente por plazos de tiempo no se pudo realizar, pero la empresa cuenta con ordenadores averiados y pantallas parcialmente dañadas, cuyos componentes son reemplazados para darles una segunda vida. En futuros proyectos, se podría considerar el uso de este material reutilizado.

- **¿Cómo se resuelve actualmente el problema que se quiere abordar (estado del arte)? ¿En qué mejorará ambientalmente la solución propuesta a las existentes?**

  En el mercado no existen soluciones que aborden de manera integral la problemática específica de la empresa, aunque sí hay métodos de trabajo que resuelven parcialmente algunos de los desafíos. Estos métodos suelen depender de la impresión de documentos para organizar la información de manera rápida y eficiente. Sin embargo, con la solución propuesta, no solo se optimizan estos procesos, sino que además se elimina por completo la necesidad de impresión, reduciendo el uso de papel a 0. Este hecho contribuye significativamente a la sostenibilidad al reducir el consumo de recursos y minimizar el impacto ambiental.

10.3 Económica

- **¿Se ha estimado el coste de la realización del proyecto (recursos humanos y materiales)?**

  Se ha realizado una estimación de costes, teniendo en cuenta los recursos humanos, recursos materiales y los posibles imprevistos que afectan al coste del proyecto. Una vez desglosada y analizada cada sección, se ha creado un presupuesto final que representa la suma de todos los costes del proyecto de manera global.

- **¿Cómo se resuelve actualmente el problema que se quiere abordar (estado del arte)? ¿En qué mejorará económicamente la solución propuesta a las existentes?**

  Dado que no existen soluciones alternativas en el mercado, no es posible realizar una comparación directa con competidores. Sin embargo, desde la perspectiva de costes, se puede analizar el ahorro que la empresa obtendrá al centralizar toda la información en un único sistema. Esto permitirá que los clientes accedan directamente a sus facturas o movimientos sin necesidad de realizar llamadas

para solicitar dicha información, lo que reducirá tanto el tiempo invertido en gestión como los costes asociados al soporte al cliente.

10.4 Social

- **¿Qué crees que te va a aportar a nivel personal la realización de este proyecto?**

  Abordar un proyecto de esta magnitud me va a hacer crecer en diferentes aspectos y roles dentro de un proyecto. En primer lugar, como gestor y planificador del proyecto, me va a obligar a ser más responsable y metódico. Como desarrollador, me impulsará a mantener un mayor orden en el código y en la gestión de los repositorios. Además, también fortaleceré la comunicación en las reuniones con el poniente y el comité de dirección.

- **¿Cómo se resuelve actualmente el problema que se quiere abordar (estado del arte)? ¿En qué mejorará socialmente (calidad de vida) la solución propuesta a las existentes?**

  Actualmente, el problema que se busca abordar se resuelve de manera manual, lo que implica una gestión poco eficiente y que consume tiempo y recursos, tanto para los empleados como para los clientes. No existen soluciones en el mercado que resuelvan esta problemática.

  La solución propuesta, mejorará significativamente la calidad de vida de los empleados al reducir el tiempo y el estrés asociados a la búsqueda y gestión manual de la información. Desde el punto de vista del cliente, la solución también representa una mejora social, ya que le proporcionará un acceso fácil y rápido a toda la información relacionada, sin la necesidad de contactar directamente a la empresa. Esto incrementará la satisfacción del cliente y optimizará las interacciones entre ambos.

- **¿Existe una necesidad real del proyecto?**

  Sí que existe una necesidad real de realizar el proyecto. En la era de transformación digital en la que nos encontramos, algunos procesos dentro de la empresa siguen siendo muy manuales y rudimentarios, lo que consume tiempo y recursos que podrían ser aprovechados en otras tareas más productivas. La automatización propuesta no solo mejorará la eficiencia operativa, sino que también incrementará la calidad de vida de los empleados, al reducir el estrés generado cada vez que un cliente solicite información. Además, la experiencia del cliente también se verá beneficiada, ya que tendrá acceso a un portal web donde podrá consultar de manera autónoma toda la información relevante.

11 Testing

Para garantizar el buen funcionamiento de la aplicación antes de desplegarla en producción, se han implementado varios procesos de pruebas. Estos procesos se dividen en dos:

Experiencia de usuarios de la empresa: Se han llevado a cabo pruebas de usabilidad con usuarios reales dentro de la empresa para evaluar la interfaz del proyecto. Al incluir personas con más experiencia, se han podido identificar y corregir pequeños bugs o han podido sugerir mejoras a nivel visual.

Postman: Para validar la estabilidad y el rendimiento del backend, se han utilizado pruebas y peticiones gracias a la herramienta Postman. Estas pruebas han permitido simular solicitudes y respuestas de la API para poder verificar casos extremos o peticiones que no se habían contemplado en un inicio.

Para realizar estas pruebas se ha necesitado configurar un entorno exclusivo dentro de Postman. Como se observa en la Figura 86, se han definido variables que se repetían en las diferentes llamadas, tanto para backend propio, como para los servicios externos.

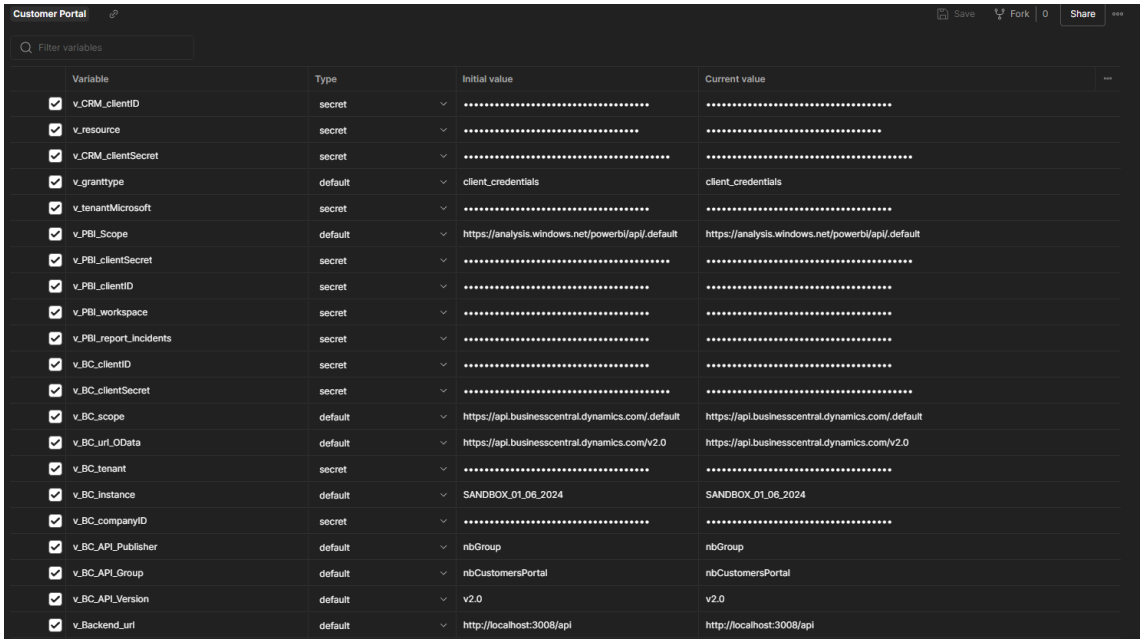


Figura 86. Configuración del entorno en Postman. Fuente: Propia

Una vez configuradas las variables necesarias para realizar las peticiones, se han creado y configurado las colecciones con las respectivas peticiones.

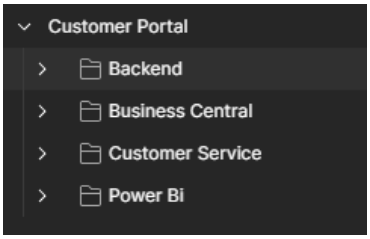


Figura 87. Organización de carpetas en Postman.
Fuente: Propia

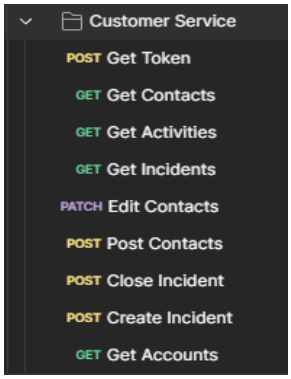


Figura 88. Peticiones a la API de Customer Service. Fuente: Propia

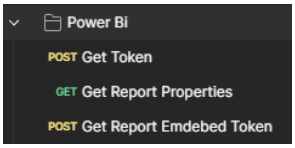


Figura 91. Peticiones a la API de Power BI. Fuente: Propia

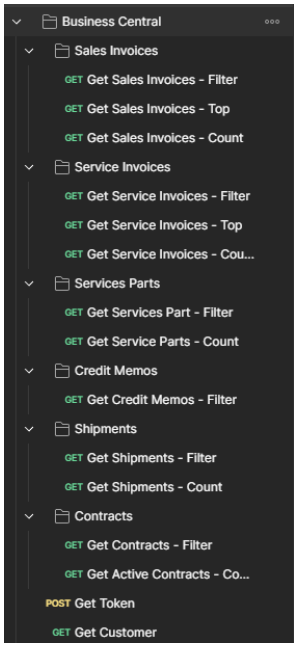


Figura 90. Peticiones a la API de Business Central. Fuente: Propia

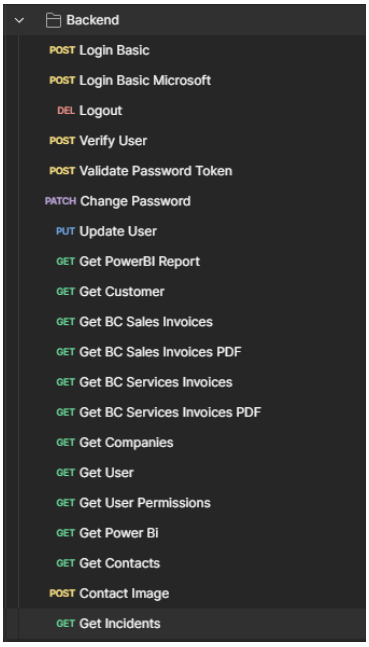


Figura 89. Peticiones a la API del Backend. Fuente: Propia

Cada petición dentro de la herramienta presenta una interfaz similar a la ilustrada en la Figura 92, donde se proporciona un campo para introducir la URL a la cual se desea realizar la petición. Además, en caso de que sea necesario, se pueden añadir parámetros al encabezado de la petición y un cuerpo de mensaje para adaptarse a los requisitos específicos de la petición. Una vez que se ha configurado todo lo necesario, se procede a ejecutar el botón de enviar y esto generará una respuesta a esa petición enviada, similar a la de la Figura 93.

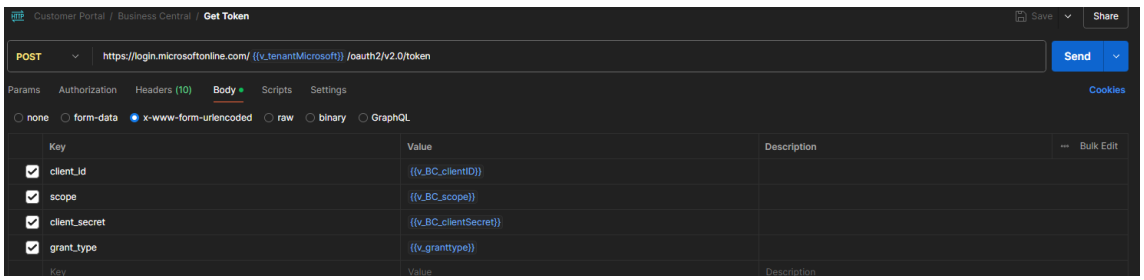


Figura 92. Pantalla para realizar las solicitudes en Postman. Fuente: Propia

Figura 93. Ejemplo de una respuesta a una petición. Fuente: Propia

12 Planificación real

El proyecto ha avanzado según lo contemplado en la planificación inicial. Aunque hubo una semana en la que no pude avanzar nada debido a temas personales, esto no afectó la planificación inicial, ya que anticipé esta pausa y dediqué más horas en las semanas previas. Sin embargo, en la planificación inicial no se especifica claramente la metodología utilizada. La fase de desarrollo del proyecto se organizó en varios sprints, cada sprint tiene la duración inicial de dos semanas.

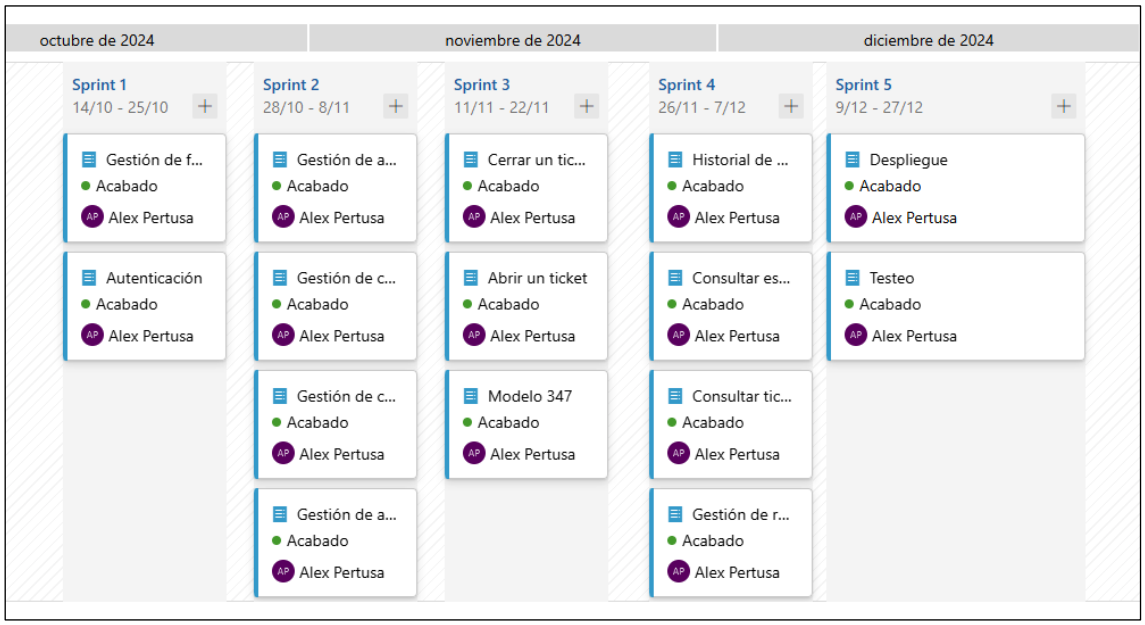


Figura 94. Resumen de los sprints en el calendario. Fuente: Propia

- **Sprint 1 (14/10/2024 – 25/10/2024):** Es la fase inicial del proyecto donde inicialicé todas las tecnologías y las dejé funcionando entre ellas. Fue un sprint en el que las horas previstas con las reales cuadraron. El único punto en el que me encallé fue en la configuración de la autenticación con Microsoft y la habilitación de permisos para obtener datos de los endpoints de Business Central, ya que no hay mucha documentación clara y no tenía los permisos adecuados a nivel de empresa para hacer cambios en la cuenta de Microsoft de la empresa.

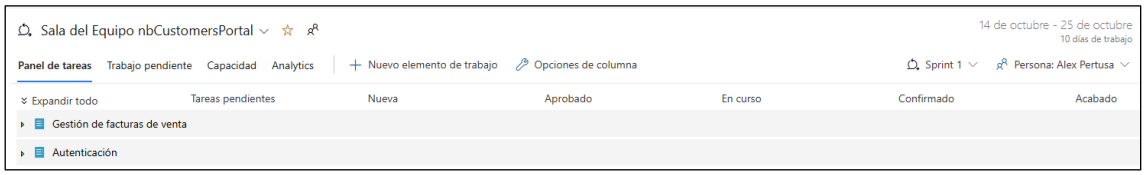


Figura 95. Sprint 1. Fuente: Propia

- **Sprint 2 (28/10/2024 – 8/11/2024):** Este sprint, a nivel de backend no supuso ningún problema, fue ir creando poco a poco los endpoints y enlazarlos en el Node.Js. El problema vino a nivel de frontend, ya que replicar los mockups que me había pasado el diseñador web empezaba a ser una tarea bastante compleja. En términos de estimación de horas, sobrestimé algunas horas dedicadas al backend, lo cual equilibró las horas adicionales que requerí para el frontend. En total, la diferencia fue de aproximadamente 4 o 5 horas, no más.

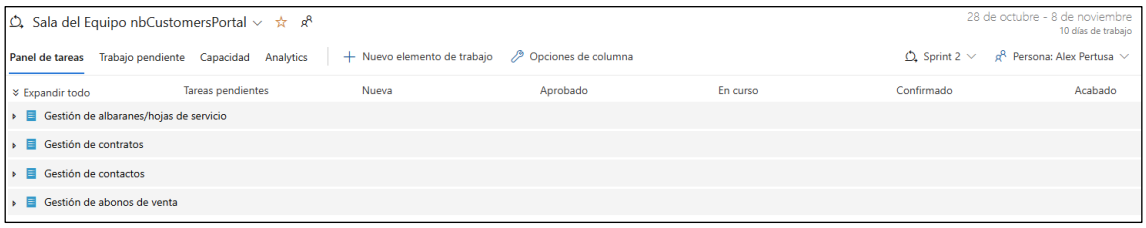


Figura 96. Sprint 2. Fuente: Propia

- **Sprint 3 (11/11/2024 – 22/11/2024):** Durante este sprint, me enfrenté la problemática detallada en el capítulo 8.3.2. Fue un momento en el que necesitaba más mockups por parte del diseñador y al estar solicitado por otros proyectos, los tiempos de respuestas eran muy lentos y yo necesitaba agilidad para poder cumplir con los plazos del proyecto. Por lo tanto, empecé a usar la inteligencia artificial para retocar mockups realizados anteriormente por el diseñador. No tuve ninguna desviación a nivel de horas ya que esta parte la realicé al principio del sprint y no me afectó al desarrollo de ninguna tarea.

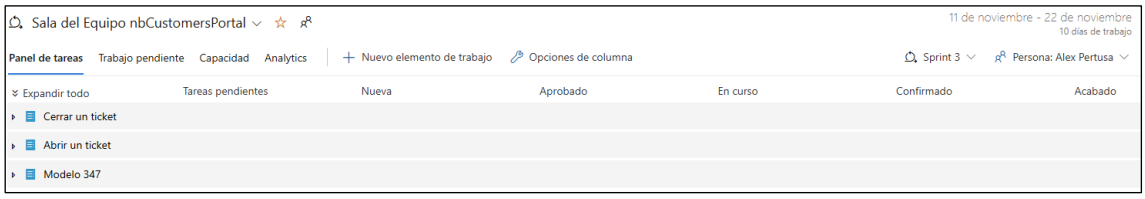


Figura 97. Sprint 3. Fuente: Propia

- **Sprint 4 (26/11/2024 – 7/11/2024):** En este sprint no me encontré ningún contratiempo y ha sido el sprint más tranquilo que he tenido.

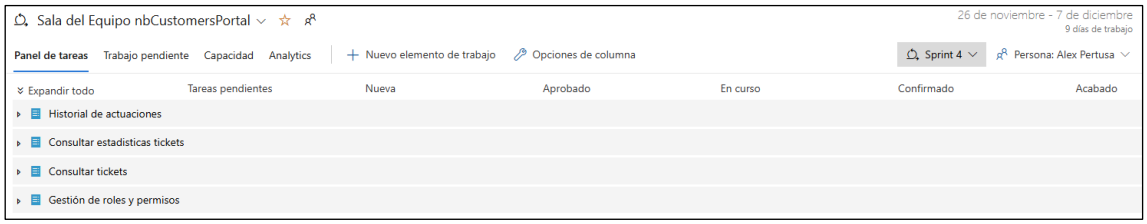


Figura 98. Sprint 4. Fuente: Propia

- **Sprint 5 (9/12/2024 – 27/12/2024):** Este ha sido el sprint más caótico de todos. En el testeo completo de la aplicación por parte de compañeros surgieron varios bugs, como era de esperar, los cuales fui solucionando. Una vez resuelto este tema, desplegué la aplicación en clouding.io y volvieron a salir varios bugs que tuve que ir mirando y arreglando sobre la marcha. Este sprint inicialmente tenía que durar dos semanas, pero debido a motivos personales tuve que parar una semana entera sin poder avanzar. No obstante, esto no afectó a la planificación del proyecto, ya que en esta fase no había que desarrollar ninguna funcionalidad nueva.

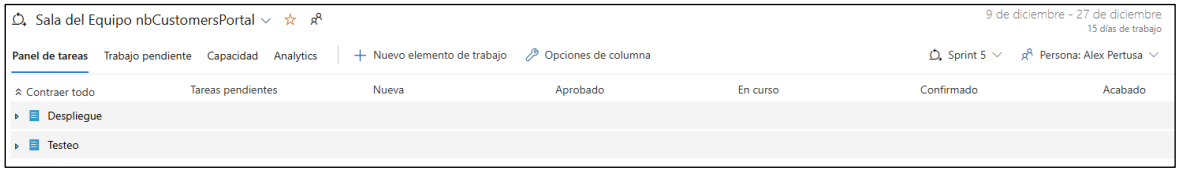


Figura 99. Sprint 5. Fuente: Propia

13 Conclusiones y futuro trabajo

Esta sección indica la conclusión del proyecto, donde se lleva a cabo un análisis final sobre el desarrollo y los logros alcanzados. Se revisa si los objetivos planteados al inicio se han logrado satisfactoriamente. Asimismo, se examina el avance en las habilidades técnicas adquiridas a lo largo del Grado en Ingeniería de Software, evaluando la contribución de las diferentes materias de la especialidad al proyecto. Finalmente, se identifican áreas para mejorar y se esbozan futuros enfoques para continuar desarrollando la aplicación.

13.1 Conclusión del proyecto

El proyecto ha logrado satisfactoriamente cumplir con los objetivos inicialmente propuestos. Se ha desarrollado una plataforma web que facilita al cliente el acceso y la revisión de toda la información relevante relacionada con la empresa. Tras varias reuniones con el comité de la empresa, se lograron especificar y satisfacer tanto los requisitos funcionales como los no funcionales, asegurando que todos los aspectos del proyecto cumplieran con las expectativas establecidas.

Además, se ha implementado con éxito la metodología Scrum para la gestión del desarrollo del proyecto. Esta metodología ha permitido una organización efectiva a través de la incorporación de sprints, los cuales estaban compuestos por historias de usuario y tareas bien definidas. Este enfoque ha facilitado una implementación ágil y adaptativa, permitiendo cambios en tiempo real.

Sin embargo, existe un desafío significativo que impide la utilización inmediata de la aplicación. Actualmente, la plataforma contiene datos y comentarios del personal de la empresa que no son apropiados para ser vistos por los clientes. Esto incluye información en las hojas de servicio y en otros documentos que podrían ser sensibles o no pertinentes para la vista del cliente. Por lo tanto, se ha decidido que la aplicación se implantará gradualmente entre los clientes, iniciando solo después de realizar una revisión exhaustiva en cada caso específico para asegurar que toda información sensible sea adecuadamente gestionada o removida.

Este desafío no se había previsto al inicio, lo que destaca un riesgo que surgió durante el desarrollo del proyecto. La obligación de revisar y posiblemente alterar los datos antes de hacerlos accesibles a los clientes, resalta la importancia de identificar los riesgos al inicio del proyecto.

13.2 Conclusión personal

A nivel personal, estoy contento y satisfecho con el trabajo realizado. Aunque me hubiese gustado poder dar acceso ya a los clientes a la aplicación, la presencia de datos e información sensible hace que esto no sea posible de momento, y esto ya se escapa de mi control directo.

Por otro lado, este proyecto me ha servido para poder realizar de manera personal todas las etapas de un proyecto. Desde el análisis de requisitos y la especificación, hasta el desarrollo, donde tuve la libertad de aplicar mis conocimientos adquiridos durante mi carrera y llevar a cabo la implementación a mi manera. En la fase final del proyecto, me encargué del análisis del cierre, destacando los logros obtenidos y los aspectos a mejorar.

El hecho de poder participar en reuniones con los directores de la empresa, ha sido una experiencia muy enriquecedora, ya que interactuar con personas con mucha más experiencia en el sector empresarial ha ampliado mi perspectiva y profundizado mi comprensión del mundo empresarial, aportando un gran valor a mi desarrollo profesional.

No obstante, el proyecto también me ha hecho ver aspectos en los que puedo mejorar tanto a nivel personal como técnico. Este ha sido mi primer proyecto dentro de un entorno empresarial, tras dejar el ámbito académico, donde los errores no tenían repercusiones reales ni causaban pérdidas económicas a nadie.

Por último, este proyecto me ha permitido descubrir el mundo de la consultoría de negocios, específicamente en los sistemas ERP y CRM, un ámbito que me ha interesado mucho y se puede complementar con los conocimientos adquiridos en la carrera.

13.3 Competencias técnicas

CES1.2: Dar solución a problemas de integración en función de las estrategias, de los estándares y de las tecnologías disponibles. [En profundidad]

Dado el problema que tenía la empresa inicialmente, esta competencia es la que más he trabajado, ya que los datos a recoger e interactuar estaban en diferentes softwares. Los desafíos incluían la restricción de usar exclusivamente MariaDB y la necesidad de elegir entre Node.js o PHP para el backend, debido a requisitos específicos de la empresa. Esto me llevó a desarrollar una solución con las tecnologías disponibles por la empresa: una plataforma en Node.Js y React que integra Business Central, Dynamics 365 Customer Service y Azure, junto con mi propia base de datos en MariaDB.

CES1.3: Identificar, evaluar y gestionar los riesgos potenciales asociados a la construcción de software que pudiesen presentarse. [Un poco]

Al ser un proyecto muy específico, tuve que realizar un análisis de los riesgos que conllevaba realizar esta aplicación, juntamente con las soluciones a esos posibles riesgos.

CES1.4: Desarrollar, mantener y evaluar servicios y aplicaciones distribuidas con soporte de red. [Un poco]

He trabajado en una aplicación web que se comunica con un servidor en la nube y cualquier cliente que tenga permisos puede acceder.

CES1.7: Controlar la calidad y diseñar pruebas en la producción de software. [Un poco]

Al no tener una restricción a la hora de realizar pruebas por parte de la empresa, he podido realizar pruebas con otros usuarios de la empresa y con la herramienta Postman, tal y como se explica en el capítulo anterior.

CES2.1: Definir y gestionar los requisitos de un sistema software. [En profundidad]

Es otro de los puntos que más he trabajado desde el inicio del proyecto. Todos los requisitos funcionales y no funcionales están explicados en el capítulo *Especificación de requisitos.*

En una primera reunión con los directores de la empresa presenté la idea del proyecto y en pocas semanas después ya tenía un esbozo con las ideas que querían proyectar ellos juntamente con las mías, para acabar realizando la especificación completa de requisitos.

Para los requisitos no funcionales he utilizado la plantilla de Volere y para los requisitos funcionales los he descrito de manera extensa utilizando historias de usuario.

CES2.2: Diseñar soluciones apropiadas en uno o más dominios de aplicación, utilizando métodos de ingeniería del software que integren aspectos éticos, sociales, legales y económicos. [Un poco]

Una vez realizada la especificación de requisitos, tuve que investigar sobre las leyes vigentes a nivel de protección de datos para poder dar acceso a los usuarios cumpliendo las legislaciones correspondientes.

13.4 Integración de conocimientos

A lo largo de la carrera, diversas asignaturas han contribuido, en mayor o menor medida, a la realización de este trabajo de final de carrera. No obstante, al enmarcar el proyecto dentro de la especialidad de software y enfocarlo en el desarrollo de una solución web, destacan especialmente aquellas asignaturas cuyos conocimientos han resultado más relevantes y aplicables en este contexto:

- **PES (Proyecto de especialidad de software):** Esta asignatura prepara al alumno como si fuera un trabajo de final de grado en grupos. Es una simulación a un proyecto real dónde se parte desde cero con una idea y hay que llevarla a cabo. Entre los conocimientos adquiridos, en mi caso, empecé a trabajar con NodeJS como backend.

- **GPS (Gestión de Proyectos Software):** Esta asignatura aborda las diferentes metodologías y enfoques de trabajo utilizados en la realización de proyectos de software. Entre los conocimientos adquiridos destaca la metodología Scrum, que se ha adoptado como la forma de trabajo principal para llevar a cabo este proyecto.

- **ER (Ingeniería de Requisitos):** En esta asignatura se aborda el análisis de proyectos de software desde una perspectiva menos técnica y más orientada a la gestión, ideal para un gestor de proyectos que interactúa directamente con clientes para satisfacer sus necesidades. Se pone especial énfasis en la habilidad para comprender a fondo los requisitos del cliente, una competencia clave que permite desarrollar soluciones personalizadas y efectivas.

- **AS (Arquitectura del Software):** En esta asignatura se aprende a como estructurar un proyecto de software desde la parte de la arquitectura, un aspecto clave para poder desarrollar este proyecto. De los conocimientos adquiridos, se puede destacar el uso de patrones de diseño para tener una aplicación escalable.

- **ASW (Aplicaciones y Servicios Web):** En esta asignatura se enseña cómo funcionan las aplicaciones web de hoy en día, fomentando el uso de API REST. En sí, todos los conocimientos sobre API y servicios web los utilizó en este proyecto.

13.5 Futuras funcionalidades

A lo largo del proyecto, tanto a mi como a las personas responsables de la empresa se nos han ocurrido nuevas ideas para mejorar la experiencia con el cliente y dar un salto más grande a nivel empresarial.

- **Integración con Docusign:** Con la integración de DocuSign, la plataforma evolucionará de ser un simple visualizador de contratos para revisar precios y servicios contratados, a una herramienta que facilita la firma automática de dichos contratos. Esto permitirá que las firmas de los responsables de los contratos se recojan y almacenen directamente en la plataforma, automatizando el proceso de aprobación y ejecución contractual y mejorando significativamente la eficiencia operativa. Esta funcionalidad no solo agiliza las transacciones, sino que también asegura la autenticidad y el cumplimiento de los acuerdos establecidos.

- **Responsive:** Tener la aplicación responsive asegura que esta se adapte a cualquier tamaño de pantalla, incluidos los dispositivos móviles. Esto mejora significativamente la accesibilidad y la experiencia del usuario, permitiendo que los usuarios interactúen con la plataforma cómodamente desde cualquier dispositivo y lugar.

- **Multiidioma:** Al tener clientes o contactos que hablan exclusivamente inglés y estando ubicados en Cataluña, la funcionalidad multiidioma de la aplicación garantiza que estos usuarios puedan acceder y utilizar la plataforma tanto en inglés como en catalán. Además, esta característica está diseñada para adaptarse y expandirse a nuevos idiomas según sea necesario, lo que permitirá atender a clientes de otras regiones en el futuro.

- **Permisos avanzados:** Actualmente, la aplicación cuenta actualmente con una separación de permisos de visualización entre las áreas de administración y soporte. En el futuro, se quiere aumentar la granularidad de estos permisos, implementando un sistema que permita controlar el acceso a nivel de página individual. Esta mejora en la gestión de permisos facilitará una mayor personalización, adaptándose a las necesidades específicas de cada usuario dentro de la plataforma. Con esto se consigue que los diferentes roles de cada empresa vean únicamente las pantallas específicas a su rol.

14 Referencias

[1] Qué es un Software erp, tipos y ejemplos – Wolters Kluwer. https://www.wolterskluwer.com/es-es/expert-insights/que-es-un-software-erp-tipos-y-ejemplos. Consultado el 22 de septiembre de 2024.

[2] Soluciones de negocio para la empresa – nbGroup. https://www.nbgroup.es/. Consultado el 22 de septiembre de 2024.

[3] Qué es un CRM, para qué sirve y usos comunes – David Gonzalez. https://davizgonzalez.com/blog/que-es-un-crm/. Consultado el 23 de septiembre de 2024.

[4] Microsoft Dynamics 365 Business Central – Goom Spain. https://www.microsoftdynamics365.com/business-central/. Consultado el 23 de septiembre de 2024.

[5] Microsoft Dynamics 365 Customer Service – nunsys. https://www.nunsys.com/dynamics-365-customer-service/. Consultado el 23 de septiembre de 2024.

[6] Modelo 347: qué es y cómo presentarlo – Wolters Kluwer. https://www.wolterskluwer.com/es-es/expert-insights/para-que-sirve-el-modelo-347-y-quien-debe-presentarlo. Consultado el 24 de septiembre de 2024.

[7] Dynamics 365 Connector – Zendesk. https://www.zendesk.es/marketplace/apps/support/240704/dynamics-365-connector/. Consultado el 24 de septiembre de 2024.

[8] Portal Clientes para Microsoft Dynamics NAV, Business Central & AX. https://www.grvppe.es/es/portal-clientes-dynamics-connect. Consultado el 24 de septiembre de 2024.

[9] Cómo funciona la Metodología Scrum: Qué es y cómo utilizarla – iebs. https://www.iebschool.com/blog/metodologia-scrum-agile-scrum/. Consultado el 25 de septiembre de 2024.

[10] Flujo de trabajo de Gitflow – Atlassian. https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow. Consultado el 25 de septiembre de 2024.

[11] Microsoft 365 – Microsoft. https://www.microsoft.com/es-es/microsoft-365. Consultado el 25 de septiembre de 2024.

[12] GanttProject – GanttProject. https://www.ganttproject.biz/. Consultado el 25 de septiembre de 2024.

[13] Draw.io – JGraph Ltd. https://www.drawio.com/.Consultado el 25 de septiembre de 2024.

[14] DBeaver – DBeaver. https://dbeaver.io/. Consultado el 25 de septiembre de 2024.

[15] Azure devops – Microsoft. https://azure.microsoft.com/es-es/products/devops. Consultado el 25 de septiembre de 2024

[16] Infraestructura Cloud a tu servicio – Cloudi NextGen. https://clouding.io/. Consultado el 25 de septiembre de 2024.

[17] Visual Studio Code. https://code.visualstudio.com/. Consultado el 25 de septiembre de 2024.

[18] Postman. https://www.postman.com/. Consultado el 25 de septiembre de 2024.

[19] ¿Cuántos días laborables en el año 2024?. https://www.dias-laborables.es/cuantos_dias_laborables_en_ano_2024_Catalu%C3%B1a.htm. Consultado el 7 de octubre de 2024.

[20] Volere Requirements Specification Template - Volere https://www.volere.org/templates/volere-requirements-specification-template/. Consultado el 10 de octubre de 2024.

[21] ¿Qué es HTTPS? – CLOUDFLARE. https://www.cloudflare.com/es-es/learning/ssl/what-is-https/. Consultado el 4 de noviembre de 2024.

[22] HTTP – Concepto, para qué sirve y cómo funciona. https://concepto.de/http/. Consultado el 4 de noviembre de 2024.

[23] Protocolo de control de transmisiones (Transmission Control Protocol) – IBM. https://www.ibm.com/docs/es/aix/7.1?topic=protocols-transmission-control-protocol. Consultado el 4 de noviembre de 2024.

[24] Figma. https://www.figma.com/es-es/. Consultado el 10 de diciembre de 2024.

[25] v0.dev https://v0.dev/. Consultado el 27 de diciembre de 2024.

[26] React. https://es.react.dev/. Consultado el 27 de diciembre de 2024.

[27] Vite. https://es.vite.dev/. Consultado el 27 de diciembre de 2024.

[28] Esmodules – Medium. https://medium.com/codecoolture/es-modules-qu%C3%A9-son-y-c%C3%B3mo-puedes-empezar-a-utilizarlos-e88c49043593. Consultado el 27 de diciembre de 2024.

[29] Hooks – Openwebinars. https://openwebinars.net/blog/react-hooks-que-son-y-que-problemas-solucionan/. Consultado el 27 de diciembre de 2024.

[30] Node.js. https://nodejs.org/es. Consultado el 27 de diciembre de 2024.

[31] Express js. https://expressjs.com/. Consultado el 27 de diciembre de 2024.

[32] Axios. https://axios-http.com/es/docs/intro. Consultado el 27 de diciembre de 2024.

[33] Swagger. https://swagger.io/. Consultado el 30 de diciembre de 2024.

[34] ¿Qué es API RESTful? – Amazon. https://aws.amazon.com/es/what-is/restful-api/. Consultado el 30 de diciembre de 2024.

[35] Programming in AL – Microsoft. https://learn.microsoft.com/es-es/dynamics365/business-central/dev-itpro/developer/devenv-programming-in-al. Consultado el 30 de diciembre de 2024.

[36] Power Platform – Microsoft. https://www.microsoft.com/es-es/power-platform. Consultado el 30 de diciembre de 2024.

[37] Power Automate – Microsoft. https://www.microsoft.com/es-es/power-platform/products/power-automate. Consultado el 30 de diciembre de 2024.

[38] MariaDB. https://mariadb.org/. Consultado el 31 de diciembre de 2024.

[39] React Design Patterns: The Container/Presentational Pattern – Medium. https://medium.com/@vitorbritto/react-design-patterns-the-container-presentational-pattern-775b91aa0c49. Consultado el 2 de enero de 2025.

[40] React Design Patterns: Provider Pattern – Medium. https://medium.com/@vitorbritto/react-design-patterns-provider-pattern-b273ba665158. Consultado el 2 de enero de 2025.

[41] Middleware in Node js – Medium. https://medium.com/@arorashivansh2661992/middleware-in-node-js-ed4eee917ff0. Consultado el 2 de enero de 2025.

[42] Using OAuth to authorize Business Central web services – Microsoft. https://learn.microsoft.com/en-us/dynamics365/business-central/dev-itpro/webservices/authenticate-web-services-using-oauth. Consultado el 3 de enero de 2025.

[43] Qué es el SSH y cómo funciona – Axarnet. https://axarnet.es/blog/ssh#:~:text=El%20SSH%2C%20Secure%20Shell%2C%20es,de%20seguridad%20a%20los%20usuarios. Consultado el 7 de enero de 2025.

[44] Putty. https://www.putty.org/. Consultado el 7 de enero de 2025.

[45] FTP: qué es y cómo funciona – Xataka. https://www.xataka.com/basics/ftp-que-como-funciona. Consultado el 7 de enero de 2025.

[46] WinSCP. https://winscp.net/eng/download.php. Consultado el 7 de enero de 2025.

[47] Cómo usar PM2 para gestionar una app en Node.js – Clouding.io. https://help.clouding.io/hc/es/articles/4402736917394-C%C3%B3mo-usar-PM2-para-gestionar-una-app-en-Node-js. Consultado el 7 de enero de 2025.

[48] Registro A de DNS – CLOUDFLARE. https://www.cloudflare.com/es-es/learning/dns/dns-records/dns-a-record/. Consultado el 7 de enero de 2025.

[49] CLOUDFLARE. https://www.cloudflare.com/es-es/. Consultado el 7 de enero de 2025.

[50] nginx. https://nginx.org/en/. Consultado el 7 de enero de 2025.

[51] Cómo funciona el cifrado TLS/SSL – GeoTrust. https://www.geotrust.com/es/how-does-tls-ssl-work. Consultado el 7 de enero de 2025.

[52] Certbot. https://certbot.eff.org/. Consultado el 7 de enero de 2025.